Body Sensor Networks Reliability Enhancement Algorithms in Sport and Biomedical Applications

Author

Alhusseinawi, Haider

Published

2017-05

Thesis Type

Thesis (PhD Doctorate)

School

Griffith School of Engineering

DOI

https://doi.org/10.25904/1912/3888

Copyright Statement

Downloaded from

Griffith Research Online

Body Sensor Networks Reliability Enhancement Algorithms in Sport and Biomedical Applications

Haider Sabti [Al-Husseinawi]
B.Eng and M.Eng

School of Engineering
Science, Environment, Engineering and Technology
Griffith University

Submitted in fulfilment of the requirements
of the degree of Doctor of Philosophy

May 2017

ABSTRACT

Wireless Networking is the fastest growing segment of the networking industry. In medical and sport industries, Body Sensor Networks (BSNs) are used to monitor the human body. BSNs have numerous design challenges, including network topology, reliable data delivery in a timely manner, and minimal power consumption. Other performance parameters, such as throughput and coverage, are equally important.

The object of this project was to achieve an efficient and optimal design of a BSN that acquire movement and physiological data, using sensors at various places on the body and sending data to a receiving unit (gateway) located on the chest for re-transmission of information to a remote computer (locally or in the cloud) for further processing and presentation. This is important in sport for the athletes, coaches and the viewing public, it provides real-time monitoring for rapid improvement in player performance during training or rehabilitation sessions. An investigation into the best node locations on the human body was made to achieve maximum connectivity to a receiving unit (gateway) on the chest, and the best angular window for the nodes to transmit data during typical human movements, such as running and walking. Preliminary results showed that, while the distance between the transmitting and receiving nodes changes significantly, the presence of scattering from limbs causes the most significant effect on the received signal strength. These measurements demonstrated that a received signal strength greater than -70dBm (the radio communications threshold) can vary from 20% to 74% of the recorded time for different node locations. The use of an accelerometer sensor at each node allows these positions to be identified in real-time for burst transmission to occur reliably. Wireless accelerometer sensor modules were used to determine the link performance by recording the data and traffic lost on different runners and for different transmitter locations on the human body (foot, leg and arm), to identify these time windows from the diverse angles of rotation of the human limbs during running. The results showed that the sensor on the wrist gives the best connectivity. An approximate swing time calculation algorithm was employed to find the swing time effect on these losses. Different data rates were tested against traffic loss and showed 98% and 62% of reliability at 250kbps and 2Mbps respectively.

With a central node on the chest, a novel energy-efficient time multiplexing transmission method for on-body wireless communication was implemented based on the human rhythmic movement of running. The running style of each individual allows the network to self-calibrate the communication scheme so that transmissions occur only when high link reliability is predicted. This technique takes advantage of the periodic running actions to implement a dynamic time division multiple access (TDMA) strategy for a five-node body network with very little communication overhead, long sleep times for the sensor transceivers and robustness to communication errors. The results showed that all wireless communications were successful, except when two nodes attempted to use the transmission medium simultaneously. An aggregated network reliability of 90% was achieved, compared to 63% when employing traditional time multiplexing algorithms. The results also showed a trade-off between the channel occupancy and traffic generated to provide high channel reliability for the body network. An advanced gesture transmission technique was adopted to collect acceleration data and to predict the best limb position for communications while the athlete is moving. This reduced the overall transmission power and increased the reliability. As a result, data losses were reduced from 30% to 1%, compared to continuous communications.

Experimental measurements were reported on five human subjects moving in formations on a grassed field with a smart algorithm that takes advantage of the Carrier Sensing Multiple Access (CSMA) technique. The algorithm forces the sensor nodes to repeatedly sense the carrier frequency to recognize when other nodes are in or out of coverage. The test reports different sink node locations on the body and the effect of transmitter power on the network reliability. A sink placed on the head position provides a normally distributed coverage of approximately 5 meters with link reliability of at least 80%. The smart algorithm showed 100% successful wireless communications between sink nodes within coverage when the nodes were programmed not to use the transmission medium simultaneously. The neighbours list and time of existence of all neighbours for each node were recorded and modified accordingly at each stage of the test. The availability of this information in real-time can be used to determine athlete proximity in the playing field and their performance with respect to time spent on one or multiple locations on the field.

STATEMENT OF ORIGINALITY

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

(Signed)______[redacted]______

Haider Sabti [Al-Husseinawi]

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACK	Acknowledgment
ADC	Analogue to Digital Convertor
AM	Amplitude Modulation
AoA	Angle of Arrival
ASIC	Application-Specific Integrated Circuit
BER	Bit Error Rate
BSN	Body Sensor Network
CDMA	Code Division Multiple Access
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTP	Collection Tree Routing Protocol
CW	Continuous Wave
D-MAC	Duty Cycle Medium Access Control
DSP	Digital Signal Processor
DSSS	Direct Sequence Spread Spectrum
ECC	Elliptic Curve Cryptography
ECG	Electrocardiography
EE	Energy Efficiency
EEG	Electroencephalogram
EEPROM	Electrically Erasable Programmable Read-Only Memory
EIRP	Equivalent Isotropically Radiated Power
EMG	Electromyography
ETSI	European Telecommunications Standards Institute in Europe
FCC	Federal Communications Commission in North America
FDMA	Frequency Division Multiple Access
FHSS	Frequency Hopping Spread Spectrum
FM	Frequency Modualtion
FoA	Frequency of Arrival
FPGA	Field Programmable Gate Array
FSTS	Fairness Spatial TDMA Scheduling

FTSP	Flooding Time Synchronization Protocol
GPRS	General Packet Radio Service
GSM	Global System for Mobile communication
GUI	Graphic Unit Interface
HIPAA	Health Information and Portability Accountability Act
H-MAC	Heartbeat driven Medium Access Control
I/O	Input / Output
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
ISM	Industrial, Scientific and Medical Bands
LAN	Local-Area Network
Li-poly	Lithium Polymer
LOS	Line of Sight
MAC	Media Access Control
MCOT	Mobile Cardiac Outpatient Telemetry
MCU	Microcontroller Unit
MEDiSN	Medical Emergency Detection in Sensor Networks
MEMS	Micro-electromechanical systems
MICS	Medical Implant Communication Service
MIT	Massachusetts Institute of Technology
NLOS	Non-Line of Sight
NO-ACK	No Acknowledgment
OFDM	Orthogonal Frequency Division Multiplexed
PC	Personal Computer
PCB	Printed Circuit Board
PDA	Personal Digital Assistant
PER	Packet Error Rate
PID	Packet Identification
PMs	Physiological Monitors
QoS	Quality of Service
RBS	Reference Broadcast Synchronization
RF	Radio Frequency
RPs	Relay Points

RSSI	Received Signal Strength Indicator
Rx	Receiver
SBST	Software-Based Self-Test
SMA	Sub Miniature version A
S-MAC	Sleep Latency Medium Access Control
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
SpO2	Saturation of Peripheral Oxygen
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
TDMA/CCA	Time Division Multiple Access with Clear channel assessment
T-MAC	Timeout Medium Access Control
ToA	Time of Arrival
TPSN	timing-sync protocol for sensor networks
Tx	Transmitter
UART	Universal Asynchronous Receiver/Transmitter
UHF	Ultra High Frequency
UMTS	Universal Mobile Telecommunications System
UWB	Ultra-Wide Band
UW-BCN	Ultra-Wideband Body Centric Network
VHDL	VHSIC Hardware Description Language
VHF	Very High Frequency
WBSN	Wireless Body Sensor Network
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WMTS	Wireless Medical Telemetry
WSN	Wireless Sensor Network

ACKNOWLEDGMENTS

It is a humbling experience to acknowledge those people who have, mostly out of kindness, helped along the journey of my PhD. I am indebted to so many for encouragement and support.

I would like to express my heart-felt gratitude to my principal supervisor Professor David Thiel for his encouragement, guidance and patience; I am one of the fortunate few to have an accepting and supporting supervisor as Professor Thiel.

My sincerest thanks are extended to my associate supervisor and the head of the Electrical and Electronic Department Associate Professor Steven O’Keefe for his time and advice. Ms Lynda Ashworth for her substantial administrative support.

My friends and colleagues in the science research group are thanked for their help, valuable discussions and time we spent together.

To my family, you should know that your support and encouragement was worth more than I can express on paper.

My parents you have always placed a great deal of trust in me and made me who I am, thank you for planting big dreams in a small child head.

My wife and son who joined me in this lengthy trip, thank you for being home. I feel proud in what we have achieved, grateful and blessed to have you on my side.

My brothers and sisters who believed in me and supported me in every way possible, thank you for your endless love and motivation.

CHAPTER 1

INTRODUCTION

1.1 Background

Recently, there has been increasing interest from researchers, system designers, and application developers on a new type of network architecture generally known as Body Sensor Networks (BSNs) or Wireless Body Area Networks (WBANs), that is feasible thanks to the advances on lightweight, small-size, ultra-low-power, antenna design, communications strategies, and intelligent wearable sensors. A Body Area Network is formally defined by IEEE 802.15.6 as, "Short-range, wireless communications in the vicinity of, or inside, a human body (but not limited to humans) are specified in this standard" [1]. In more common terms, a BSN is a system of devices in close proximity to a person’s body that cooperate for the benefit of the user. A BSN consists of a collection of small sensor nodes placed around the body. These may be attached directly to the skin or as part of special clothing [2]. Each node has a small power source and the nodes collectively communicate with a central node (like a Personal computer or PDA), which in turn can connect to the internet or local host computer. The data from the sensors can be relayed to a particular application or person for monitoring and further processing.

These devices have facilitated new levels of wireless interaction with the environment and between humans. The applications for BSNs are wide and vary in nature, including patient monitoring, sports, gesture recognition (capturing human movement), biomechanical analysis of movement, physiological measurements and so on; a complete survey of BSNs is presented in [3]. Utilizing wearable or implanted BSN in sports not only offer mobility for users, but also provide real-time feedback by continuously monitoring and analyzing sensed information. This can help professionals to effectively measure the athlete training sessions and perform early diagnosis and treatment to prevent injuries and support the rehabilitation process. Many studies have been carried out to provide patients with the best and most comfortable possible care, to closely monitor the athlete performance in sport fields and for assisted living and entertainment [4-7]. “Sports bands” worn on the wrist are now a popular accessory for amateur and professional athletes. They are used to monitor and interpret movement (accelerometers and gyroscopes) and position (GPS, magnetometers). Some parameters, such as heart rate, expired breath, and foot impact, require an array of various sensors distributed across the body (e.g. chest, shoe). The use of a hub in such a wireless sensor network (WSN) is necessary for both data synchronization and data storage. The hub can also be used for off-body communications to a coach or way-point in a long-distance race. In non-sporting activities, the general health and well-being of a person may require 24/7 monitoring and consequently, very high data transfer and cloud storage are required to consider it as an option. These advances have been possible by improved battery technology, low-power wireless transmission, advanced sensor technology and highly functional microcontrollers [1].

An example of real-time physical monitoring networks is presented in [8], in which the system helps classify the body activity by recognizing the angles calculated from the acceleration segments. Another study showed an investigation of the joint flexion angles using a knee goniometer application (KGA) developed on a smart phone. The knee range measurements reliability were validated using the standard universal goniometer (UG) [9].

Careful consideration of important characteristics of body sensor networks (BSN), such as sensing accuracy, long life time and network latency, is essential as weak wireless links and node failures reduce the system reliability. The physical dimensions minimize hindrance to the natural movement of the body while still supporting node placements [10]. One important problem associated with BSN is the effect of body movement on the wireless connection. For example, the absorption [7] and obstruction [8] of wireless signals through the body, and data collision [9] are common difficulties to be overcome. During normal body movements, the limbs interfere with the line of sight (LOS) of the transmitting and receiving nodes, which makes it difficult to sustain continuous wireless communications and affects the radiation pattern and polarization of the antennas [11]. In addition, there has been a lot of interest in the different antenna designs proposed to minimize this effect [12].

1.2 Research ethics

The expedited ethical review includes information that addresses the purpose of the study, the activities involved in the study, the screening criteria for participants, health risks to participants, and the expected benefits or findings of the study.

All experiments were conducted under Griffith Ethical Approval ENG/20/13/HREC and, to ensure that the volunteers are not injured during the tests, they were asked to run short distances at a moderate self-selected speed that was sufficient to reach that speed and record the acceleration data.

The experiments were designed to involve very little risk to the participants as it reproduces usual running movements at all levels of performance. In rare cases, muscle strains or fatigue can occur, in which case the study was stopped and medical action taken. To mitigate these risks, all activities conducted within the test are at submaximal effort levels through the normal range of motion.

Participants were recruited using word-of-mouth contacts and through gyms with sprinting classes. Depending on their level, participants were classed by their recent sprinting experiences. All participants were assigned an alphabetic letter to ensure their privacy and remain anonymous in all stored and published results, as shown in Table 1-1. To minimize health and safety risks, all participants had to answer the following questions:

1. Do you have any current injuries?

2. Do you have any current illnesses?

3. Have you experienced any injuries in the past 12 months which might affect your performance?

4. Have you ever had an injury illness or concern that may affect your safety in these exercises?

Normally, the setup of the motion sensor on the participant takes approximately 5 minutes for each person, with a further 10 to 15 minutes to ensure all devices are operate correctly. Each test took around 15 minutes and was followed by 5 minutes for sensor removal. Thus, for each participant, the full procedure took around 30 minutes, but depending on the number of sensors available, several participants were monitored simultaneously.

The participants were made aware that the experiment would be stopped in any of these cases:

1. If discomfort is felt at any time.
2. If they experience fatigue.
3. At any time or for any reason the participant can stop the experiment and request the gathered information to be deleted.

Table 1-1 List of participants who participated in this research and their level of experience.

Participant ID	**Age (years)**	**Height (cm)**	**Reach (cm)**	**Weight (kg)**	**Running experience**	**Training hours/week**	**Gender**	**Previous injuries**
A	28	175	158	85	2 years	3	Male	Yes
B	32	178	160	68	0 years	0	Male	No
C	22	170	165	70	3 years	1	Male	No
D	22	180	175	72	0 years	2	Male	No
E	23	182	179	69	0 years	1	Male	Yes
F	22	185	180	73	3 years	1	Male	No
G	26	180	180	70	0 years	2	Male	No
H	21	173	180	76	2 years	1	Male	No
I	29	159	155	55	0 years	0	Female	No
J	26	181	179	70	2 years	2	Male	No

1.3 Thesis Contributions

The main purpose of this work was to develop and evaluate measurement, monitoring and performance control functions for wireless body sensor communication systems that can capture and monitor human actions during the course of running. Small, smart wireless nodes were located on the human body and connected to a local network to provide both On-body and Off-body communications. As time synchronization is essential for data-combination from the multiple sensors, a method to compensate for clock drift between multiple wireless sensor nodes was required to increase the network reliability. Network reliability is the ability of a network to carry out a desired operation such as "wireless communication" and can be defined in terms of the speed and amount of information received. This is an important characteristic of any wireless network and can be affected by many parameters, including the number of nodes, overlapping transmissions and total amount of traffic generated on the network [13].

In [14], a prediction based on a routing algorithm is presented. The algorithm forces the sensor nodes to measure the quality of all incidental links to choose the forward link according to the highest predicted quality. The algorithm also employs a source authentication method to ensure network security while increasing the routing reliability.

The main Quality of Service (QoS) metric evaluated in this work is the delivery ratio (DR), which is a measurement of the network communication reliability. The result is a BSN which can provide real-time feedback, using movement-based time division multiplexing, to the user or professional personnel. The research objectives were:

- To modify the hardware structure of the wireless sensors node and develop a new software platform to provide real-time feedback of swing and stance time.
- To conduct static and dynamic measurements to evaluate the consistency of these values after many running cycles for different runners, and analyse the data symmetry between the right and left limbs during human action.
- To design a calibration algorithm for the wireless body network to allow calculation of the running parameters of different users, as they have different speeds and running style.
- To develop a dynamic gesture recognition algorithm for human movement detection and body monitoring.
- To examine the functionality of the designed algorithms when different communication technologies are applied, such as time and frequency multiplexing, gesture burst transmissions and carrier sensing coordinated by the human movement.
- To optimize the central node functions to provide both on-body and off-body communications, and investigate the ability to include multi-hop and dual-channel wireless communications between the sink nodes of different players to increase the system reliability.

However, there were many challenges that needed to be addressed:

- Network lifetime: Power consumption in a BSN is important since most or all devices are battery powered. Replacing or recharging in short intervals will be impractical, so power consumption is a major concern. As most energy is consumed by the radio, there are some limitations on the use of the radio transmission. After computation and communication is finished, a node should also be able to sleep completely (a sleep node is not transmitting or listening to any other node) and wake up at a predetermined moment using a hibernation technique. Radio transmission time is reduced by data compression and interpretation techniques.
- Synchronization and real-time feedback: The ability of the network to transmit data in real-time, without error, is important in this project. Time synchronization between two sensors in one BSN needs to work seamlessly.
- Network failure: Different mechanisms to ensure robustness are required. This ensure that the data frames get through to the correct receiver without collisions or other errors.
- Calibration: The usability of BSN by different users has always great importance. The sensor network should be able to monitor and report information without manually changing the BSN setting for each different user.
- Cost: To be able to compete in the international market it is essential that the components are at lowest possible price. This is very important when the product is mass-produced.

1.4 Chapters summary

Chapter 2

This chapter presents a literature review of the published work. It outlines the wireless body sensor network features and challenges and the large area of applications in which this type of network can be implemented. The ability of wearable sensors to provide an easy solution for capturing and processing athlete movements while eliminating the need for wired connections is reviewed. The related work in the area of monitoring physiological information during human action is given. Applications of BNSs in sport and biomedical fields were discussed fully.

Chapter 3

This chapter presents the developed wireless accelerometer-based device that has been used for monitoring the running activates and the adopted structure for storing the sensed data. It shows the accelerometer unit settings that can be adopted to provide a certain data reading at a specific sampling rate. Then it reports the work that has been conducted to analyse the performance of a wireless body sensor network during running activities for different transmission data rates, power settings and for different locations and participants. The swing time and the time difference between swing algorithms, based on experiments, are presented.

Chapter 4

This chapter outlines the software and the algorithms developed which addresses the key issues in relation to identify the best possible transmission link around the body to enhance the network reliability. It also presents the effect of retransmitting the lost signals on the network reliability and capacity. It shows the functionality of the developed calibration and dynamic transmission algorithms in recognizing important different participants' running activities and human gestures.

Chapter 5

This chapter presents the experiments that have been adopted on the sink node coverage to allow human to human/environment wireless connection. It shows the tracking and positioning Carrier Sensing (CS) technique that has been used to identify neighbour in dynamic environment, the multi-hop sink node transmissions and dual frequency body network operation. The developed feedback software transfers the gathered athlete data collected from the body network to a computer for analysing and understandable presentation.

Chapter 6

Summarises the key outcomes of this research and highlights the contribution made to the study field based on the experimental results. Possible future research work is presented in this chapter.

1.5 Publications

Some parts of this thesis research has been published, as shown below:

1. **H. A. Sabti** and D. V. Thiel, "A Study of Wireless Communication Links on a Body Centric Network during Running," *Procedia Engineering,* vol. 72, pp. 3-8, 2014.
2. **H. A. Sabti** and D. V. Thiel, "Movement based time division multiplexing for near real-time feedback body area network applications," *International Workshop on Antenna Technology (iWAT),* pp. 22-24, 2014.
3. **H. A. Sabti** and D. V. Thiel, "Node Position Effect on Link Reliability for Body Centric Wireless Network Running Applications," *IEEE Sensors Journal,* vol. 14, pp. 2687-2691, 2014.
4. **H. A. Sabti** and D. V. Thiel, "Time multiplexing-star shape body sensor network for sports applications," *IEEE Antennas and Propagation Society International Symposium (APSURSI),* pp. 969-970, 2014.
5. **H. A. Sabti** and D. V. Thiel, "Self-Calibrating Body Sensor Network Based on Periodic Human Movements," *IEEE Sensors Journal,* vol. 15, pp. 1552-1558, 2015.
6. **H. A. Sabti** and D. V. Thiel, "High Reliability Body Sensor Network Using Gesture Triggered Burst Transmission," *Procedia Engineering,* vol. 112, pp. 507-511, 2015.
7. R. M. Hagem, **H. A. Sabti**, and D. V. Thiel, "Coach-Swimmer Communications Based on Wrist Mounted 2.4GHz Accelerometer Sensor," *Procedia Engineering,* vol. 112, pp. 512-516, 2015.
8. **H. A. Sabti** and D. V. Thiel, "Wearable Wireless Sensor for Real-Time Angle Measurements," *Journal of Fitness Research,* vol. 5, Special Issue, pp.28-29, 2016.
9. Garry A. Einicke, **Haider A. Sabti**, David V. Thiel, and Marta Fernandez "Maximum-Entropy-Rate Selection of Features for Classifying Changes in Knee and Ankle Dynamics During Running", *IEEE Transactions on Biomedical Engineering,* 2017.
10. **H. A. Sabti** and D. V. Thiel, “CSMA Identification of Moving Neighbours in Body Sensor Networks,” *Accepted in International Sports Engineering Association Conference, 2017.*

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter presents a comprehensive study on features and challenges of wireless body sensor networks for sport and biomedical monitoring applications. It shows related work in the area of monitoring biomedical and physical signs of patients and athletes during the action. It also reports work that has been conducted to analyze the performance of wireless body sensor network for identifying the best possible transmission link to enhance the network reliability for on and off-body communications. The most important challenges for efficient real-time monitoring body sensor networks with reliable communication are introduced and thoroughly discussed. The chapter focuses on presenting new studies, techniques and novel designs to address these challenges. The different routing techniques have been investigated, to identify the best possible routes for nodes to successfully deliver information with minimum transmission overhead. Different methods developed to allow longer sleep times and change the radius of wireless transmissions in order to find the optimum network routing are discussed. The synchronization issues have been studied and presented for the body network specific requirements to allow real-time monitoring information. The extended network lifetime and reduced nodes power consumption were reviewed as an important feature of any network design, and different algorithms were shown to overcome this problem.

2.2 Body sensor networks architecture

Wireless sensor networks (WSNs) are normally composed of spatially distributed autonomous sensors that are used to cooperatively monitor physical or environmental conditions. Typically, WSNs contains thousands of tiny sensor nodes that communicate over wireless channels and perform distributed sensing and collaborative data processing. However, a specific application of these networks involves the placement of several smart sensors over the human body, known as a Body Sensor Network (BSN). These networks require careful consideration with respect to the designed application.

Comparing the body sensor networks with other wireless networks, it is crucial to consider the effect of the body on these devices. Body networks involve the placement of several sensor nodes inside or on the body to allow human-computer interaction. Figure 2-1 shows a typical body sensor network, which consists of multiple nodes distributed on the human body. Each sensor node has a specific function according to the application and depending on the required parameter to monitor, sensor nodes collect the data and may perform some computational analysis on the data before forwarding it to a central node, called a sink node [15, 16]. The number of sensor nodes and the application type actually affect the network topology to be used. Normally, there are one or more sink nodes located in a central position that acts as a hub to collect all captured (sensed) parameters of interest and forward them to the personal server or a router to provide an interface towards other networks or professional staff. Often the sink will be equipped with a larger battery, together with a display or an uplink to a cellular network [17].

Body sensor networks are limited by the scarce energy source, memory and processing capabilities, and by the node distribution, lifetime, and operating protocol. There are many wireless features of BSN that appear comparable to those used in WSNs, such as the network architecture, energy control, radio technology, network security, and communication protocols. However, the influence of the human body on the wireless devices significantly affects the radio communication [2, 18]. Moreover, the locations of nodes on the body and, more specifically, the paths to other nodes have a large impact on the channel quality. As such, a slightly different posture can result in significantly different channel characteristics, such as when the line of sight communication is impossible. This means that different people will also have different channel quality, as both the shape of the body and the type of movement have influence on the channel reliability [19, 20].

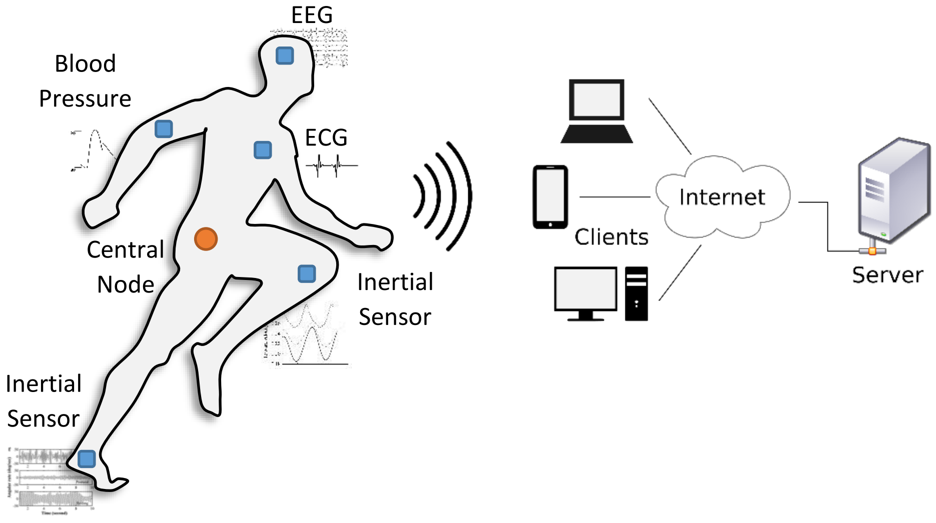


Figure 2-1 Body sensor network structure.

The variety of commercially-available sensors and ongoing miniaturization of hardware embedded devices and sensors have made possible the development of more practical on-body communications during human movement actions using wearable wireless nodes [15, 17]. These intelligent sensor nodes capture various physiological signals of medical interest. For example, an ECG sensor can be used for monitoring heart activity, an EMG sensor for monitoring muscle activity, an EEG sensor for monitoring brain electrical activity, a blood pressure sensor for monitoring blood pressure, a tilt sensor for monitoring trunk position, a breathing sensor for monitoring respiration, while the motion sensors can be used to discriminate the user's status and estimate her or his level of activity [21-24]. The hardware of a sensor node usually consists of five major components, namely the microcontroller, RF module, flash memory, sensors and a power source [25]:

- Embedded Processor: it is responsible for processing the data, scheduling the node tasks and manage the functionality of other hardware components. There are different types of processors available for sensor nodes such as Microcontroller, Digital Signal Processor (DSP), Field Programmable Gate Array (FPGA) and Application-Specific Integrated Circuit (ASIC). All can be configured using different programming languages such C, C++, Verilog, VHDL, etc. The microcontroller is the preferred processor for sensor node applications because of its flexibility to connect to other components and low price.

- Transceiver: it is responsible for the wireless communications between nodes. The wireless transmission medium can vary in frequency from radio frequency (RF), laser and infrared.
- Memory: there are two different types of memories in a sensor node, internal (in-chip memory) and external memories. Internal memories are used to save the programming code for the Microcontroller and external memories are used to save collected data.
- Power source: it defines the sensor lifetime and the ability to work probably. The most common power sources are small rechargeable batteries. However, there are alternative sources for providing energy, such as solar and wind.
- Sensor: A sensor is a converter that measures a physical quantity and converts it into an electrical signal which can be analysed and processed. There are innumerable types of sensors used in machines, aerospace, sports, medicine, manufacturing and robotics that are usually calibrated according to known standards.

2.3 Advantages and limitations of using BSN

There are many advantages associated with the use of wireless body networks. Wireless sensors can make a huge difference in terms of providing accurate data when compared to traditional examination methods. Sensors can detect real-time information and send measurements in seconds to be viewed through computer systems. The sensed physiological data can be displayed, saved and retrieved through the sensor network to ensure high reliability and accessibility of information. In some cases, the measurements gathered in BSNs are sensitive to the subjects being monitored; therefore, privacy and security in BSNs can promise only authorized access to information. BSNs allow increased range of human movements during the monitoring process since there is no physical attachment required to a monitoring machine and all information are transferred wirelessly through an air interface. This, provides support for continuous monitoring in working/training environment and acquiring more realistic data compared to laboratory measurements or simulated data. BSNs help to provide emergency response, send direct feedback for the connected subjects, and enables early detection of unusual behavior with respect to the information being gathered. BSNs provide cost effective services that help to save time and reduce the load of working professionals [26]. On the downside, network faults or connection loss is a serious problem, since it affects the process of transmitting the monitored parameter. This is especially important in health-related applications, where patients are at risk due to loss of important information. The intermittent and transient faults are most common sources of failure for body sensor networks. According to [27, 28], more than 80% of faults in BSNs are results of intermittent and transient faults. The software and hardware of the faulty system may exhibit intermittent faults. This is a special case of the transient fault that can originate from inside the system. A transient fault is usually produced by external agents, such as heat, rain, humidity, electromagnetic radiation (interference), etc. The objective of BSNs is to ensure and encourage the mobility of the users; therefore, body networks are required to be capable of configuring and maintaining themselves automatically, i.e. self-organization and self-maintenance should be supported. This is a very hard task to be achieved, especially for implanted sensors, since there is no sensor working indefinitely without maintenance.

2.4 Technical issues and challenges for BSN design

Unlike regular wireless sensor networks, there are specific challenges associated with human body monitoring. The greatest difference is the need for high communication reliability with each sensor node, as opposed to the redundant characteristics of WSN nodes. This corresponds to the typical biomedical application of BSNs, where only a single sensor per vital parameter is used. Moreover, the scale of BSNs is very small compared to classic large scale deployments of WSNs. Body network designers normally deploy up to twenty nodes on a single person, while the designed protocols of wireless sensor networks typically support hundreds of nodes displaced over hundreds of meters. A comprehensive comparison between the technical design requirements and challenges for these networks is shown in Table 2-1. The most important parameters for the design consideration of wireless body sensor networks are described in the following sections [15].

Table 2-1 Schematic overview of differences between WSN and BSN [19].

Challenges	**Wireless sensor network**	**Wireless body sensor network**
Scale	Environment (m or km)	Human body (cm or m)
Node number	Many nodes in wide area	Fewer, limited in space
Result accuracy	Node redundancy	Node accuracy and robustness
Node tasks	Performs a dedicated task	Performs multiple tasks
Node size	Size is not important	Size is essential
Network topology	Likely to be fixed or static	Dynamic due to body movement
Data rates	Most often homogeneous	Most often heterogeneous
Node replacement	Easy, disposable nodes	Challenging for implanted nodes
Node lifetime	Months or years	Hours or days, smaller battery
Power supply	Accessible and frequently replaced	Inaccessible and difficult to replace in an implantable setting
Power demand	Large, supply is available	Lower, supply is limited
Energy scavenging	Solar and wind power	Motion and thermal output of body
Biocompatibility	Not a consideration in most applications	A must for implants and some external sensors
Security level	Lower	Higher, to protect information
Impact of data loss	Compensated by redundant nodes	High QoS and real-time data delivery are required
Wireless technology	Bluetooth, ZigBee, GPRS, WLAN, …	Low power technology required

2.4.1 Energy-efficient networks

The power consumption determines the lifetime of the wireless sensor node. Each hardware component has different power requirements and, normally, wireless body networks use an electric energy source, such as small batteries. The major consumption of energy in a body sensor network lies in communications. In such a system, nodes communicate between each other, which requires significant amount of energy. Replacing or recharging the batteries in short intervals will be impractical; therefore, several promising techniques have been explored to minimize the power consumption and extend the network lifetime, such as low-power listening [29] and wake-up radios [30]. Both are based on reducing the power consumption by keeping the transceiver is in sleep mode (idle mode).

Proper design of the MAC layer protocol increases the lifetime of a node by limiting the energy consumption in wireless communication [31-34]. The major energy loss in wireless sensor network is due to collisions, idle listening, overhearing, traffic fluctuations and protocol overhead. These sources of wastage can be eliminated by using a master-slave architecture with time division multiple accesses with clear channel assessment (TDMA/CCA) network access scheme [21]. The main goal is to reduce power consumption contributions from idle listening, overhearing, and collision; this can be achieved by turning off the radio from time to time to reduce the duty cycle. S-MAC [32], T-MAC [35] and D-MAC [36] are proposed to solve the idle listening problem. All of these different techniques work to apply a synchronized duty cycle schedule for all sensor nodes. In [37], the problem of energy expenditure for centered body sensor networks was addressed. The proposed network protocol operates using a Time Division Multiplexing Access (TDMA) technique, allowing the sensor nodes to communicate only at their allocated time bursts and sleep the rest of the time. This method helps to preserve the energy with very little communication overheads by controlling the timing through the proposed MAC layer protocol. Another work [38] discussed the issue of contextual sensing to save energy. Here, the sensor nodes only join the network and transmit data following positive event detection (events based on context), with an online system used to alert observing professionals. This method cuts the workload and stretches the network lifetime by sensing conditional events.

Additionally, the lifetime of a sensor can be increased by optimizing the power consumption [39]. A simulation measurement to optimize the energy consumed by embedded processors using a Software-Based Self-Test (SBST) was discussed in [40]. This method leads to extend and maximize the energy gains for WSN architectures by offering dynamic frequency scaling features and clock gating to reduce the power consumption in individual nodes. The result of the simulations shows that the energy saved at the processor level is approximately 37%. Improving the lifetime of the battery will also increase the lifetime of the wireless sensor network. Although there is much effort on designing low-power sensors, there are other energy sources used, such as scavenging techniques. A combination of lower energy consumption and energy scavenging is the optimal solution for achieving autonomous wireless body area networks. Solar cells can provide sufficient energy under direct sun; however, they may not perform adequately when in shade that is resulted from body movements, and will limit the options of sensors placement on body (such as under clothing). More promising methods are motion [41] and body heat [42] energy scavenging techniques that can fit the needs of body-centric networks.

2.4.2 Network reliability and quality of service (QoS)

One of the main challenges in BSNs is their effectiveness in conveying important information through reliable wireless communication while maintaining an acceptable Quality of Service (QoS) performance. Proper QoS handling is an important part of wireless transmission in order to guarantee that the monitored data is received correctly especially when requested by health care professionals. Network reliability and the wireless network performance are driven by the network’s ability to successfully handle traffic and is affected by many parameters, such as operating frequency, transmitter power, receiver sensitivity, channel data rate and the distance between the wireless nodes [13, 43]. The reliability can be considered either end-to-end or link-based. An investigation based on the radio propagation performance of an Ultra-Wideband Body Centric Network (UW-BCN) for multiple on-body nodes and body postures was reported [44]. The experiments were conducted by adopting the Orthogonal Frequency Division Multiplexed (OFDM) model, and results were analysed by monitoring the Bit Error Rate (BER) and Signal to Noise Ratio (SNR) to measure the system performance. It was found that a BER for on-body links of equal to or less than 0.1% can be achieved for a number of transmitter locations. Therefore, the study provides suggestions based on the node locations for an optimal link quality.

The QoS requirements in body sensor networks are tightly related to the context of its application [45, 46]. Unfortunately, in body sensor networks, the propagation of the radio waves is affected by the presence of the human body as an obstacle [11]. During the natural human movements, the body limbs can interfere with the line of sight (LOS) propagation path between the transmitting and receiving nodes, resulting in a multipath fading and scattering effect of the transmitted signals, which affects the network reliability. In [11], a comprehensive indoor multipath statistical analysis was made on the human body effect in a narrowband body sensor network. The experiments included stationary and mobile scenarios for body-worn sensors analysed in an anechoic chamber and in a typical indoor environment. The wireless nodes were operating in the 868MHz band, and the fading channels were analysed using different propagation models, such as Nakagami and Rician fading [47], to determine the optimal fit. The final results showed that mobile movements can increase the multipath disturbance, as compared to stationary arrangements.

In the emergency monitoring applications, such as human survivor location in disaster recovery situations, the QoS is mostly needed to make sure that information reaches its destination with the and minimum latency. For example, it has been claimed that latency (time delay) must less than 125ms for the medical applications and 250ms for the non-medical applications. Jitter (variation in latency) also must be controlled [3]. These applications may utilize different optimization techniques and uses whatever recourses available to characterize priorities and ensure high QoS. This does not only affect the communication layer, but also other resources including energy and processing. A study on the Quality of Service (QoS) performance in a medical wireless body sensor network was made in [46]. An access protocol was proposed to prioritize the critical from the non-critical health information. This protocol classifies the patient health packets in the MAC layer and assigns the level of importance to these packets before routing through the network. It also cuts off the retransmission of less important (non-critical) packets to create additional channel space for urgent health information in case of network collision. This increases the QoS for only the critical traffic, while maintaining a specified QoS level for non-critical packets. The designed protocol was tested through simulation and resulted in a better QoS and larger throughput for the prioritized packets.

2.4.3 Security and privacy

The privacy laws and regulations for wireless standards in sports and biomedical applications raised serious concerns with regard to the body sensor network security in terms of safe and proper use of information restricted to authorized persons. The Health Information and Portability Accountability Act (HIPAA) in United States, and the European Union Directive 2002/58/EC prohibit unauthorized access to information and recommend the use of security keys to prevent piracy listening on the wireless channels that contain important data [36]. Security and privacy protection mechanisms require frequent use of the wireless network links to authenticate and encrypt the information. While this uses a significant part of the available energy, it's considered as an essential part of the BSN and cannot be neglected. Different applications have different security requirements; for example, military applications cannot afford exposure of sensitive information or giving access to an important network resource, while training and activity monitoring in sports network applications only require the verification user's identification. The security requirements of the overall system are represented by the confidentiality, data integrity, accountability, availability, and access control. Where encryption methods are used to guarantee these security requirements, there is a challenge of developing efficient key management protocols.

Many encryption mechanisms proposals for key distribution protocols were developed. A method for enforcing security and patient privacy using 802.15.4 sensor nodes was reported in [48]. An Elliptic Curve Cryptography (ECC) key distribution was proposed, where a low power microcontroller was used as a security processor in each sensor node to employ the ECC signals as the security key for different patients. This enabled unique key distribution by providing the unique ECC print for that specific patient. In [49], the timing information of the patient's heartbeats was used to identify individuals in secured body sensor networks. The proposed security measure was based on the analysis of the patient's biometric trail (inter-pulse intervals of the heartbeat in this case), unlike the traditional security distribution key methods that involve the continuous creation and update of initial keys for body sensors. The scheme performance rates were calculated based on the successful identification of candidates. The overall results showed that the timing identification scheme can be a good solution for security in body sensor networks. In [50], a physical signal based authentication scheme was proposed. The system generates identity information for mutual authentication from the unique body biometric characteristics.

On the other hand, users' authorization and privacy control over their data can still be hacked. Potential physical-layer privacy attacks on wireless ubiquitous systems presented in [51] show privacy flaws which enable the attacker to monitor the log of daily living activities in smart homes and assisted living facilities by the use of two types of information known as "timestamp" and the "fingerprint" of each radio message. The proposed work suggests that the Received Signal Strength Indicator (RSSI) value of transmitting signals is used to identify the presence of activity transmission, and the frequency of transmission and duration of each transmission are used to classify the activity type. The authors also developed an algorithm that follows a number of design guidelines to enhance privacy. The suggested solutions include the use of a signal attenuator outside the house or in rooms to mask the user's activities from attackers. Also a random transmission delay and fake data transmission can cover the periodic transmission of data. While the authors proposed a guideline to enhance user confidentiality from eavesdropping attacks, privacy breaching algorithms and their encountered measures are being developed on a daily basis.

2.4.4 Usability and user-friendlessness

For most of the BSN applications, the network ought to be capable of configuring and maintaining itself by design (i.e. self-organization and self-maintenance must be supported); for example, in sports and biomedical applications the aim of BSN is to ensure and encourage the mobility of the users to live independently with the assistance of high-quality healthcare services. Although BSNs enable unconstrained movement of the athletes compared to wired sensors [52], they may add some movement restrictions. The size and weight of sensor nodes, in addition to the number of sensors carried by the human participant, are important features that have influence on the user comfort and mobility. The nodes may be distributed around or implanted in the human body and can largely affect the subject's natural movement. Many researchers are developing compact designs and working to minimize the sensor weight and dimensions to obtain a full body measurement of the individual movement pattern using these wireless sensors, so that the coaches can easily locate weak points where more training is required. However, there is no universal sensor that can detect all signals and different sensors must be placed on different places to detect acceleration, temperature, heart rate, blood pressure, etc. The exact location of each device will depend on the application, e.g. a heart sensor is usually placed in the neighborhood of the heart, while a temperature sensor can be placed almost anywhere on the skin. Researchers seem to disagree on the ideal body location for some sensor nodes (i.e. motion sensors) as the interpretation of the measured data is not always the same [26].

In [53], a survey on the BSN user-friendliness among a small elderly group was presented. The experiments were conducted while they were performing everyday activities and only half of the tested group considered the system as user-friendly. The node size and weight reduction add limitations to the hardware and software complexity that is required to perform several tasks, and the ability to monitor different psychological data simultaneously. Network optimization at all levels is essential to ensure a user-friendly wearable body sensor network and natural interfaces with immediate response capabilities.

2.4.5 Data processing, fusion, and compression

In a wireless sensor network, the gathered information can be stored locally in the memory of the node or externally in a storage device. The process of handling requests for retrieving and/or saving information may consume a large portion of the node energy. Although the use of an external source allows further manipulation and processing of the data, it also involves a frequent use of the wireless medium to send requests and receive queries to/from the targeted central storage unit [54]. There are different approaches to tackle the problem associated with the resultant trade-off between limited storage space and high energy consumption; one way is to avoid recording raw data at the sensor nodes and looking for events of specific importance to save storage space on the local memory; another solution is given by accessing multiple external storage units within range such as the sink nodes to bring a balance in the overall network storage and escape the bottleneck end problem resulting from having only one storage centre. In [55], mobile nodes (referred to MULEs) were presented as dynamic and mobile storage nodes roaming the network space area to catch information when the wearer passes by the fixed sensor nodes. The short-range transmission between the MULEs and sensor nodes substantially reduces the power consumption. The paper provides a detailed discussion with regards to the number of deployed MULEs and their storage capacity according to the network performance, in terms of latency and success rate.

The node hardware design introduces some limitations on the network capabilities represented by sensing, storing and handling of information. For example, some network applications may require the measurements of multiple parameters where the final outcome is a fusion of these parameters (extracted summary), rather than the raw data. This largely increases the load on the network capacity and the hardware design may act as an obstacle to achieving its purpose. In such applications, the introduction of external storage units and data management methods is a necessity for delivering these requirements. The servers can help reduce both the storage space and processing load on the wireless sensor network. In [56], a multi-dimensional range query mechanism and in-network data storage structure were introduced. The purpose was to reduce the total budget of queries by introducing a load balance throughout the wireless networks. Targeted events are mapped from the source nodes in the corresponding space range using a General Packet Radio Service (GPRS) routing protocol. The overall results show a reduction in the average query cost and hotspot routing in large scale disperse sensor network. However, this technique relies on the use of GPRS, which consumes a large portion of the battery powered sensor node.

A distributed spatial-temporal data storage with similarity based searching functions was introduced to overcome the challenges of handling network aggregated quires with minimum produced searching latency [57]. This scheme proposes a zone classification of data where the zones are defined by the sensing time of information, attribute and geographical locations of sensors. All sensed data are mapped according to its zones and stored in a specific node in each zone. Adding a high level of classification to the stored data, allowed the targeting of only relevant sensor nodes in a congested network. This allows the user to monitor and retrieve information that has similarities in the particular zone with a small number of queries send throughout a zone based routing. While this technique shows a significant improvement compared to the traditional approaches for high capacity data fusion applications, such as environmental monitoring of habitat or ocean currents, advanced computing and localization algorithms are included in the scheme of dynamic networks where sensor nodes are moveable and transmission is unreliable.

2.5 Wearable sensors and wireless communications

Wireless communications solve the reliability and security issues related to communications and addresses applications where mobility and convenience are important. Unlike wired communication, there is no requirement for a transmission guide (such as conducting wires or optical fibers) to send signals [58]. However, there are power and frequency restrictions set by the Federal Communications Commission in North America (FCC) and the European Telecommunications Standards Institute in Europe (ETSI) that prevent the random use of frequency channels and transmit power for different applications such as AM, FM, VHF, UHF, etc. Several frequency bands, in particular, the industrial, scientific and medical bands (ISM), are license-exempt and can be used for low power and research purposes [59].

The continuous evolution of wireless communications is a result of the increasing needs of the quality of service, data rates and diversity of the offered services. Meeting the ever-expanding requirements requires innovations in architectures, protocols, spectrum sharing techniques, and interoperability between heterogeneous networks. In this context, wireless sensor networks represent a promising solution to enhance network performance with a pervasive coverage at low-cost and energy consumption that can be deployed in indoor or outdoor environments and is based on existing or emerging wireless network standards. The importance of the BSN in future communications resulted in a new standard (IEEE_802.15.6) regarding the use of this technology in different applications that involve smart sensors placed in or on the human body, to share information among themselves and external units [60]. Wireless communication within a wireless sensor network may be based on infrared, light, microwave radio and even near-field coupling through skin conductivity. Microwave radio communication, especially using ZigBee wireless technology, is a popular approach [61, 62]. The increasing use of wireless networks made this technology reliable for usage with other existing wireless sensor networks, by using a RF technology to interconnect in-body, on-body and off-body wireless sensors [63].

2.5.1 Communication protocols

Many wireless network standards are used in body sensor networks; each come with advantages and disadvantages when adopting with a specific sensor network application. Wireless body sensor networks may communicate externally with other networks (which may themselves be body sensor networks), using the available wireless technologies that cover a wide area and offer the possibility of ubiquitous worldwide wireless mobility [64]. These are short, medium and long-range communications, and include Bluetooth, ultra-wide band, Wireless LAN (Wi-Fi), WiMAX, GSM, GPRS, UMTS and satellite communications. The 802.15.6 standard considers effects on portable antennas due to the presence of a person (varying with male, female, skinny, heavy, etc.), radiation pattern shaping to minimize the Specific Absorption Rate (SAR) into the body, and changes in characteristics as a result of the user motions [60].

Risks must be considered and addressed in order to develop the best solutions for sensor monitoring. For example, some sensors may lose their power or connectivity due to long working time or large distances between neighbor nodes. Therefore, they will be an inefficient addition to the network and can be considered as losing endpoint (where all transmission attempts are lost). Proper network designs must report the faults of dead sensors in order to prevent losing vital information. Moreover, errors must be considered when using the transmission of data through a wireless system, such as Wi-Fi, Bluetooth or ZigBee [65]; to ensure a high network reliability.

2.5.1.1 Bluetooth technology

Bluetooth, is an open standard wireless technology for short distances. It allows a communication bandwidth of up to 1Mbps, which is more than sufficient for most intelligent sensors. It operates in the 2.4GHz industrial, scientific and medical (ISM) band, occupying 79 channels. The transmission range can extend to several meters depending on the transmitter used. The radio layer uses frequency hopping spread spectrum (FHSS) coding, to support low-latency and high-throughput operation. Bluetooth uses a channel hopping nominal rate of 1600 hops/sec [66]. This requires all devices to remain synchronized within a few microseconds. In wireless sensor networks, nodes may only send the data once per hour and it would be highly inefficient to force synchronization for the entire hour. A good example for the efficient use of Bluetooth technology in wireless sensor network is a small number of processors and nodes, which can communicate by using Ad Hoc techniques [36].

Bluetooth technology in BSN has the advantages of effective wireless communication of low cost and power. These features are attractive since they promise a high data rate from a single chip data acquisition system operating at nominal power [67]. Bluetooth provides a gross data rate of 1Mbps, which is enough data to enable multimedia real-time applications. It is a mature technology that is already integrated into many cell phones and PDA devices. Bluetooth uses the unlicensed 2.4GHz spectrum, which ensures zero operating costs for spectrum usage. However, there is little interest in Bluetooth sensor networks, as it might not be the perfect candidate for certain body sensor network applications. The complexity in demanding frequent hopping code for a master-slave topology adds more burdens to the operating network represented by the high-end synchronization demands for all nodes at all times to follow the same hopping scheme and large amount overhead communications between the nodes, this adds to the power demand and susceptibility to interference from other devices operating in the same frequency range.

2.5.1.2 ZigBee wireless technology

ZigBee is another short-range wireless communication technology which provides self-organized, multi-hop and reliable networking facility with long battery lifetime. It is based on the IEEE 802.15.4 standard, thus operates in the unlicensed RF bands at 2.4GHz (global), 915MHz (America) and 868MHz (Europe). ZigBee technology is already being used in many applications and electronic devices. It can be found in many house applications and hospital electronic sensors. Therefore, ZigBee can provide an easy integration with the surrounding wireless environment. Unlike Bluetooth, ZigBee does not require constant coding synchronization for the hopping algorithm. It uses a Direct Sequence Spread Spectrum (DSSS) to ensure hop-sequence synchronization which does not have to occur prior to initiating communication. This provides a distinctive advantage over the channel hopping mechanisms in Bluetooth technology, by allowing more snooze time for the sensors and ensures that synchronization is only required at the time of communications [68-70]. This is one of the main advantages regarding power consumption of low-duty cycle devices and addresses a major shortcoming of Bluetooth.

ZigBee networks use a star, mesh and tree topology where data are transmitted through a constructed route and might consume a lot of resources in terms of bandwidth and power consumption. There are three categories of ZigBee devices [69]; ZigBee coordinator that acts as a backbone of the network tree automatically initiates the formation of the network and maintains the overall network knowledge. It is the most sophisticated of the three devices, requiring the most memory and processing power. ZigBee router which is a smart node that links groups together and provides multi-hoping for messages to extend the network coverage. It associates with other routers and end devices. Routers not only increase the network coverage but also reliability. ZigBee end devices are battery-powered wireless nodes can only communicate to the routers or coordinators through the network tree. They carry limited functionality to control cost and complexity [71].

2.5.1.3 Wireless fidelity (Wi-Fi)

The most recognizable, set of protocols to be considered for use in BSNs are those based on the IEEE 802.11 family of standards [72]. The general Wi-Fi usage demonstrated by the 2.4GHz wireless communication protocols, granting there is a second band at 5.8GHz used in 802.11a-based networks. The main motivation for considering the Wi-Fi/WLAN approach is commercial: the rapid growth in mobile computing has led to low-cost wireless networking in both the home and the workplace and also made ‘wireless hotspots’ a common occurrence in public spaces such as coffee shops and shopping malls. This means that the hardware required is both plentiful and low cost.

The main advantages of having a BSN with WiFi communication capabilities are represented by the easy utilization of the existing WiFi networks and continuous access to other networks [73] allowing real-time uploading and streaming of information to or from the internet cloud. On the other hand, the WiFi technology increases the energy consumption through frequent and large data streams and overhead communications while connected to the internet and significantly affecting the overall network lifetime

[74]. A hybrid BSN-WiFi system design with an external link to a WiFi network through one (could be the sink node) or multiple hops can be used to solve the power limitation problem.

2.5.2 Network topology and MAC protocols

Nowadays, wireless networks have been introduced everywhere with different deployed topologies where sensors are used to monitor environment surrounding humans. A hybrid network design has also been introduced to allow communications between human bodies and around networks and sensors. The low powered and low memory sensors can manage only to send the captured information to the nearest node devices within communication range. These nodes forward the data in a one-hop or a multi-hop routing to its destination, depending on the used network topology and communication protocols.

The main reason for developing and deploying a specific Medium Access Control (MAC) protocol in body sensor networks is represented by the necessity of having a collision-free data transfer throughout the designed network topology. MAC protocols can significantly reduce the network overhead communication and major power wastages resulting from network collisions. Many wireless communication techniques have been adopted to increase the network capacity and enhance the reliability such as Carrier Sense Multiple Access (CSMA) [61], Time Division Multiple Access (TDMA) [37], Frequency Division Multiple Access (FDMA) [14], and Code Division Multiple Access (CDMA) [75]. The use of multiple access technologies provide consistent and burst signal transmission over time, frequency and code [76]; which can support the reliable communication for the wireless nodes without packet collision. However, these approaches are subject to energy cost, such as the idle listening phase in the case of CSMA; time synchronization required by standard TDMA which requires additional energy to send and receive timing information between all network sensors to get their clocks synchronized. A detailed comparison of the use of CSMA and TDMA in wireless sensor networks can be found in [77]. Furthermore, FDMA and CMDA require complex hardware to perform frequency hopping and frequency signal spreading techniques [37]. A comprehensive study on the design requirements of the MAC protocols for body sensors networks is presented in [78].

MAC layer routing protocol was developed to work efficiently with applications of body sensor network in [37]. The MAC protocol uses the TDMA approach to avoid power wastage resulting from the communication collisions and reducing the amount of unnecessary idle listening and communication overhead. The protocol controls the timing and transmissions between the master nodes and monitoring stations to ensure collision-free data transfer, with an additional time slot reserved for packet re-transmission to avoid major packet losses and increase the network robustness for unexpected communication errors. With little communications overhead and longer sleep times, this technique increases the overall network lifetime by reducing energy demands. A fairness spatial reuse of the Time Division Multiple Access (TDMA) techniques that combines a number of experiential algorithms have been developed to provide the best solutions for the transmission between nodes [79]. The proposed network topology consists of several wireless sensor nodes along with a multi-hop radio network, where the quality, distance to the sink node and working time of the link are different for each sensor. Therefore, every node in such a network is operating differently along with its conditions, which might be more problematic in a larger network. This solution can assign the resources of the network to the nodes in order to lead similar end-to-end delivery data rates. The simulation results of the fairness spatial TDMA scheduling (FSTS) showed that the maximum rate of transmission data within the network capacity and the difference in the end-to-end delivery rate was dramatically reduced.

The network topology can be defined by the BSN application requirements. There are many factors involved in shaping the network topology, such as the network size, spreading of nodes, network characterization (dynamic or static), deployed MAC protocols, surrounding environments, etc. Selecting the right topology can considerably enhance the network performance and prolong its lifetime by optimizing the number of hops required for information transmission and delivery. A demonstration of the significance of topology and hierarchical designs as a factor to enhance the body sensor performance in medical monitoring applications was presented in [80]. The integrating of the Body Sensor Networks (BSN) with Wireless Mesh Networks (WMN) was proposed, to improve network coverage and quality. The simulation results of the integrated network design involved the study of several limiting factors such the network latency, MAC delay and total throughput in various network environments and subject numbers. In [13], a study was carried out on the rate of packet delivery for a dense wireless sensor network (that consist of up to sixty nodes) in three different environments, including indoor and outdoor locations. The results indicated that pre-evolution and simulation of the network routing schemes and wireless protocols (such as medium access protocols) can play a significant role in the future design of sensor networks. Overall, the use of multi-hopping techniques and advanced routing schemes have were studied to increase the communication range of wireless networks.

2.5.3 Routing, spatial and collaborative transmission

The main functions of the wireless sensor network are detecting events, recording data and sending the information to a designated source. Losing data will affect the network reliability represented by the wireless feature. Nodes in the network can communicate between themselves and other networks; therefore, different networks utilize different topologies to ensure the successful delivery of information. In body sensor networks, data are usually routed through a central (sink) node or through neighbor nodes over multi-hop transmission to its destination. There are different techniques and algorithms used to define the routing rules for such networks. Mostly the surrounding environment and the application type impact largely on these routing settings.

The variable (spatial) node transmission radius is one technique used to influence the routing path chosen for a group of nodes and consequently to improve the wireless network performance. The issue of energy consumption has been addressed by implementing the optimal multi-hop routing in a wireless sensor network in [81]. The optimal routing is defined as a function of radio power transmission that is affecting the overall network lifetime. The designed node transmission range is affected by the surrounding neighbors for a certain network density. Selecting a higher transmission range to include more neighbors (increase the node degree) will save energy for the multi-tier network. Also, it will improve the communication efficiency and reduce the redundancy of multi-hops retransmission required for the node broadcast transmissions. However, there is a trade-off with this proposal, represented by the increased energy required at each node for extending the transmission ranges. In [82], a study on the impact of different variable-range transmission power control in wireless multi-hop sensor network was proposed. Each node in the system consistently controls its transmission range, to reduce the average traffic on the nodes carrying larger data loads to the system in a multi-hop network. A comparison between fixed (common) transmission range and variable transmission range based routing protocols has been made. The results showed that the common range transmission protocol can only carry half of the traffic capacity of the variable-range power control method. Therefore, common-range transmission based on routing protocols is limited in power capacity. This work has shown the need of considering variable-range power control design in BSNs. A self-organized node formation and optimal routing in a scalable large wireless sensor network were presented in [83]. A set of smart protocols and algorithms that support dynamic formation in highly constrained energy wireless sensor networks were introduced. The results show that the smart coding supports the functions of slow mobility for these nodes with regards to dynamic topology network designs and information processing between the resultant subnetworks.

In a sparse network configuration, node failures due to battery problems or any other reason throughout the network, change the network topology and introduce a difficulty in sustaining overall network connectivity. The same can be expected for dynamic networks where nodes change their locations continually. Cooperative transmission is one solution to provide reliable and energy-efficient body sensor networks by introducing the concept of cooperative diversity, where nodes in a sparse setting network can transmit the same information cooperatively at the same time to overcome the effect of wireless channel fading. In [84], a solution to the problem of discontinuity in network connections is presented through the use of cooperative transmission. The cooperative transmission protocol direct nodes to retransmit the received information at the same time and using the same channel information, thus, increasing the overall Signal to Noise Ratio (SNR). The information propagates through the network until it reaches a destination. The redundancy of retransmitting the same received signal by the nodes can be set at the protocol level. A novel three-stage relaying cooperative communication scheme was proposed in [85]. The scheme was compared to the conventional cooperative communication for on-body wireless transmission and measurements of Packet Error Rate (PER) and Energy Efficiency (EE) were recorded. The simulation results showed that the use of three cooperative nodes provides a higher network reliability with lower PER compared to a single node and two cooperative nodes. While the cooperative transmission concept results in larger transmission coverage and provides a solution for reaching out to discounted nodes from the network, it will also increase the overall energy consumption by introducing unnecessary data transmission at all time slots (particularly when a traditional node single-hope transmission could have delivered the message). A hybrid protocol that uses cooperative node transmissions only when necessary and a traditional multi-hop transmission used otherwise, provides an optimized solution to the problem at the cost of implementation difficulty.

2.5.4 Synchronization methods

Time synchronization in body sensor networks is important to ensure optimal network performance. Sensor nodes are normally programmed to function in a time-based manner, for instance, measuring certain parameters, detecting an abnormality in behavior and transferring data. Nodes are expected to operate flawlessly while communicating with other nodes within the network. Throughout the communication process, the nodes handle and share information requests across the network. Since these sensors run on their on local clocks, a unified time is required where all nodes need to be synchronized to share time accuracy through the network. These network synchronization methods rely on the message exchange between nodes [86]. Furthermore, the ability of the network to transmit data in real-time, without error, is a vital aspect in most of the BSN applications. In such applications, the time synchronization between sensor nodes must work seamlessly. However, some of the sensors in the system have limited resources of energy, storage, computation and bandwidth along with the high density of nodes which might lead to unstable synchronization for the network. Different synchronization methods for real-time feedback have being used. There are four main packet delay factors in any wireless synchronization system; sending time, accessing time, propagating time, and receiving time [22].

- Sending time is the time required to construct a message by the sender in order to transmit it to the network.
- Accessing time is the MAC layer delay time when accessing the network. This include the queue for transmission waiting time in a TDMA protocol.
- Propagation time is the time necessary for the bits to by physically transmitted

on the medium.

- Receive time is the message processing time of the receiver node.

In [87], a Reference Broadcast Synchronization (RBS) scheme for sensor networks was reported. Unlike the traditional synchronization methods that use two-way exchange messages between sender and receiver nodes, this approach uses a “third party” that broadcast a reference beacon signal for all receivers to allow end to end receiver synchronization. In the RBS scheme, a third-party node sends a reference beacon to their near neighbors. The timestamp is not included in the reference beacon. However, the time of arrival is included via receiving nodes as a reference point for comparing clocks in order to calculate their relative time offsets. Therefore, the time will be based on when the node receives the reference beacon, and all receivers will exchange their timing information and be able to calculate the offset. This results in better precision than the traditional synchronization algorithms since uncertainty from sending time are removed. The receive time will be the only source of error that can affect the RBS while neglecting the propagation error. While the single broadcast will propagate to all receivers at essentially the same time and uncertainty of the sender can be eliminated, this is only true when the node coverage areas are relatively small compared with scattered density of the network. A timing-sync protocol for sensor networks (TPSN) was proposed [88]. The protocol follows a hierarchical topology in the network, where every node is assigned to a level and only the root node assigned to level zero. Nodes at a certain level are synchronized with the parent nodes at the higher level by exchanging time-stamp messages via two-way handshake approach. TPSN is a tree-based scheme designed for multi-hop networks to ensure the timing information can accurately propagate through the network. The root node sends the time sync packet in order to set the time synchronization procedure to achieve network wide synchronization. The results claim that the TPSN multi-hop protocol works to minimize the uncertainty of the sender, and achieves two times better accuracy than RBS.

A study was made to overcome the issue of time synchronization in a distributed body sensor network [89]. A unique Flooding Time Synchronization Protocol (FTSP) was proposed. The protocol was designed to provide precise timing between the wireless sensor platforms. The protocol stamps the MAC layer and estimates the error in clock shifts by comparing the flooding of synchronization messages and analyzes the time differences and uncertainties to dynamically update the network topology. The system was implemented and evaluated in a multi-tier network of 60 nodes. The average hope synchronization error was calculated within a one-microsecond interval. A new technique was described in [90] for clock synchronization in wireless sensor network called consensus clock synchronization. This technique provides an internal synchronization to a virtual consensus clock. The new technique aims for reducing the errors of the clock between different geographical locations of nodes close to each other and achieves long and effective synchronization time. Simulations results based on a mesh network have been used to illustrate the synchronization protocol effectiveness.

A novel Heartbeat driven Medium Access Control (H-MAC) protocol for body sensor networks was presented in [91]. The heartbeat rhythm information was used rather than using the periodic synchronization beacons in order to accomplish time synchronization. Bio-signals of the heartbeat rhythm were monitored and recorded using the sensor nodes placed on the human body to detect waveforms peaks. In the waveform, these peak points were used as a natural time source across the body network to achieve time synchronization without turning on the radio. The results showed that H-MAC can extend the network lifetime radically and reduce the overall network communication overhead compared to other synchronization technique. Moreover, an active recovery scheme for network resynchronization was developed and tested using a discrete event simulator OMNet++ [92] with real ECG data.

2.5.5 Frequency, propagation, and interference

The node coverage gives an indication of the size of the physical area at which the node can communicate within a wireless network, however, this is not always the case. Some nodes can extend their coverage beyond their radio transmission range through the use of multi-hop techniques. The multi-hop transmission protocols require utilizing routing tables and neighbor lists for a given transmission range. This will increase the overall cost, overhead transmissions of the network and reduces its lifetime. In the propagation channel, there are some parameters that affect the quality of service provided. Signal obstruction, multipath effects and the presence of dielectrics can affect the quality of the signal received. Moreover, the presence of the human body as an obstacle affects the propagation of the radio waves in a body sensor network [93]. Normally in BSNs the transmission range is shorter than other networks. In radio communications, a considerable amount of the transmitted power is reflected and absorbed by the human body [2]. In addition, human body movements interfere with the line of sight (LOS) propagation path between the transmitting and receiving nodes resulting in multipath fading, shadowing and scattering effects which affect the network reliability.

There is a wide range of wireless channels available to body sensor applications. The main frequency bands that are defined in the United States by Federal Communications Commission (FCC) are the Medical Implant Communication Service (MICS) band; the Industrial, Scientific and Medical (ISM) band; Wireless Medical Telemetry (WMTS) band and Ultra-Wide Band (UWB) [94]. The MICS allows a bi-directional radio communication between implanted nodes and medical station operates. The maximum allocated transmission power is limited to an Equivalent Isotropically Radiated Power (EIRP) of 25 microwatts to avoid interference with other users of the same band. MICS operates in the frequency range of 401MHz-402MHz and 405MHz-406MHz [95]. ISM bands are radio frequency bands that are reserved for applications in the industrial, scientific and medical purposes; a popular one operates in the 2.4MHz to 2.5MHz. Many applications use these bands as the communication frequency, such as cordless phones, Bluetooth devices, wireless computer networks, microwave ovens, and life support equipment. Thus, the use of these bands is limited to low power and short range communications [96]. WMTS operates in the frequency bands 608MHz-614MHz, 1395MHz-1400MHz and 1427MHz-1432MHz, and was introduced to overcome the interference problem of digital television. In UWB technology, the information is spread over a large transmission bandwidth, more than 500MHz in the 3.1GHz–10.6GHz band. Applications of UWB involve the locating and tracking of the wireless sensor with precision [97].

Due to the popularity of wireless body sensor networks, many users and applications might utilize this technology over a small area, such as a sports center. The wireless transmission of multiple BSN users may cause inter-user interference, and result in poor network performance in terms of efficiency, latency, and energy consumption. As a consequence, the service quality is affected [98, 99]. For example, most BSN applications operate in the 2.4GHz frequency band, which might result in problems of coexistence. The inter-user interference is caused by simultaneous transmissions from two or more body sensor network users in close range. Many techniques and approaches have been proposed to overcome this issue. An example study shows adopting carrier sense multiple access (CSMA) with collision avoidance (CA) and the Packet Error Rate (PER) as an evaluation metric for inter-user interference [100].

2.5.6 Network traffic and stability

Different BSNs applications require different sensor types and numbers attached to the human body, most applications use any number in the range of 2 - 20 nodes. Network size and traffic control will be different for each application type. In sport monitoring applications, the speed of communication and quality of data transmission are important factors in the design of such networks. The quality of service provided defines the ability of real-time monitoring of subject vital information and the maximum network load that can be handled without latency. The data rate requirement is different for each application and it can vary from 10kbps to 10Mbps. While sport and bio-medical applications normally require lower data rate specifications, users are usually loaded with different sensors. Therefore, the overall network traffic will be a result of these factors. Different techniques have been introduced to maintain the required body sensor network operation with large traffic. These include error detection and corrections, dynamic and spatial routing protocols and code multiplexing of the transmitted signals [3].

BSNs are expected to operate for a long time with efficient power management and would not be affected by the body movements. However, the presence of human body between the radio communication channel of the displaced sensor nodes introduce multipath and scattering effects, thus increase, the complexity of the radio propagation modeling for body-worn channels. To ensure network stability, pre-test and simulation are required for the on and off-body communication channels. A BSN consisting of small sensor nodes with highly integrated electronics operated on battery power was tested to evaluate the network stability in real environment conditions [11]. These nodes share the same timing information which allows synchronizing measurement across the network. The Received Signal Strength Indicator (RSSI) was monitored and recorded at different rates for all nodes and for both stationary and mobile body movement. The collected data was put into Nakagami, Rician, and Rayleigh Fading Models to characterize the channels performance. It was shown that Nakagami fading provides an optimum fit for stationary networks while the Rice distribution was the optimum fit for mobile channels. The test showed that, for different body location and body action, there is a different resulted RSSI value. The power distribution frequency function was used to identify correlation for the channel scattering effect for all different scenarios and fading models used. Overall the result showed that taking all network design requirements and specifications into account, special consideration should be given to sensor locations which will play a significant role in setting the limits of the channel paths.

2.6 Applications of the Body Sensor Networks

2.6.1 BSN in sports and bio-medical applications

The enormous availability of sensory devices encouraged the use of BSN in many possible applications, such as lifestyle, assisted living, athlete monitoring and sports analysis, post-operative care, entertainment functionalities, interlinking of components in home entertainment products, navigation support, a museum or city guide, and infant monitoring [16]. A complete survey can be found in [3, 101]. Alarm-Net is a wireless sensor heterogeneous network for assisted-living and residential monitoring that was specifically designed by the Wireless Sensor Network Research Group at the University of Virginia in 2006 [102]. Alarm-net consists of body sensor networks and environmental sensor networks. A three-layered scheme consisting of *Body Networks, an Emplaced Sensor Network and AlarmGate* were proposed to assisted-living and home environment. The Body networks are body-mounted wireless sensors usually worn by patients, elderly or residents to sense individual physiological data. The Emplaced Sensor Network are environmental sensors, such as temperature, dust, motion, light are deployed in the living space to sense the environmental conditions and provide information about the patient's activity, movements, and location. The stationary sensor nodes form a multi-hop network and forward the data to the nearest AlarmGate which is an Internet protocol (IP)-based network serving as gateways between the wireless sensor and IP networks. The AlarmGate is also connected to a back-end server to allow user interaction with the system. The use of BSN in such applications can help the user by providing feedback with relevant information that is necessary to determine the next step in that specific application field. In the sporting area, it’s not possible to take many different readings from an athlete without having them on a treadmill in a laboratory. The ability to measure various parameters during real life competition, a race, or to monitor athletes closely would give coaches a more accurate picture of athlete strengths and weaknesses.

In addition, the ongoing development of body wireless sensors network has improved the quality of health care services, increasing their accuracy and reliability [103]. The importance of wearable sensor networks had attracted a lot of researchers and industry providers to invest in this kind of technology, mainly because of the promising features of these networks [80]. BSNs can monitor patients continuously and whether they are at home, in a hospital or elsewhere. Patients will no longer need to be connected to large machines in order to be monitored. One application is capturing vital parameters when patients can be monitored in their homes or when they are mobile. A BSN health monitoring system introduced by [4] consists of wearable motion and heart rate sensors and a home gateway server to manage data processing, gathering and forwarding to the medical server. This technology is used in biometric sensors to serve people who undergoing cardiac rehabilitation or having difficulties in leaving their houses (such as the elderly). The deployment of wireless body sensor networks in healthcare applications has huge potential to provide patients with the best and the most comfortable possible care, the ability to provide emergency response to big disasters, the chance to enhance the elderly lives and study the human behavior for early detection of clinical deterioration. Not all of these applications require all sensors mounted on the human body, some would involve a uniform distribution of the sensor nodes in key service buildings (such as hospitals) along with the human body for emergency responses [16].

In 2006, Massachusetts Institute of Technology (MIT) introduced a wearable blood pressure sensor using a MEMS accelerometer and a hydrostatic-based oscillometric method to allow accurate measurement during hand free movement that can provide continuous 24-hour monitoring and help diagnose hypertension, heart disease problems [104]. A new approach to provide real-time feedback for the cyclist was proposed by [5]. The system involves three wireless sensors placed on the cyclist's leg, thigh and foot to measure the angular movement during biking sport using computational algorithms on the recorded data from accelerometer sensors. Another application uses the body sensor network to identify the quality of a golf swing with respect to the angle of wrist rotation. The proposed system uses a quantitative model and signal processing methods to provide feedback on the quality of movement for training purposes [105]. CodeBlue [106] is a wireless communication infrastructure that was developed at Harvard sensor networks Lab in 2004 to handle a mass casualty scenario and provide robust communications with a high degree of mobility and minimal packet loss. The designed modules operate on publish/subscribe basis and published information can be filtered to transfer only significant changes in the status of the patients to avoid traffic congestion and network overhead. The sensor nodes transmit a stream of vital signals, names, locations and identities to the subscribed local host (PC or PDA) which is accessed by the medical staff acquiring information for a particular patient. The use of ad hoc networking allows the flow of information from the sensor nodes to its destination and the placement of additional fixed sensor nodes in hospital rooms increase the network coverage. The loss of any sensor node is dispensable and the network is considered as self-organizing by allowing re-calculation of the best route to deliver information.

MEDiSN or Medical Emergency Detection in Sensor Networks [107] is a health care system developed at Johns Hopkins University, was specially designed for patients' monitoring in hospitals and during disaster events. It comprises a two–tier network that consists of multiple physiological monitors (PMs) and relay points (RPs). The PMs are mobile, battery-powered motes, equipped with medical sensors for collecting patients' physiological health information (e.g., blood oxygenation, pulse rate, electrocardiogram signals, etc.). These PMs can temporarily store sensed data and transmit it to the relay points (RPs). The stationary RPs are self-organized into a bidirectional routing tree that forwards sensed data to the gateways and vice versa. MEDiSN was designed to provide reliable communication and to meet the application's quality of service (QoS) by using a collection tree routing protocol (CTP) to forward the measurements from RPs to the gateway. It is connected with a back-end database that constantly stores medical data and presents them to authenticated GUI clients.

Ubiquitous Monitoring Environment for Wearable and Implantable Sensors (UbiMon) [108] was designed to address the issues related to using wearable or implemented sensors (e.g., electrocardiography (ECG), saturation of peripheral oxygen (SpO2), and blood oxygen) for distributed mobile monitoring. The project objective was to allow patient monitoring using a local processing unit which can be held by the patient in the form of any portable device (such as PDA or mobile phones). This portable device is programmed to collect data from the sensors and to detect any abnormalities in the patient health to provide immediate assistance by acting as routers and transfer information to the central station using either short (Bluetooth / WIFI) or long range communications (3G / GPRS). MobiCare [109] operates in a timely manner to provide a wide-area mobile patient monitoring system that allows the medical staff to remotely analyze physiological data. The time requirements to be considered in patient monitoring vary significantly from few minutes to hours depending on the patient status. Similar to UbiMon, the MobiCare system uses a BSN that has wearable sensors (e.g., ECG, SpO2, and blood oxygen). The medical sensors collect the patient's data and broadcast them to the MobiCare client. The MobiCare client aggregates the body data and sends them using a cellular link to the MobiCare server.

2.6.2 Monitoring kinematic changes

Many different analytical techniques were used to investigate the identification of human motion and especially the analysis of running as an essential part in almost all kind of sports. It has been demonstrated that features regarding human motion can be extracted from the changes in the path loss (RSSI) over time, at least under controlled conditions [110]. In general, the motion of the human body with respect to walking or running is referred to as the gait cycle. The full gait cycle refers to the period of time from where the foot first contacts the ground until the same foot first contacts the ground again. In each gait cycle, there are two phases: the stance phase and the swing phase. The stance phase refers to any time in which the foot is in contact with the ground, whereas the swing phase is any time when the foot is not in contact with the ground (i.e. time between each consecutive foot contacts with the ground) [111]. During the natural human running movement the swing time is identical for opposite legs and arms [112]. It was found that, the variation in the swing time between human arms and legs of different runners is very small at their top speed (around 400ms) [112]. Gait cycle shortens with respect to speed, however, as speed increases the time spent in swing phase increases and stance time decreases. In walking the time of stance phase is longer than that in swing phase (more than 50% of the gait cycle) and there is a period when both feet are on the ground. In running there are no such periods, on the contrary there is times where both feet are airborne during the gait cycle. For elite athletes, the time spent in the swing phase is much bigger than that spent in the stance phase.

Gait analysis requires the capture of body movement actions such as walking or running. The most common methods of video recording and subsequent image processing require complex analysis algorithms. Studies such as the analysis of gait patterns extracted from video recording [113] and identifying the parameters to achieve faster sprinting speed [112] were made on statistical calculations of the time spent in the swing and stance phases. However, these systems are expensive, only allow measurements in a restricted volume, and the markers are easily obscured from vision resulting in incomplete data. The new concept for capturing human movements utilize body sensor networks with movement sensors. More recently, body-mounted sensors have been used to obtain kinematic values [114]. A wearable sensor network for monitoring body kinematics was used in [115]. The nodes use three-axis accelerometers and magnetometers connected to a microcontroller embedded and the network can distinguish between transition and rotation movements. The sensors can also measure heart rate and temperature. The information is transferred to a base station using a 2.4GHz transceiver. However, no real-time interaction with this system was reported.

2.6.3 Physical activity recognition

Human activity monitoring provides the professionals with a data set for analyzing player performance, identify injuries and mapping the human gesture. BSN have been widely used for such applications because of the monitoring accuracy of the required parameters in a real-life environment and ability to provide real-time feedback, among other features. In [116], the kinematic features of 21 runners were monitored through wearable sensors to identify fatigue in exhausting real life exercise. During the experiments, inertial sensors were used to capture the participants' running movements in order to evaluate and extract variables that are related to fatigue. The study investigates the consistency of many kinematic changes including the foot contact with the ground, step frequency, arm movements, heel lift, body rotation and swing time. This type of application requires high timing precision in mapping the information to give a thorough picture with regards to all of the recorded data points. Body-worn sensors can also be used for individual identification and entity authentication in a secured network. In [50], a novel study was made on the possibility of identifying entities within a selected body network through physiological signals. The biomedical sensors were placed on 12 subjects to capture the inter-pulse heartbeat of each individual (the time between beat to beat). This unique time difference was used as a proof of identity to grant access to the network. While the proposed scheme suggests the subjects biometric signal as the generated security key, the key code predictability increases when the subjects are not expending energy and inter-pulse heartbeat key remains the same.

Furthermore, sensor nodes can be used to identify human activities, recognize gestures and provide a real-time monitoring environment in a home or work body networks. In [8], a body sensor network of seven wireless sensors for physical activity recognition was proposed and tested. The system employs an accelerometer sensor to record the body movements and flexion angles of the participated individuals. The monitored acceleration measurements help in identify repetition in body movement and identify human gestures. The body network provided a flawless identification accuracy for low-level movements such as sitting, standing and walking, and high identification accuracy for more complex activities. The speed of event detection for activity monitoring networks varies depending on the simplicity of the recognition algorithm, while the network latency will be a limiting factor.

2.6.4 Positioning and location detection

Tracking a player's movement and the constant monitoring of their location is another important application of body sensor networks, where the tracking algorithms provide an estimation of the subject position in a predefined space area. In [93], a 2.4GHz wireless body sensor network was proposed for tracking and positioning of players in an indoor environment. The study investigates the instability of the received power with respect to the distance of the player from a fixed spectrum analyzer while a continuous wave was transmitted from a wireless node on the player waist. A free space and two ray models were used to estimate the signal reflections and validate the technique. While the simulation and measurement showed a good result for distances of 10 meters between the transmitting waist beacon and the receiving spectrum analyzer, this location information was calculated in the lab at a different time and was not provided to the coach in real-time. A more accurate wireless network positioning system can be introduced through deploying more advanced signal processing techniques. Alternatively, multiple nodes and simple measurements of changes in the path loss on all links can be used to reconstruct, in a tomography-like process, the locations of things within a monitored zone, including both "active" things (that have RF emitters) and "passive" things (without RF emitters, only detected via shadowing). In [64], more complicated approaches such as the Time of Arrival (ToA), Frequency of Arrival (FoA), and Angle of Arrival (AoA) were deployed and compared to find an optimal distance estimation. Unlike the ToA method, which requires precise time synchronization between all nodes to estimate their positions based on the travel time from sender to receiver, the AoA approach does not require any synchronization and uses a single or array of directional antennas to detect nodes location based on the angles of at least two or more nodes. These two techniques become unreliable in dynamic networks; instead, the FoA method provides more accurate outcome through measuring the variation in the frequency caused by location changes of nodes (such as that, when the relative speed between any two nodes is zero then ToA or AoA can be used to estimate location). The study showed that for both static and dynamic networks, a hybrid system that employs all of three techniques combined results in lower location estimation error compared to the results of each technique operated independently. However, such a system requires a large computational power in terms of hardware and software design and can consume a large amount of the network resources and increase the communication overhead. Further optimization is required to reach an optimal goal and understand the balance between available network resources and the required location accuracy for a specific application.

2.6.5 Real-time feedback

BSNs in sport can provide a real-time feedback for athletes which leads to a better understanding player performance and can help coaches to effectively measure training sessions. The provision of accurate information can prevent injuries and support the rehabilitation process through monitoring performance levels over time [117, 118]. The process of identifying key elements to provide a detailed breakdown of a training or competition session is known as activity classification and can be used to better understand the individual demands of the sport. As engineering is a continuously evolving industry, every year new possibilities in activity monitoring and classifications emerge. These technologies can be applied to better understand the physical demands on the athletes. CardioNet is one of the leading providers of Mobile cardiac outpatient telemetry services. They provide real-time feedback solutions for monitoring health care of outpatients. Their focus is on diagnosis and monitoring of cardiac arrhythmias with the produced name Mobile Cardiac Outpatient Telemetry (MCOT) [119-121]. This sensor sends electrocardiogram (ECG) information of each heartbeat to a small portable monitor or Personal Digital Assistant (PDA) for every heartbeat. When the monitor receives abnormal signals, it automatically sends the ECG capture information to the cardio-Net monitoring center [122].

Live and continuous body monitoring has been demonstrated under normal physiological activities [123]. The BSN monitors ECG signals and body temperature and operates with low power requirements so that it can be used in a home or work environment. A real-time database was used in this multi-sensor environment to establish trend analysis and data mining where sensed information can be viewed through the GUI (Graphical Unit Interface). This kind of network can be developed to provide additional readings and measurements for human movements. Where information saved in the database can be accessed at any time. A combination of wireless technologies can be used to increase the reliability of information delivery in real-time monitoring networks. In [70], a body sensor network with a flexible design was used to transmit information using ZigBee and Global System for Mobile communication (GSM) to provide a real-time feedback. The ZigBee protocol is used to transfer the information while the subject located in an indoor vicinity and a GSM system was used when the subject is out of range. Such networks increase the range of free movement and ensure real-time feedback over a large monitoring area.

2.7 Conclusions

Studies and techniques used to optimize body-worn network design for sensor monitoring application were discussed. The main focus was to summarize and highlight the best methods used to support a real-time feedback and allow free movements range during normal human actions. Furthermore, BSN parameters were carefully investigated. This includes network security and the protection of information using ciphering algorithms, the Quality of Service (QoS) needed for network design to allow emergency responses and ensure ideal communication between on-body and off-body wireless channels through the best possible routing path. And code multiplexing techniques have been used to ensure network stability and traffic flow.

In conclusion, body sensor networks are a special type of wireless sensor network and have special design requirements. Different BSN applications have different network design requirements. Real-time monitoring body networks require careful consideration for the above-mentioned parameters to avoid issues associated with information losses, network failure and or delay in transferring the data. In this thesis, the wireless communication protocols are developed to ensure highest communication reliability for dynamic body sensor network and real-time applications involve the delivery of important physical parameters, such as the swing speed, steps count and symmetry and positioning. This increases the operational sensor life before recharging, or can lead to an overall reduction in sensor size and weight if the battery size is reduced.

CHAPTER 3

WIRELESS SENSOR NODE DESIGN AND MEASUREMENT

3.1 Introduction

This chapter presents the hardware design and modifications of the wireless sensor that was adopted and used throughout this research. The details of the electronic components used in the sensor design, the functionality of each unit and their power consumption are given. It also explains in particular the distinctive accelerometer settings and the data resolution used in this research, the importance of the monitored acceleration signals and the results derived from the accelerometer. This chapter shows the experiments conducted on the wireless connection for static and dynamic measurements and their corresponding results, the hardware and the software tool used to measure the received signal of interest (RSSI) for each connection. The analytic techniques and algorithms that were designed to characterize the wireless link performance for all the different tests are also reported.

3.2 Wireless sensor design

A body network design of five wireless transceiver sensor nodes was employed; all of the sensor nodes were developed at Griffith university [124]. The hardware structure of the nodes was modified to include the wireless transceiver ability.

Each node consisted of a MCU (ATmega324P Microcontroller Unit), wireless transceiver chip module (NRF24L01+) with a built-in PCB meander-line monopole antenna, accelerometer sensor (MMA7260QT) with acceleration sensitivity of 2g, memory (AT45DB161D) and a power source (as shown in Figure 3-1) in the form of 3.6V battery with a maximum capacity of 240mA. The nodes were programmed to work as transceivers to allow bidirectional communications and were able to store transmitted data and/or received data from other nodes. The codes that drive the node operations were implemented using C language and Atmel Studio 6.1. Each node has a Universal

Asynchronous Receiver/Transmitter (UART) interface (CP2102) which allows wired data transfers to a local computer for real-time monitoring or further processing during or after the completion of the sport activity. The PCB design and the pin connections in-between the MCU, RF circuit and other units is shown in Appendix A.

This sensor is small, lightweight, portable and low cost. The MCU has 32 pins which can be used as input/output (I/O) headers. The core includes 8 channels for analogue input coming from other sensors and a 32Kbytes Flash memory for programming with 1Kbyte EEPROM and SPI bus for communicating with peripheral devices. The processor can operate at variable clock rates from kHz up to 20MHz with 10MIPS. Some of these I/O ports and the SPI were used to connect this sensor to the wireless transceiver circuit, as shown in Figure 3-2.

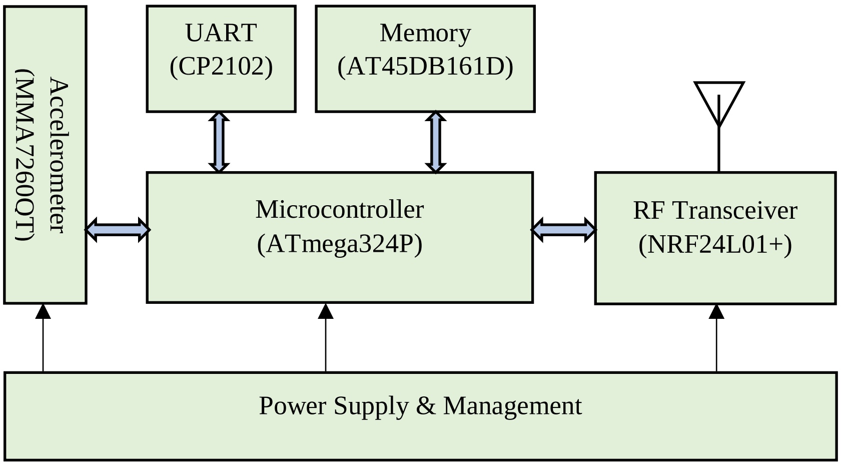


Figure 3-1 Hardware structure of the wireless sensor node.

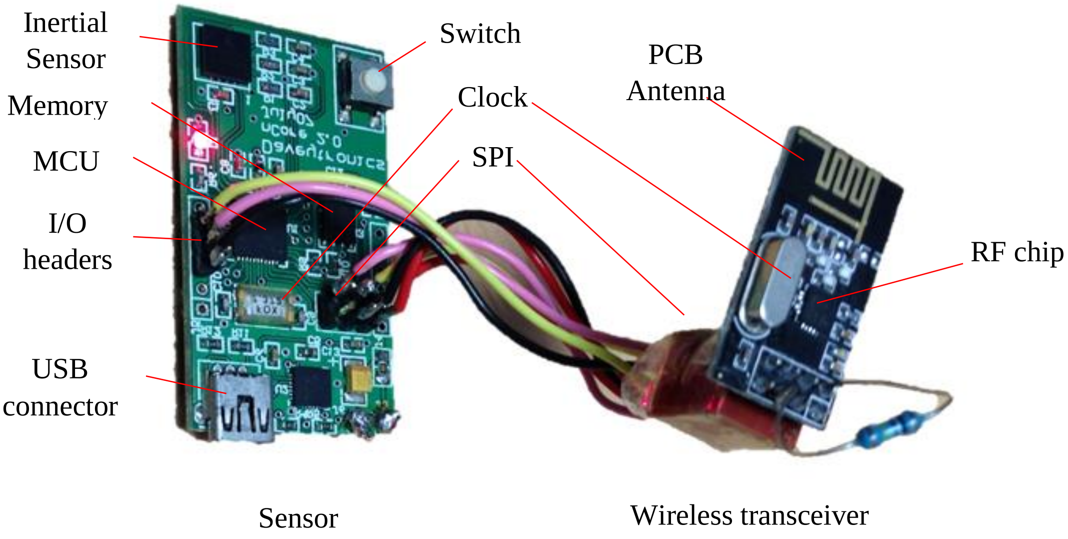


Figure 3-2 Layout of the wireless sensor components.

The radio transmissions side of body sensor network design must ensure reliability and power effectiveness. Many BSNs protocols have been discussed in Chapter 2, Section 2.5. In this research, the wireless node was chosen to operate at the ISM band with centre frequency of 2.45GHz, because of the low power consumption for these transceivers and the ability to program them with modified communication protocols suitable for the application. The commercial availability of the NRF24L01+ transceivers that can operate at those frequencies and their low cost is another advantage. Table 3-1 shows the power consumption for the designed wireless node. In the active mode, the total current drawn is 56mA and using a 3.6 V small Li-poly rechargeable battery with capacity of 240mA/h. The battery capacity gives a node lifetime of around three hours, which is sufficient to monitor a complete training session, however, bigger battery capacities can increase the network life time depending on the application requirements.

Table 3-1 Wireless node component and their power consumption.

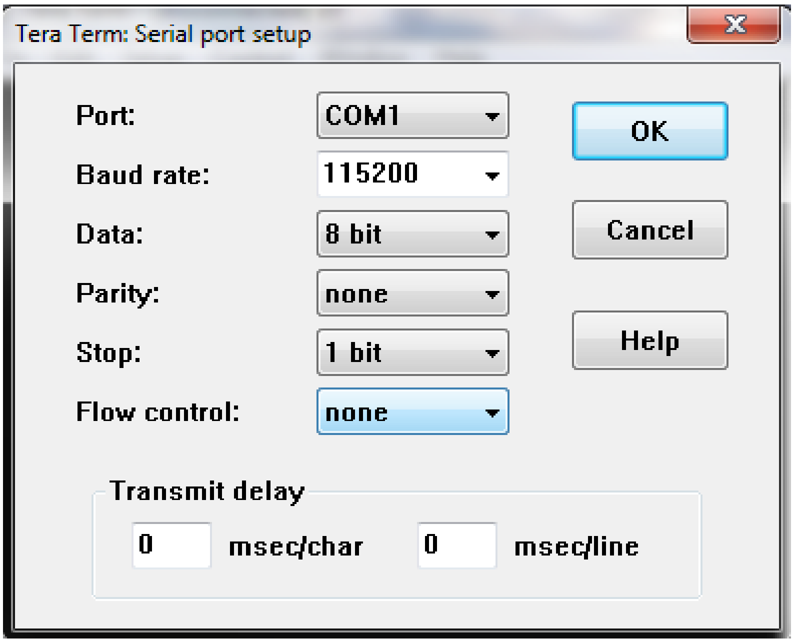
Component		**Power Consumption**	
Name	**Model**	**Active (mA)**	**Idle (µA)**
MCU	Atmega 324P	9	5000
UART BRIDGE	CP2102	26	100
Accelerometer	MMA7260QT	0.5	3
Memory	AT45DB161D	7	25
Transceiver	nRF24L01+	13.5	26
Total Power Consumption		**56**	**5154**

3.3 Data collection

The data are collected and stored in the memory of the accelerometer sensor and can be downloaded to a computer using the USB interface. Usually, the data set is arranged in three columns, which represent the acceleration values of x, y and z axes. Additional columns with information of the recorded sample number and time during a training session can be added to the data set.

All of the data used in this project were gathered according to the ethics guidelines stated in Chapter 1. Several steps were required to extract the data from the recorded format to a format ready for signal analysis. Most of the recorded formats are based on a simple text format and can either be imported directly into MatLab or extracted into comma separated value files (CSV) and imported into MatLab.

The sensor node communicates via a virtual COM port using any terminal software (e.g. Tera Tram, hyper terminal, Windows). This software can be used to configure the node serial port and download recorded sessions, Figure 3-3 shows the terminal settings for establishing a connection via serial interface. The pre-programed sensor provides a menu in USB connection mode, to allow the user to download a recorded session by choosing D from the menu and to erase all recorded sessions by choosing E from the menu. The recorded data will then appear in the terminal software window and can be saved as a text file (see Figure 3-4).



Tera Term: Serial port setup

Port: COM1

Baud rate: 115200

Data: 8 bit

Parity: none

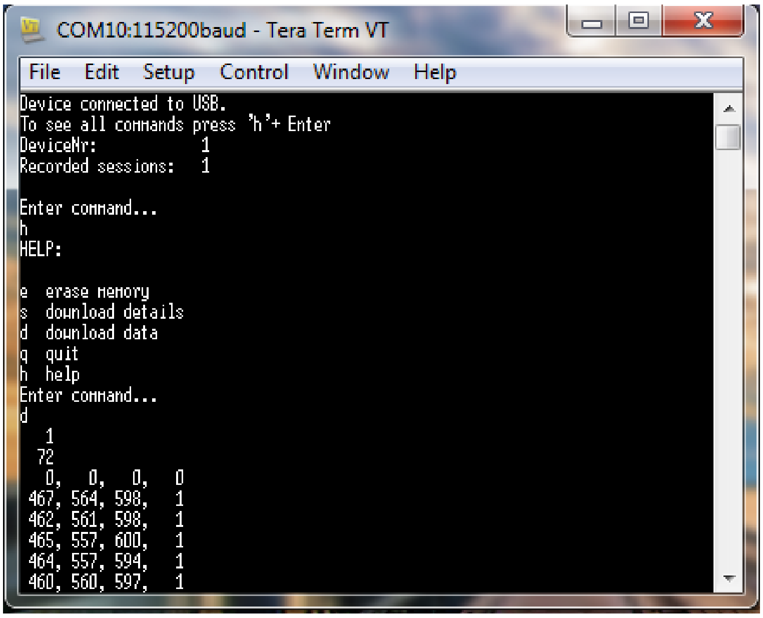
Stop: 1 bit

Flow control: none

Transmit delay

0 msec/char 0 msec/line

Figure 3-3 Terminal settings for establishing a connection via serial interface.


COM10:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
Device connected to USB.
To see all commands press 'h'+ Enter
DeviceNr:          1
Recorded sessions:  1

Enter command...
h
HELP:

e  erase memory
s  download details
d  download data
q  quit
h  help
Enter command...
d
   1
  72
   0,   0,   0,   0
 467, 564, 598,   1
 462, 561, 598,   1
 465, 557, 600,   1
 464, 557, 594,   1
 460, 560, 597,   1
```

Figure 3-4 Download menu of sensor using hyper terminal.

As this procedure of downloading recorded data is very time consuming and the text file needs to be imported into further processing software like MatLab, a more comfortable tool was developed. This new download tool developed by the author is described in Chapter 5, Section 5.6. There are many commercial and freeware software packages available for analysis of data, but more powerful algebraic software suites were preferred. Table 3-2 shows the software suites that were used in this research for programming, data processing and analysis.

Table 3-2 Overview of available software suites.

Software suite	**Version**	**License**	**URL**	**Uses**
MatLab	R2013a	Commercial	http://au.mathworks.com/	Data Modelling
Altium Designer	AD15	Commercial	http://www.altium.com/	PCB Design
Atmel Studio	6.1	Commercial	http://www.atmel.com/	Code Developing/Debugging
GeoGebra	3	Freeware	https://www.geogebra.org/	Geometry Calculator
Code Blocks	16	Freeware	http://www.codeblocks.org/	Code Editing
Dock Light	V2	Freeware	http://docklight.de/	Serial Communication Analysis
Notepad++	8.0.4	Freeware	https://notepad-plus-plus.org/	Code Editing
Tear Term	4.92	Freeware	https://ttssh2.osdn.jp/index.html.en	Terminal Emulator

3.4 Accelerometer unit settings and data resolution

The three-axis accelerometer sensor measures the analog acceleration data in three orthogonal directions. Any movement of the sensor at any given direction will produce an analog voltage reading at the sensor output which is converted by the 10 bit MCU analogue to digital convertor (ADC) to a digital form of (mV/g). The hardware limitation of MCU results in a range of digital values between (0 – 1023) which is the 10th power of 2 (1024). The accelerometers have different selectable sensitivities of (1.5g, 2g, 4g and 6g); these values represent the maximum values the accelerometer can detect. For example, when the sensitivity is set to 2g, the highest voltage derived from the accelerations that are equal to or higher than +2g are converted to a digital value of 1023, while the lowest voltage derived from accelerations equal to or lower than -2g is converted to a digital value of 0. Manual calibration is needed in order to convert these numerical values into gravitational units (g’s) with the reference to the 1g acceleration of the earth’s gravity. When the sensor is at rest, only the gravity component contributes to the acceleration. The three acceleration components on the sensor co-ordinate system can be converted to the gravitational co-ordinate system. The total acceleration is given by:

$$acc(total)^2 = \mathrm{acc(x)}^2 + \mathrm{acc(y)}^2 + \mathrm{acc}(z)^2 \qquad Eq. 3-1$$

Each accelerometer sensor had different calibration readings, since the sensitivity can be changed at any time during the operation of the device. Furthermore, the recorded acceleration values will not be exactly symmetrical for different axes as shown in Table 3-3.

Table 3-3 Maximum, central and minimum acceleration values for three axes obtained at 1, 0 and -1 (g), respectively.

g value	x-axis	y-axis	z-axis
1 g	543 mV/g	622 mV/g	522 mV/g
0 g	492 mV/g	549 mV/g	487 mV/g
-1 g	408 mV/g	525 mV/g	460 mV/g

In the calibration test of one sensor, the average of the acceleration numerical readings was recorded at stationary (idle) positions for all three axes while changing the accelerometer sensor orientation with respect to the ground. As the averaged acceleration readings are different for each axis, it is false to assume the offset resulted from Eq. 3-2 will have the same values for all axes. Any acceleration value for all three axes can be converted to the corresponding (g) values using Eq. 3-3. The resulting acceleration data in gravitational units (g's) is shown in Figure 3-5. After the conversion, the actual acceleration data does not change, however, the scale of the y-axis has now been changed from acceleration units (left side of Figure 3-5) to gravitational units (right side of Figure 3-5).

$$Offset = \ 0.5(max \ \ - min) \qquad\qquad Eq.3-2$$

$$g \ value = \frac{(acc\text{–}\,offset)}{(max - offset)} \qquad\qquad Eq.3-3$$

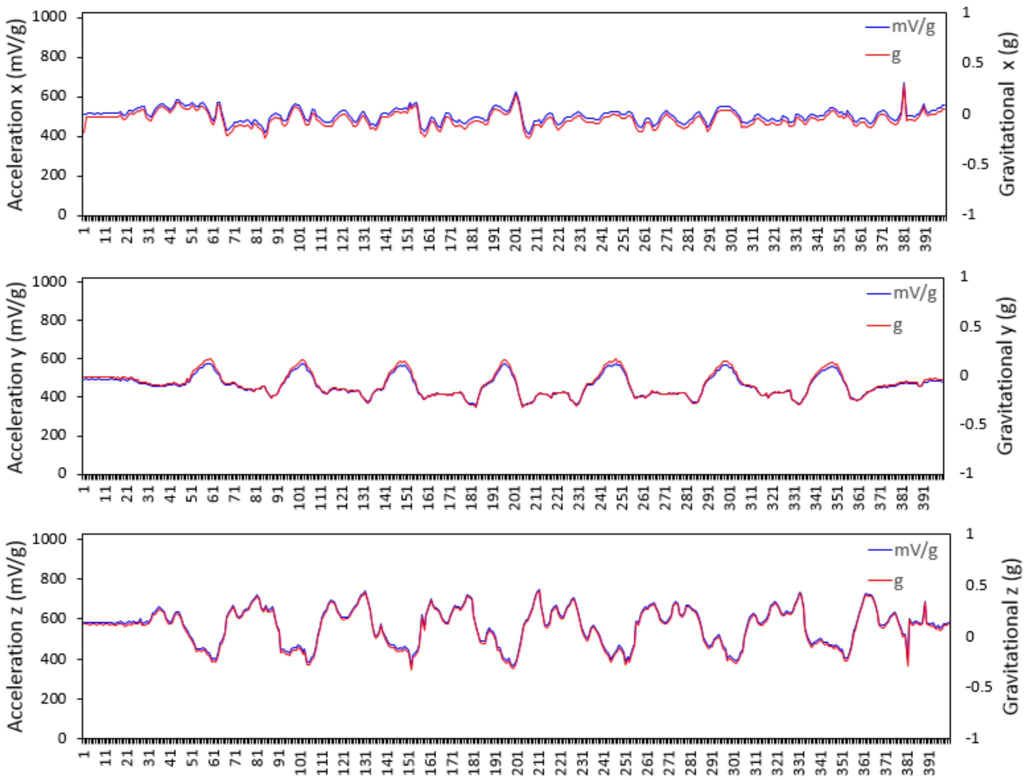


Figure 3-5 Calibration of acceleration values from (mV/g) to (g).

Where *offset* is the difference between the maximum and minimum acceleration values (recorded in mV/g) divided by two, and the *acc* is the acceleration measured in mg/v.

Additional feature of the accelerometer sensors is the estimation of the tilt angle as a function of the acceleration values. Stationary angular measurements were made by changing the tilt angle of an analog tilt meter and the accelerometer node with respect to the ground. Both devices were attached to the arm position to be moved together. In each stationary position, the acceleration values were compared with the angles obtained from the analog tilt meter. The node was configured to measure the acceleration data at a 100Hzsampling rate, and wirelessly transmit the samples off-body for real-time display. The acceleration values were recorded at 72 stationary positions from -90° to 90° with step size of 5° in both y & z directions (y & z axes are parallel to the ground at 0°).

At each stationary position, the resultant acceleration values were normalized to (g) units for all three axes and mapped against the recorded analogue angles resulted from tilting as shown in Figure 3-6. The result showed that, leaning the sensor in y direction changes the acceleration readings in the y-axis from a negative value of -1 (g) at -90°, to 0 (g) at 0°, and back to 1 (g) at 90°; same results were obtained for z-axis acceleration in z direction. This corresponds to the physical orientation of the sensor components in the y-axis or z-axis while remaining unaffected in the other axis. From these values the direction of the sensor (limb) was obtained for stationary positions. A four-quadrant circle which shows the singularity of the acceleration values is shown in Figure 3-7. For instance, if both y and z values were positive then the sensor is located in the first quadrant, similarly negative values indicate a sensor location in the third quadrant. Furthermore, the acceleration values in the x-axis change almost identically to the variation in either y or z directions and can provide a clear indication of the tilted angle. Given the digital resolution of the acceleration values and the sinusoidal dependence of the angle on acceleration, the maximum angular error was 5° providing the human body is motionless. When the limbs are moving, then the dynamic acceleration greatly increases the angular uncertainty.

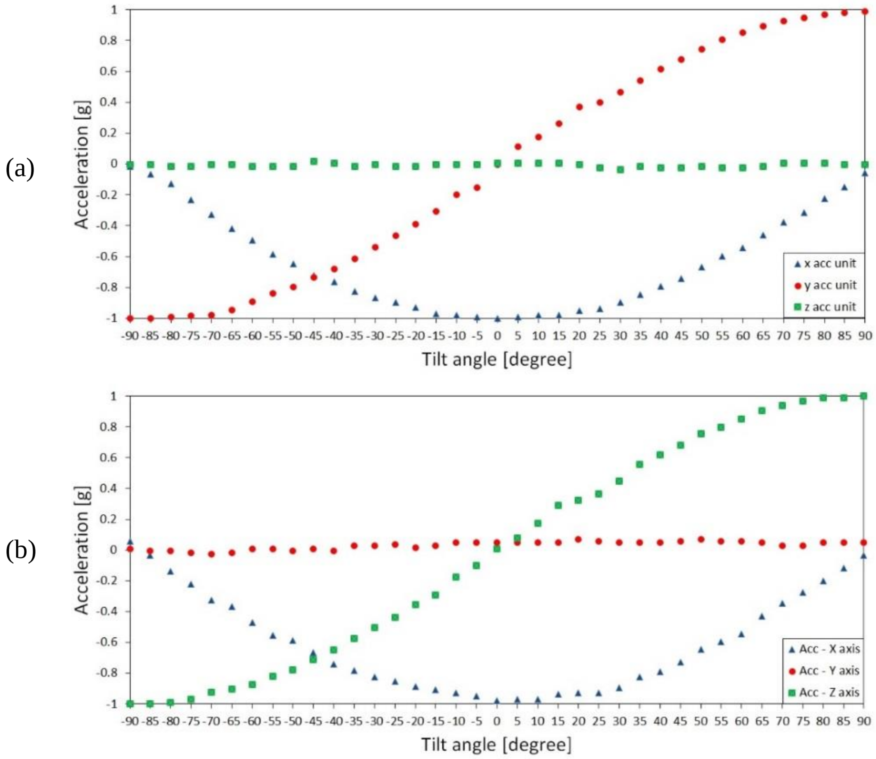


Figure 3-6 The mapped acceleration values in [g] with respect to the tilt angle sensor in (a) y direction (b) z direction.

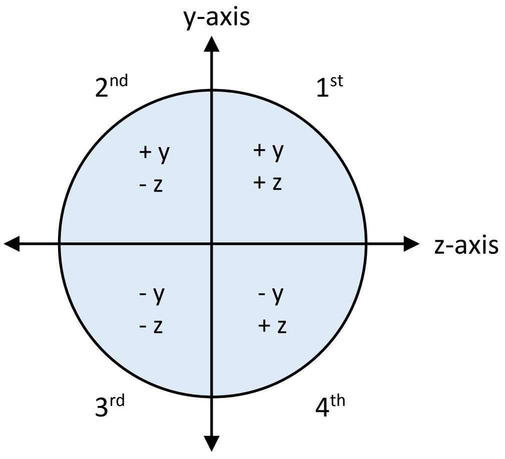


Figure 3-7 A top view of the rotation of the sensor in YZ plane.

To obtain the acceleration of a moving object with respect to the earth, the "gravity offset" needs to be subtracted from the acceleration measured. This is not required for the other acceleration directions which are perpendicular to the earth. The acceleration component $acc_i(t)$ at time t along the ith axis (i=x, y and z) is given by the equation [125]:

$$acc_i(t) = \mathrm{g}\cos(\theta(t)) + i.acc_T(\mathrm{t}) + \omega(\mathrm{t})^2\mathrm{i}.\mathrm{r}(\mathrm{t}) + \alpha(\mathrm{t}).\mathrm{r}(\mathrm{t}) \qquad Eq. 3-4$$

Where g is the gravitational acceleration, $\theta(t)$ is the time dependent angle between the axis and the vertical direction, i is the unit vector, $acc_T(t)$ is the translational acceleration, $\omega(t)$ is the angular velocity of rotation and $\alpha(t)$ is the angular acceleration, both at a distance $r(t)$ from the center of rotation.

Figure 3-5 shows the acceleration data that were recorded for the three axes while moving the sensor node along the y-axis direction with respect to the ground. As seen, a different acceleration reading was obtained at each axis depending on the movement of the body. Selecting the optimum reading for gesture or action detection directly dependent on the direction in which the sensor is moved. The measured acceleration data can either be stored in the memory or transmitted using the wireless transceiver. The test showed that, moving the sensor in the y-axis direction provides a recognizable pattern and clear readings compared to the other axes. The same can be said when moving the sensor in the x or z-axis. This test identifies the most favorable and promising orientation of the accelerometer sensor for a specific movement.

Another important feature is the acceleration data resolution or the sampling rate. The sampling rate is programmable and can be set differently for different applications. A low sample rate of (10Hz) would give 10 acceleration values per second for the human movement and therefore may not be enough to identify gestures derived from fast activities (running as an example). A larger sample rate (such as 200Hz) will give more details than required (200 sample per second), and consequently, it will consume more power from both the transmitter and receiver as well as the channel capacity and memory (as shown Figure 3-8). Therefore, 100Hz sampling rate was selected as an optimal value to fulfil the needs of this study [10].

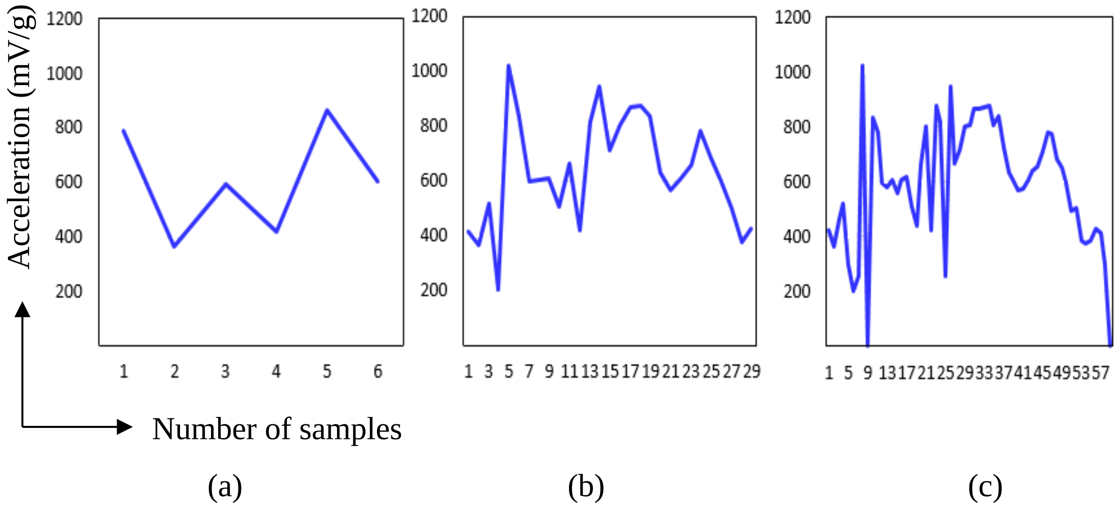


Figure 3-8 The effect of different sample rates for an acceleration signal (a) 10Hz (b) 100Hz (c) 200Hz demonstrate the effect of under-sampling.

3.5 *Wireless connection measurements*

Wireless measurements were conducted in an outdoor sport field to characterize and compare the wireless link performance when attached to a human body and in free space. In the tests, the wireless nodes were configured at same power level and frequency [0dBm and 2.4GHz], and transmission packets were saved at both the transmitter and receiver units for comparison. The free space measurements gave as result a maximum distance of 12m free of transmission errors. This range can be increased by using an RF amplifier or by using a small external antenna on the wireless sensor. Figure 3-9 compares the packets transmitted and the packets received as a function of the range; the link is not very reliable at distances greater than 12m. This range significantly changes when attaching the nodes to a human body. Figure 3-10 shows the connection reliability with respect to the received packets when the wireless nodes are attached to the human chest and back. In the body measurements test, the transmitter was attached to center chest position and the receiver was moved along the side of the chest area all the way to the center of the back, as shown in Figure 3-11. The reliability decreases significantly at distances longer than 15 cm because of the LoS obstruction.

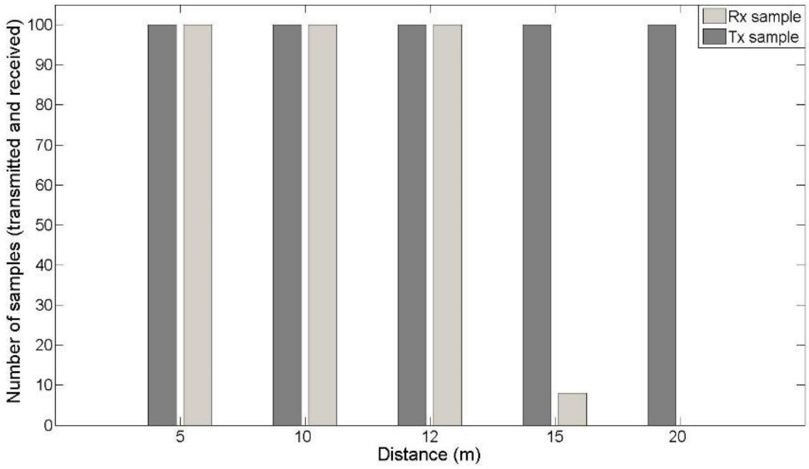


Figure 3-9 Air connectivity test showing the transmitted and received number of samples with respect to distance.

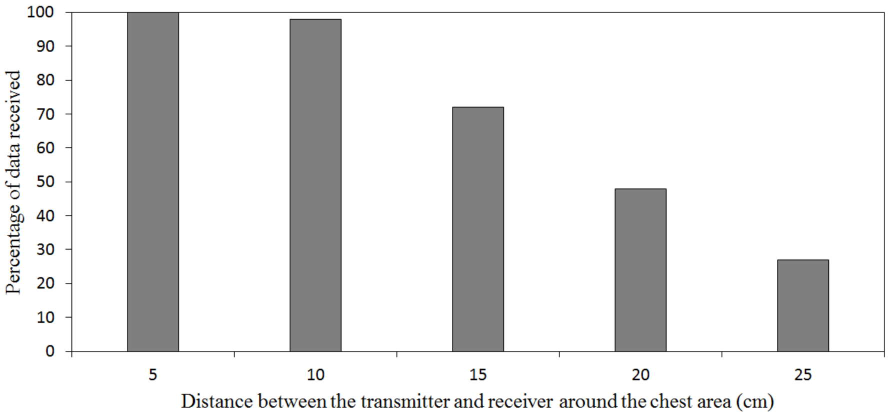


Figure 3-10 Body connectivity test showing the percentage of packets received with respect to distance. The line of sight is clear at approximately 15cm.

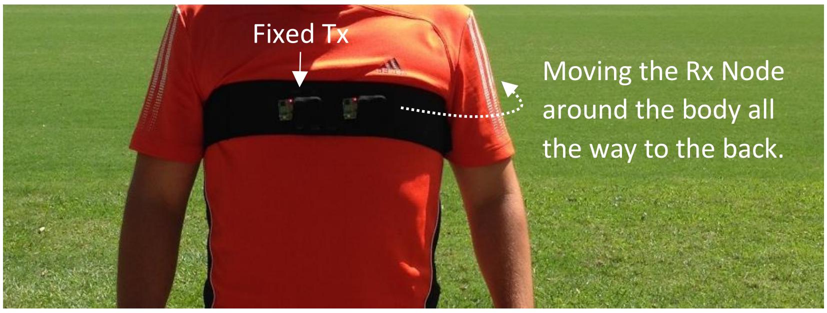


Figure 3-11 Transmitter node attached to the center of the chest position while the receiver node move along from the chest to the center of the back. Wireless measurements were taken every 5cm step.

3.6 Static RSSI measurements

In all communications systems, the received signal is affected by the transmitted power, the directivity of the transmitting and receiving antennas, the pointing angle of the two antennas, the distance between the antennas and the presence or absence of obstacles in the line of sight (LOS) path between the two nodes [25].

A linearly polarized transmitter and receiver were mounted on the body, and the signal level was measured as a function of limb position. The small, portable RF explorer receiver (WSUB1G) with an average internal noise level of -105dBm was used to monitor the received signal level from the wireless transceiver node which was set to transmit continuously at 0dBm at 2.45GHz (shown in Figure 3-12). The tests were performed using volunteer A (Chapter 1, Table 1-1) to identify the body effect on the received signal level. The measurements were taken in free space to minimize possible multipath effects from nearby objects. The CW transmitter was placed at six different locations (4 locations on the leg and 2 locations on the arm) of the human body and the participant was asked to assume different stationary positions of the arms and legs. These positions are characteristic of running movements [126]. At each position, the RF analyzer was connected to a computer and the Received Signal Strength Indicator (RSSI) was recorded using the RF explorer program as shown in Figure 3-13. Both the transmitting and receiving antennas were vertically polarized when the human stands upright with hands by the side. The one exception is for the transmitter placed on the top of the foot.

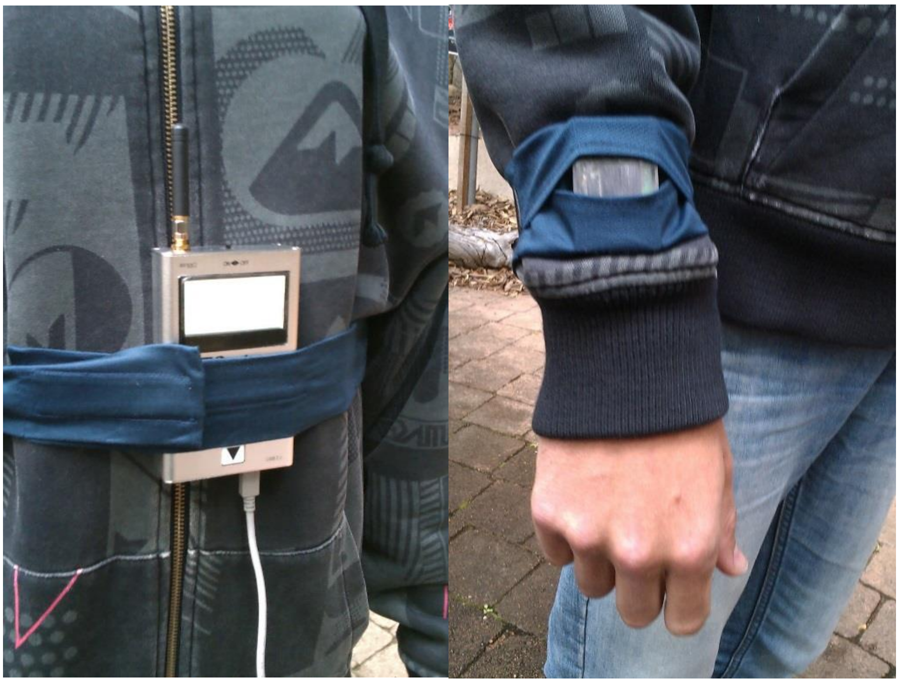
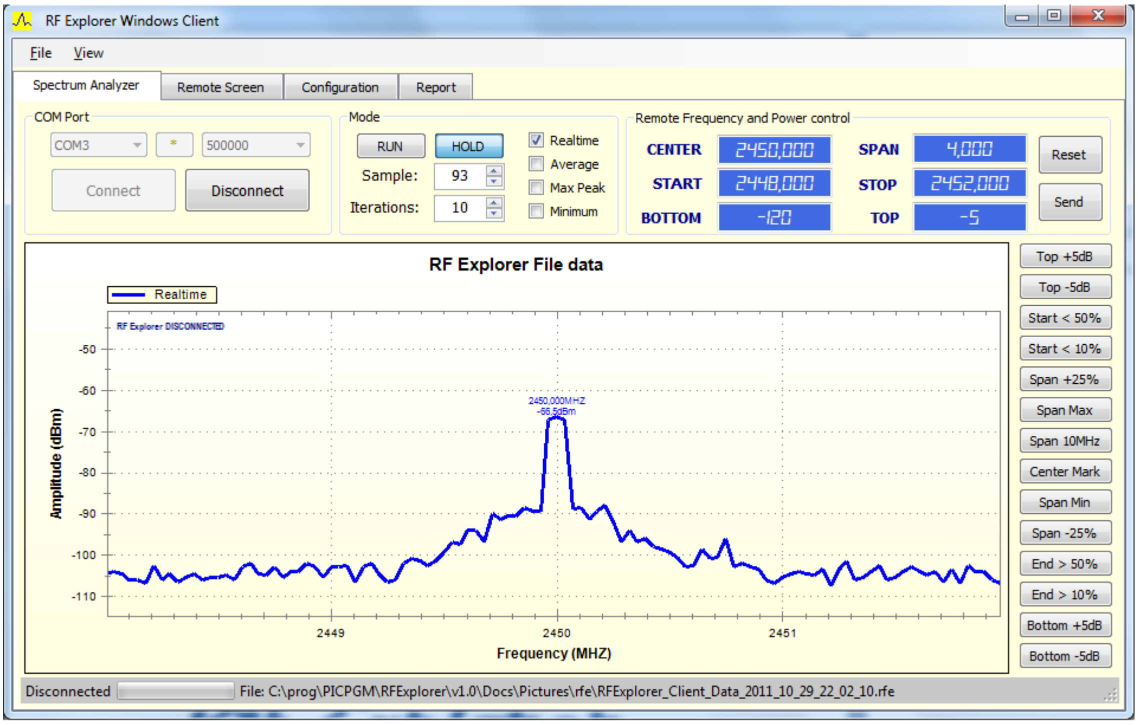


Figure 3-12 The RF spectrum analyzer showing an SMA connector and antenna (Left side). And the wireless node inside a clothing mounted on the wrist (Right side).



RF Explorer Windows Client

COM Port: COM3 500000

Mode: Sample: 93 Iterations: 10

☑ Realtime
□ Average
□ Max Peak
□ Minimum

Remote Frequency and Power control

CENTER 2450,000 SPAN 4,000
START 2448,000 STOP 2452,000
BOTTOM -120 TOP -5

Disconnected File: C:\prog\PICPGM\RFExplorer\v1.0\Docs\Pictures\rfe\RFExplorer_Client_Data_2011_10_29_22_02_10.rfe

Figure 3-13 Screenshot of the RF spectrum analyzer software shows the received signal and available monitoring functions.

The transceiver node was attached with a fabric band, to make sure the wearable wireless node is stable during movement and for ease of body mounting. The following locations were used: the outside part of the wrist and arm (above the elbow), the upper surface of the foot, the lower and upper parts of the leg and on the thigh just above the knee. These locations are shown in Figure 3-14.

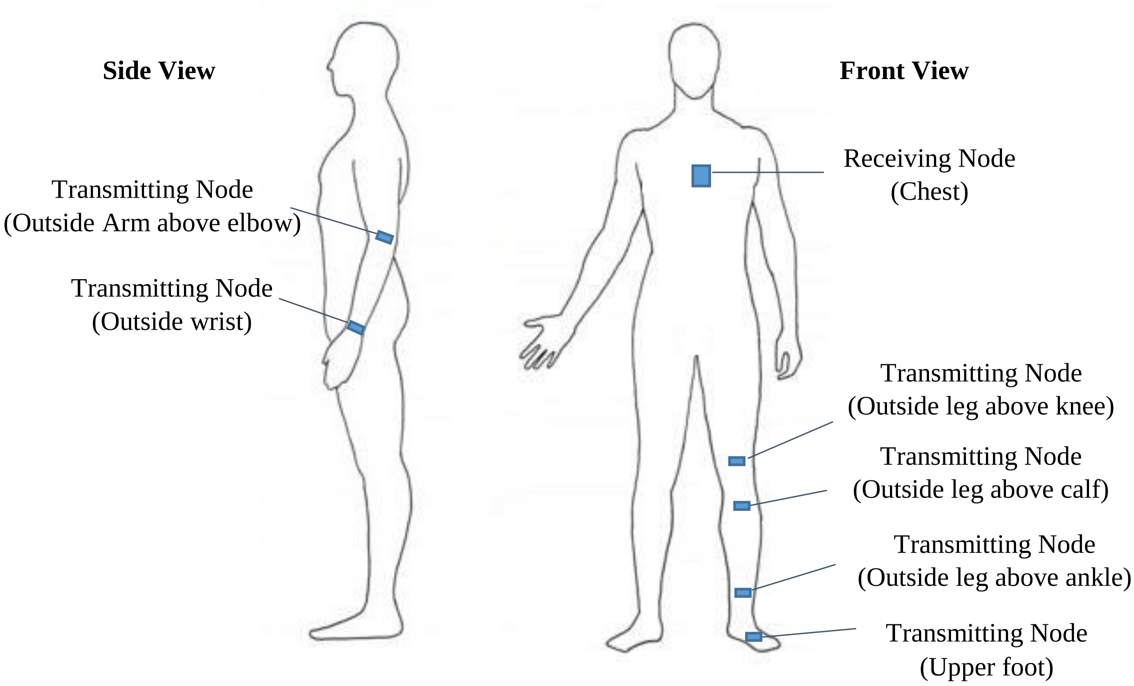

Side View
Front View
Receiving Node (Chest)
Transmitting Node (Outside Arm above elbow)
Transmitting Node (Outside wrist)
Transmitting Node (Outside leg above knee)
Transmitting Node (Outside leg above calf)
Transmitting Node (Outside leg above ankle)
Transmitting Node (Upper foot)

Figure 3-14 Wireless sensor node locations.

For all of the six wireless sensor locations, the participant was asked to assume a total of 24 positions for the cases of the sensors mounted on the leg and the arm. In these positions, the leg and arm joints were positioned on a horizontal axis that is perpendicular to the front of the body for maximum rotation in order to simulate the running and walking positions through the likely arm and leg movements. Figure 3-15 shows the leg and arm positions used in this study, and the position number. These positions were repeated after changing the wireless node location to measure the received signal level at each location. The leg and arm positions were rearranged to display the received signal level during the leg and arm movement from the very far point behind the body to the closest point in front of the receiving node attached to the chest. Figure 3-16 shows the received signal level for each of those positions.

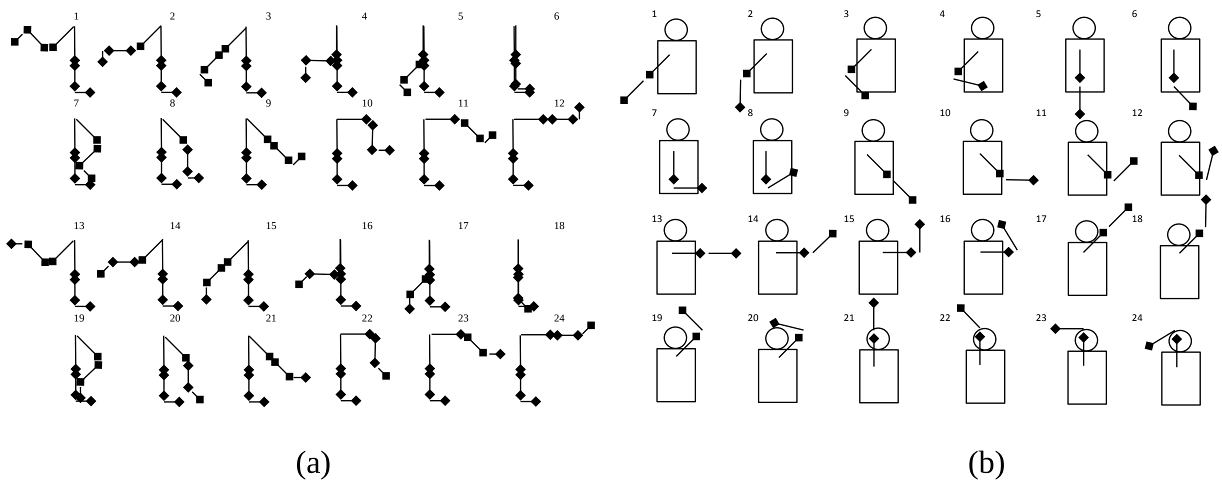
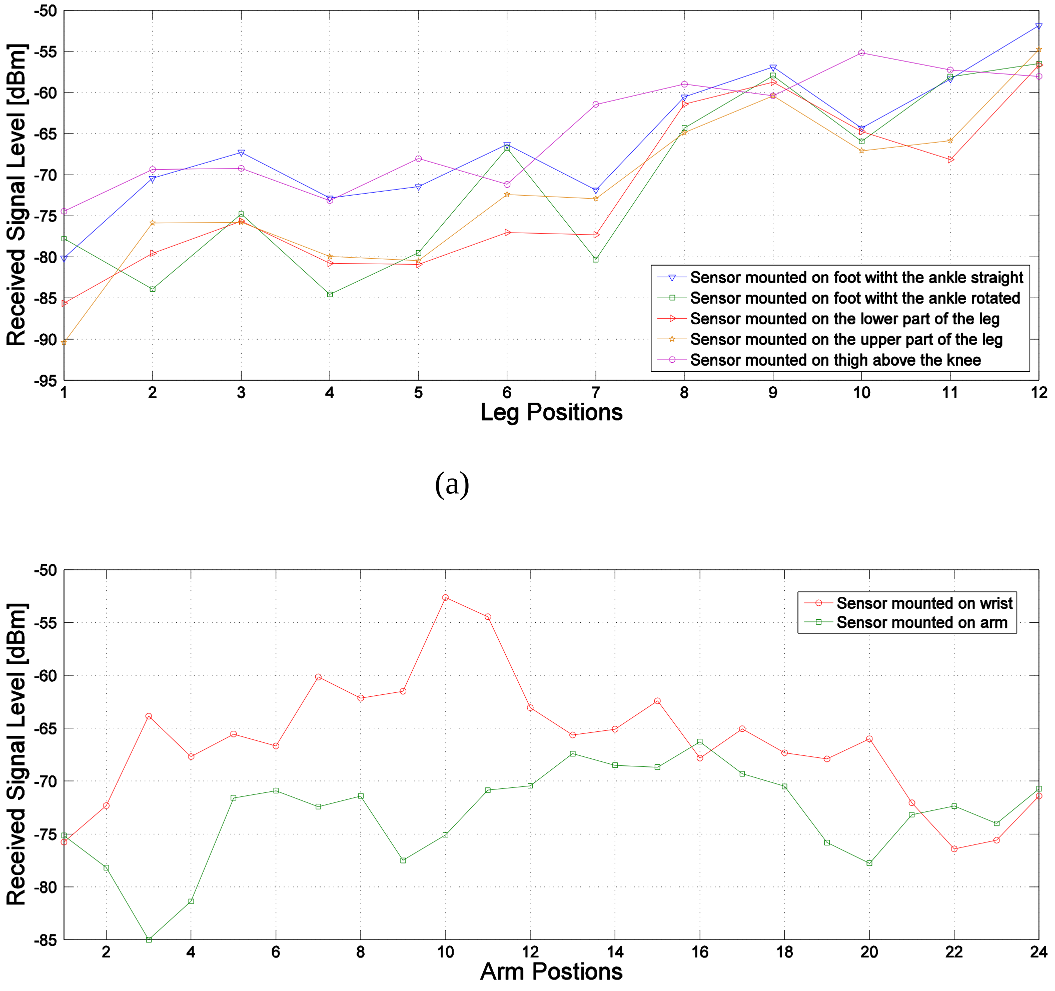


Figure 3-15 Transmitter locations on the body shown by "•" at (a) Leg positions; (b) Arm positions.

Figure 3-16 (a) shows the received signal level for all sensor locations and for all leg positions assumed by the human during running. For all of the leg sensor locations, the received signal level increased relative to the positions order [that are shown in Figure 3.15 (a)] to reach a maximum value between -52dBm and -58dBm. Figure 3-16 (b) shows the received signal level for sensor mounted on wrist and on arm locations and for all limb positions. The results show that the highest signal level during the arm movement was around -65dBm at positions 13 to 16 [as shown in Figure 3-15 (b)]. When the sensor is attached to the wrist, the received signal level increases with the arm movements to reach a highest power level above -55dBm at position 10 & 11 (when the wrist faced the chest) and decreases to a value of -72dBm as the arm moves up and away from the chest (position 24).

These results were analysed using a MatLab program to show the best possible location for a wireless node on a limb to ensure reliable wireless communication with another node attached to the chest during human walking and running. Given these conclusions from the literature [127] and [128], which were obtained for similar test conditions, BSN operating at 2.4GHz with a receiver sensitivity level of -75dBm suggest that > 95% reliability can be expected with a false alarm rate of < 1%.

The relative frequency distribution of all recorded RSSI measurements for all sensor locations and at all positions was calculated to identify the highest percentage of link reliability. Figure 3-17 shows that 74% of the total signals recorded have a received signal level greater than -70dBm when the sensor is attached to the wrist while only 20% when it is attached to the arm above the elbow. This means that during all possible positions of the human wrist [shown in Figure 3-15 (b)] the possibility of reliable wireless communication is more than three times when compared to the situation with the sensor attached to the arm. The results of 68% and 35% of a received signal of greater than -70dBm was achieved when the sensor was attached to the thigh and lower leg positions respectively (see Figure 3-14).



(a)

(b)

Figure 3-16 Received signal level for all (a) leg positions; (b) arm positions.

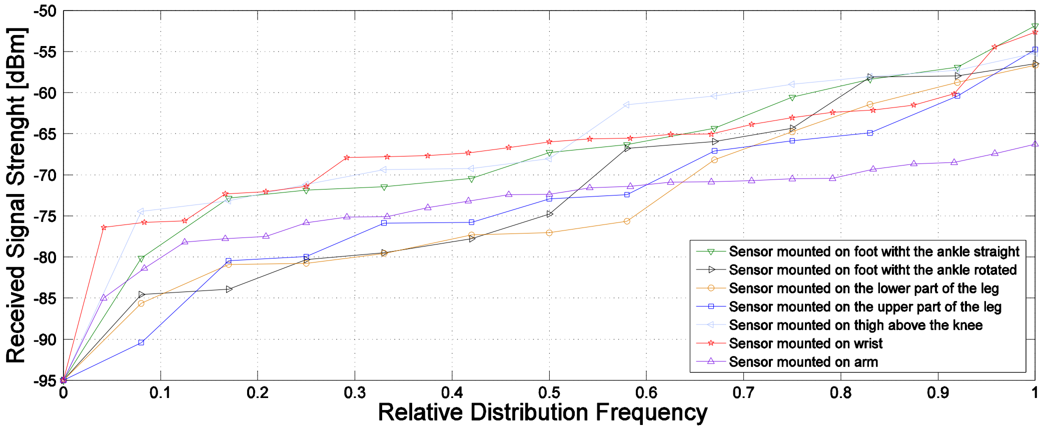


Figure 3-17 Relative frequency distribution for the received signal level at all node locations.

The position of the arm or leg and the mounted transmitting node location affect the signal level and can help to identify node-distributions that favour the existence of reliable burst wireless communications when the receiving wireless node is attached to the chest. Body measurements were taken and a geometrical and mathematics analysis software (GeoGebra) was used to calculate the distances and angles between the receiving and transmitting wireless nodes, as shown in Figure 3-18.

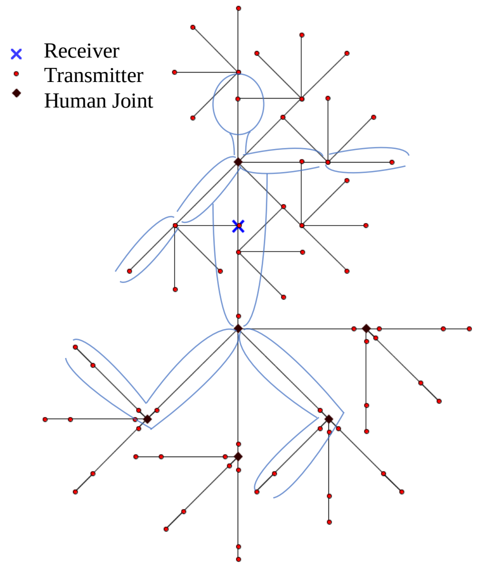
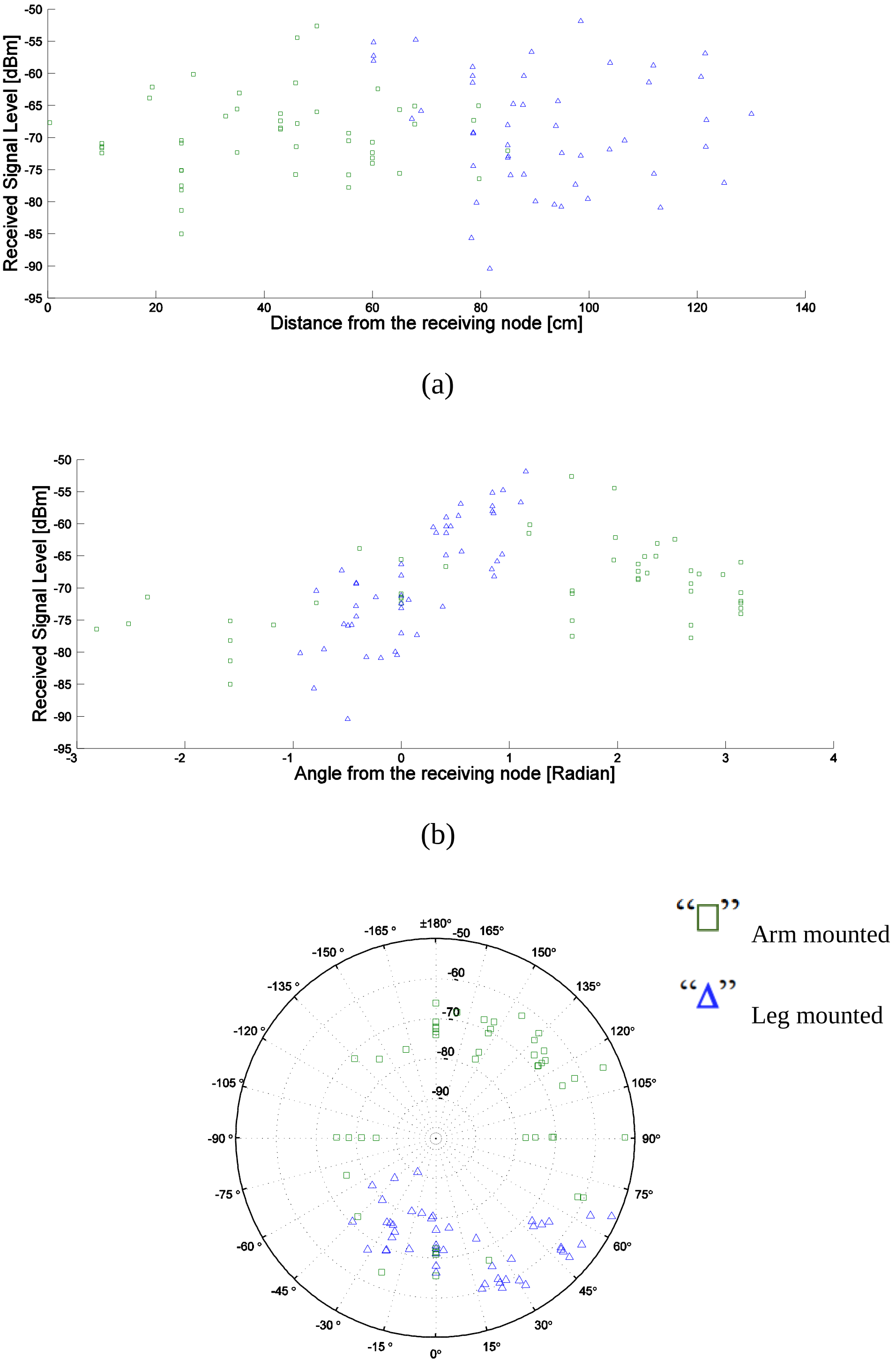


Figure 3-18 Human stick model shows the receiving and transmitting nodes for all leg and arm angle positions.

Figure 3-19 illustrates the relationship between the received signal level and the distance/angle between the receiving and transmitting nodes. It can be seen that while the distance to the chest is shorter when the sensor is installed on the wrist and arm (green square points) compared with the leg sensor locations (blue triangle points), the signal level is randomly distributed across the distance axis and the same signal level can be obtained at different distances for both leg and arm sensor locations as shown in Figure 3-19 (a). In Figure 3-19 (b), the relationship between the received signal level and receiver angle shows more consistency. In the case of the leg, the received signal level increased when the angle changed from the non-line of sight (NLOS) to LOS position. The result resembles a sinusoidal function for the arm as the angle changes. This angular relation is explained in a polar plot of the received signal level in Figure 3-19 (c). The received signal level is higher for the LOS positions. A received signal level of at least -70dBm was obtained for all limb positions at angles of (17 to 65 degrees) and (110 to 150 degrees) for the leg and arm sensor locations respectively. The repetitive movement of the human leg and arm during running or walking makes it easier to achieve reliable communications within those angular windows. *(Note: There are two angular windows).*



(a)

(b)

(c)

Figure 3-19 Received signal level as a function of the (a) LOS and NLOS distance, (b) angle in radians and (c) angle in degrees between the receiving and transmitting nodes for “arm mounted” and “leg mounted” transmitter positions.

3.7 Sensor window of availability calculations

Data recorded for each running position described in Section 3.5 revealed a set of angular windows for each sensor location at which a reliable wireless communication can occur (RSSI ≥ -75dBm). These angular windows are repetitive in a periodic movement such as running and can be converted into time windows once the swing time of the runner has been determined. A method to calculate the time windows for different runners was defined by comparing the accelerometer data for a number of swings in order to provide the most accurate swing value, as shown in Figure 3-20. In one example, and in the calibration mode of the proposed technique, the swing time used to calculate the time windows for each wireless node was set to 400ms. This value was selected based on the studies discussed in Chapter 2, Section 2.7.

Associating the RSSI measurements for the angle windows with the swing time calculations while assuming a constant limb speed through the swing action, these angular windows can be presented as a function of the time. Table 3-4 shows the time windows relative to the swing time during running. The cycle takes two swing times and begins at 0ms when one leg and opposing arm stretch far behind the body while reaching the extreme position in front of the body at 400ms and ends back at 800ms (see Figure 3-21).

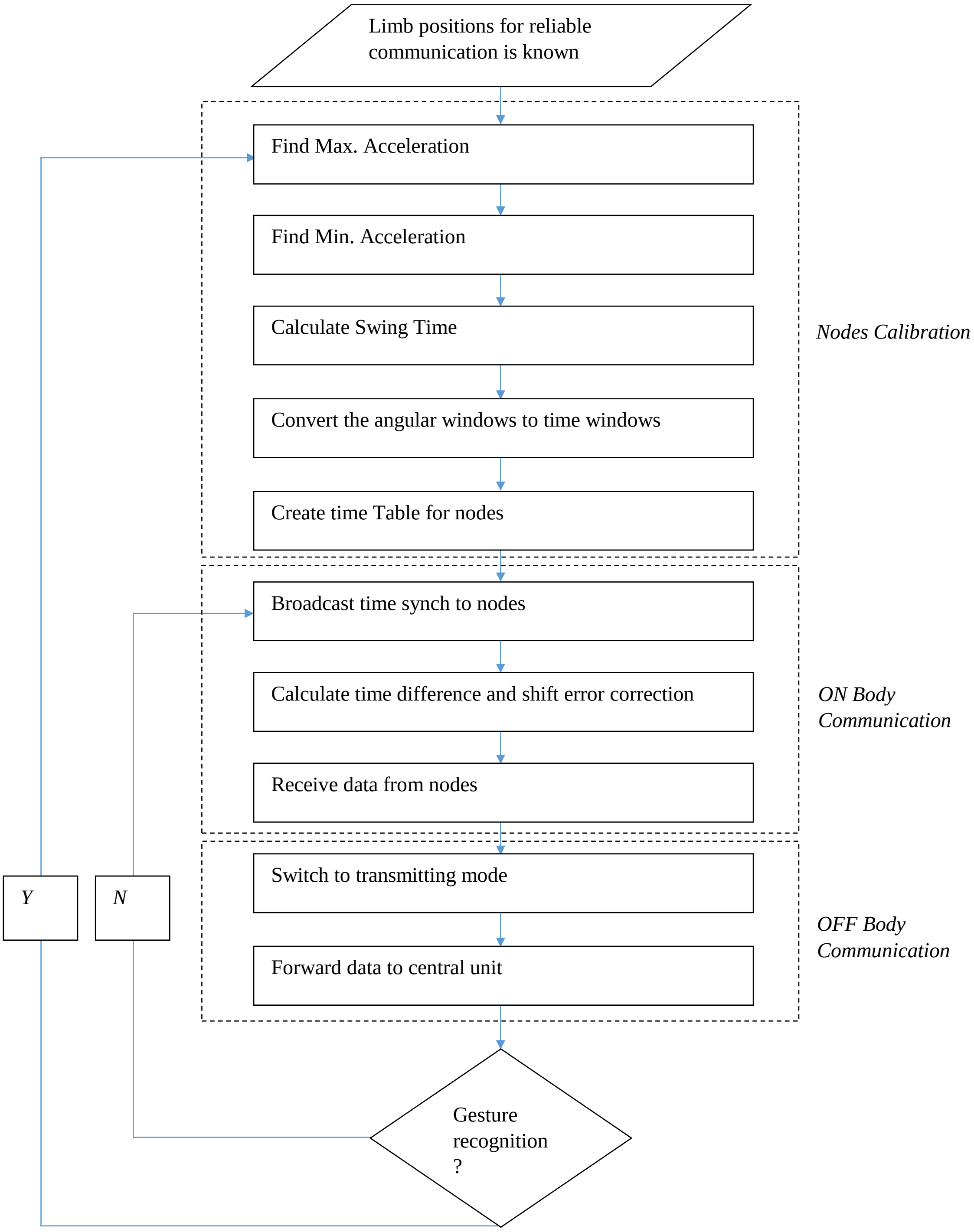

Limb positions for reliable communication is known
Find Max. Acceleration
Find Min. Acceleration
Calculate Swing Time
Convert the angular windows to time windows
Create time Table for nodes
Nodes Calibration
Broadcast time synch to nodes
Calculate time difference and shift error correction
Receive data from nodes
ON Body Communication
Switch to transmitting mode
Forward data to central unit
OFF Body Communication
Y
N
Gesture recognition ?

Figure 3-19 Flow chart describing the node calibration and (ON body and OFF body) communication process of the hub node.

Table 3-4 Sensor time windows relative to the swing time of one running cycle start at t=0 and ends at t=400ms. Each sensor location shows a different start and end times for the reliable windows connection within the 400ms cycle keeping the received signal thresholds < -75dBm.

Time (ms)	Sensor mounting location					
	Wrist	Foot	Lower Leg	Upper Leg	Thigh	Arm
Start time	121	223	277	245	245	382
End Time	400	400	389	353	335	384
Total Time	279	177	163	108	90	2

In Table 3-4, the longest time windows for continuous reliable communication is provided for a wireless node attached to the wrist of the human body. The existence of LOS between the chest and wrist during the course of the second swing allows the nodes to preserve consistent wireless connectivity. Converting these time windows to data values determines the connection time and so the transmission capacity of each sensor location in a swing period. It shows that for a maximum data rate of 250kbps, a total data traffic of nearly 8.72kB can be provided when the sensor is attached to the wrist during a single swing time, while only 62.5bytes for arm positions.

Gait analysis requires the capture of body movement actions, such as walking or running. As indicated in Chapter 2 - Section 2.7, the most common methods use video recording and subsequent image processing using complicated analysis algorithms [113]. The new techniques for capturing the human movement are based on using three axis accelerometer sensors with 100 samples/s per channel. Each sensor records 80 samples per two swing actions and sends these samples at the allocated time window to the central unit which in turn will sends them off the body (locally or to the cloud) for further analysis. The data sent from each sensor (excluding the overhead) was 480bytes because each single 3 axis accelerometer sample equals 6bytes. With a data rate of 250kbps, each sensor will require a time burst duration of 15.36ms to transmit the sampled data to the central node. This equates to five times the period available to retransmit the collected data. The hub node sends synchronization pulses each 400ms;

however, the nodes receive these pulses each 800ms due to the swinging action of right and left limbs during running. As a result, the nodes have to maintain time synchronization for 800ms depending on their local clock and this might produce a time shift from the hub node time, which can be corrected using a simple version of two nodes time shift error correction algorithm [90]. Alternatively, packet level time synchronisation via time stamps, can be used to achieve microsecond accuracy.

Figure 3-21 shows the required time Table calculations for synchronization, data collection and transmission. As the running is a periodic movement, only three swing actions are presented. After calibration, it is not significant whether the right or left leg/arm will give the first swing. In this example, the first swing start at t=0 and was assumed to occur when the right leg and left arm move forward and the second swing is when they reverse position. At the first swing, the central node broadcasts its time and a time Table of transmission for each node so that each wireless node is synchronized with the central node. After synchronization, the node switches to receive mode in order to collect the data from the wireless sensors attached to the left wrist and right leg.

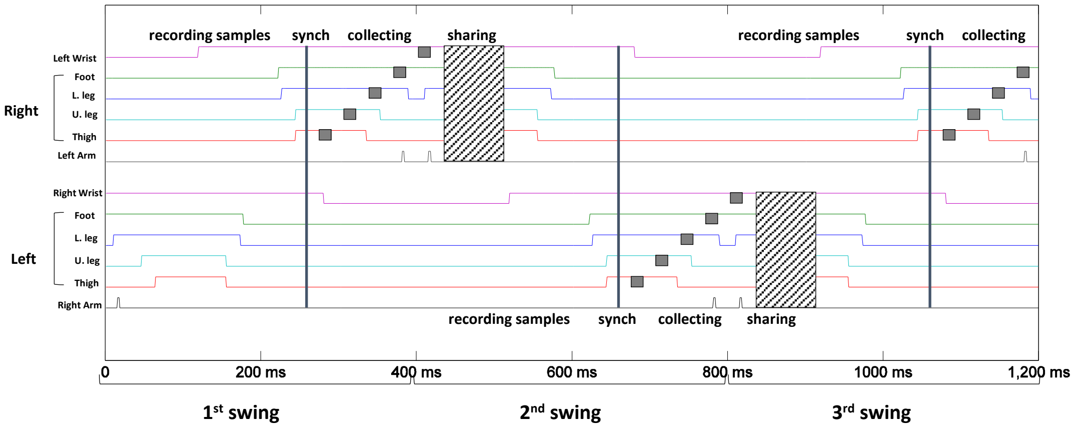


Figure 3-20 Wireless node time allocation and synchronization during running.

The process of synchronization and data collection takes nearly 170ms, after which the central wireless node forwards this information to a local computer in order to provide quasi-real-time feedback data. In this model, the delay between the sensor signals and computer recorded data is less than one second. The same procedure is repeated at the second swing; however, this time the data sets collected and transmitted were for the right wrist and left leg and the process continues along the running action.

3.8 Dynamic wearable node measurements

Practical experiments were performed to investigate the effect of body movements, sensor locations and wireless setting on the successful transmission rate during running. The test involved the participation of human volunteers, who were asked to wear the wireless transmitter and receiver nodes and then run in a sports field; an outdoor environment was chosen to minimize possible multipath effects from nearby objects. The volunteers were fully informed about all relevant information of the project as per the code for responsible conduct of research (as described in Chapter 1, Section 1.2). The participants were asked to run a short distance at their moderate speed starting from a stationary position and ending at another stationary position; this distance was long enough to let the runners reach the maximum speed without exceeding the limits. The swing speed was calculated instead of the running speed to find a direct relationship between the speed of the limb and the link reliability. In fact, the swing speed varies little with running speed [112].

The nodes were configured to continuously transmit and receive data at the same power level and frequency [0dBm and 2.4GHz] and data rates of 250kbps, 1Mbps and 2Mbps, respectively. The running data of five runners of different styles and abilities were analysed to assess variability. The wearable nodes were attached in fabric bands to ensure stability during movement and for easy on body mounting. The transmitting node was placed at the locations shown in Figure 3-14.

During these tasks, body movement acceleration data with a sampling rate of 100Hz (acceleration setting are explained in Section 3.3) were logged and then sent from the transmitting node, changing the transmitter location around the legs and arms of the human body allow link comparisons to be made for different node positions. The data were also logged by the receiving node located at the chest. The data were downloaded from the accelerometer sensor after each test via a virtual COM port using Tera Term freeware, and imported into further processing software (MatLab). The transmitted and received packets were compared in order to calculate the transmission reliability percentages. These tests were conducted using the system described in Figures 3.1 &

3.2. The Nordic transceiver was designed to send and receive payloads in a packet format composed of preamble, address, packet control, payload and CRC (cyclic redundancy check). These features ensure that each packet detected is new and there are no bit losses. If errors are detected, the packet is ignored. This allows a measurement of link reliability by calculating the packet loss ratio. Figure 3-22 shows the packet format, where the MSB (most significant bit) is located on the left side.

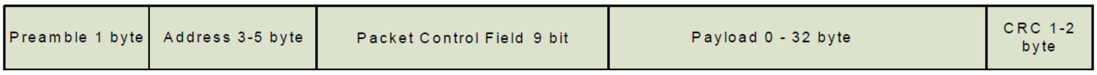


Figure 3-21 Packet format with payloads (0-32 bytes).

The preamble is a one-byte long sequence that is used to synchronize the receivers demodulator to the incoming bit stream and ensure enough transitions to stabilize the receiver. The address is 4-5 bytes long and can be configured as a unique code for each transceiver to prevent crosstalk between multiple wireless nodes and detect the source of transmission. The packet control field includes a 6-bit payload length field that specify the length of transmitted/received payloads in bytes, a 2 bit PID (Packet Identification) which is used to detect if the received packet is new or a retransmission, and 1 bit NO-ACK (No Acknowledgment) flag to request acknowledgement of any received packets. The CRC is an information code used for error detection that can be 1 or 2 bytes long. It is used mainly to check the overall sent and received bits for a packet.

3.8.1 Swing time effect

To study the effect of running style of different participants (more specifically the swing time) with regards to one sensor location on the wireless link connection reliability, a data rate of 2Mbps was selected to transmit the information captured by the wireless nodes installed on the arm (transmitter) and chest (receiver) of five different participants (participants A, B, C, D and E were selected for these tests). Table 3-5 shows the successful data transmission rate, and the average swing time calculations of the five participants when running at uniform speed. The physical parameters of the participants (such as weight and height) are listed in Chapter 1, Table 1-1.

Swing time measurements were made using an algorithm developed in MatLab (The code is given in Appendix B). The algorithm checks the minimum acceleration values of a sample and, after selecting the lowest one, defines a pointer to indicate its position. The same process is repeated for another sample window to calculate the time difference between the first and second pointers of the lowest values which equals to the swing time. Figure 3-23 describes the procedure used to calculate the different swing times. The sample window was set manually (after monitoring the gait movement and calculate each gait running cycle) for each runner. Nevertheless, it could be defined by using another calibration algorithm.

Table 3-5 Participant pool and successful packet transmission rate from the arm (transmitter) to chest (receiver) during running.

Participant ID	Swing Time (ms)	Success Rate
A	510	54%
B	490	56%
C	400	85%
D	420	95%
E	400	98%

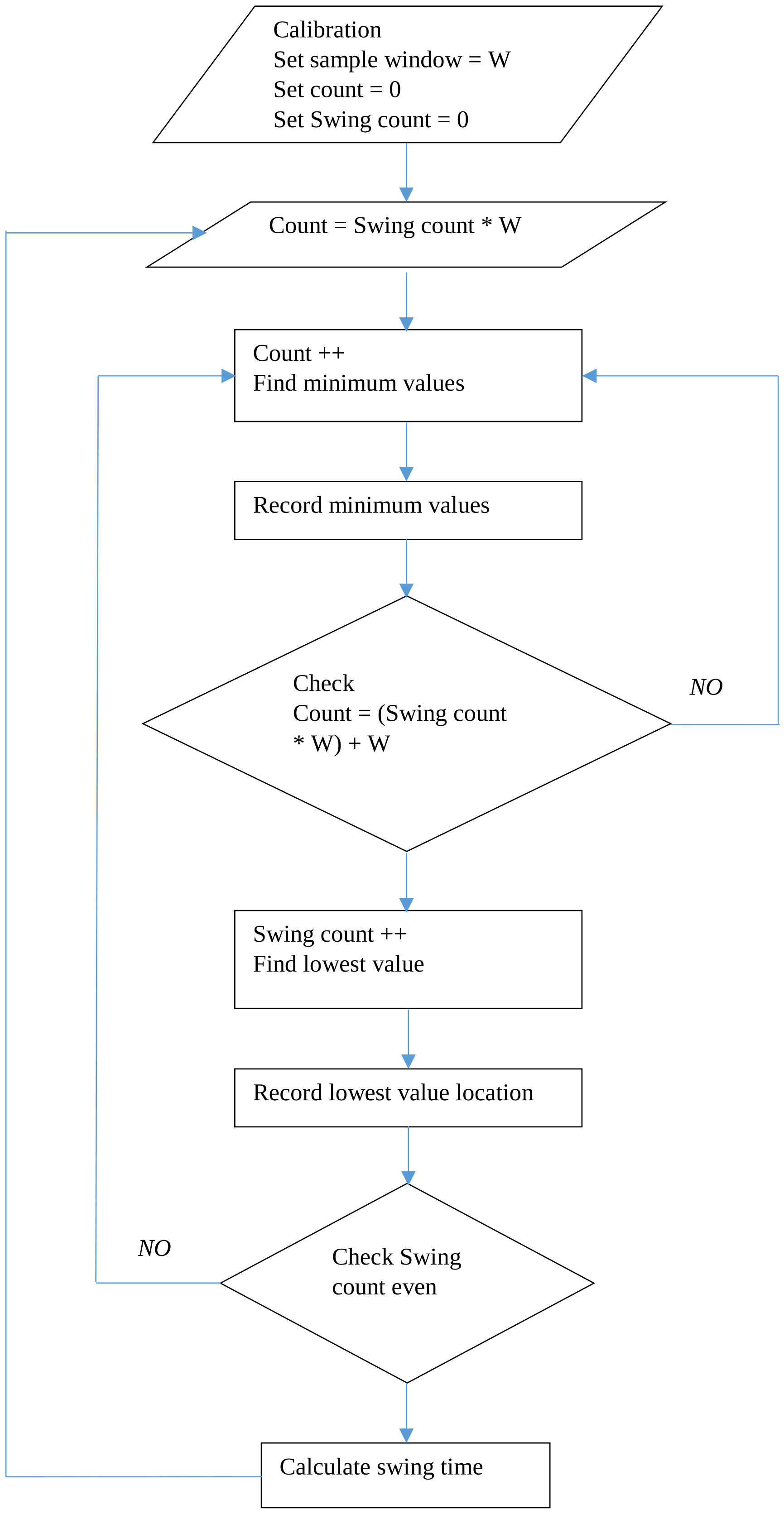


Figure 3-22 The swing time calculation algorithm runs in an infinite loop. The swing time is calculated for each pass through the loop.

The differences in swing time are derived from the participants' running style which is mainly influenced by the running experience and the characteristic of limb movement during the action of running. The data losses of the wireless nodes for different runners are strongly correlated (correlation coefficient of 89%) with the longest or shortest swing time a runner can have.

Figure 3-24 shows the recorded data of participant A at the arm location after applying the swing time calculation algorithm. The shadowed area represents the starting and stopping times required by the participant to get into a moderate running speed at constant velocity. These shadowed areas were not considered to calculate the average swing time of participants, since the swing time and sensor location will be different to those recorded when the participant is running at constant speed.

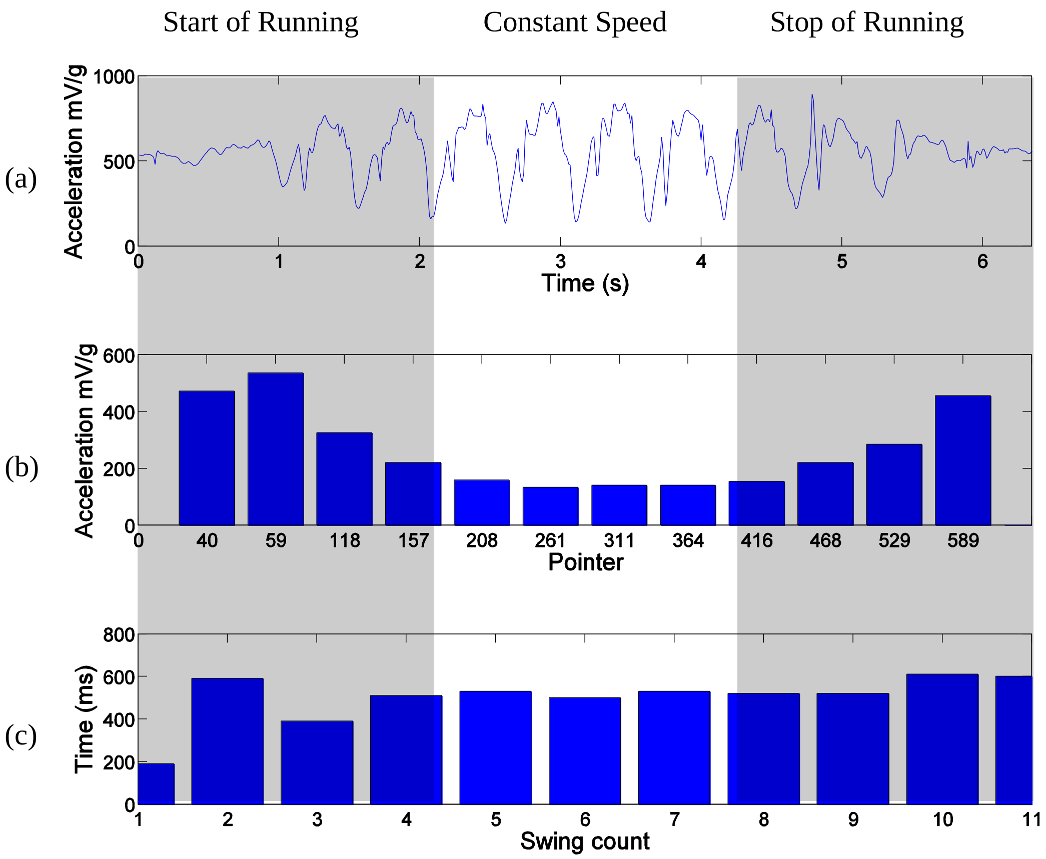


Figure 3-23 Participant A (a) acceleration data at arm location (b) lowest values in the sample window (c) swing time calculated values.

3.8.2 *Sensor location effect*

To examine the effect of different sensor locations and data rate speeds on the wireless link reliability, a single subject (Participant A) was asked to wear the wireless sensors in six different locations for channel data rates of 250kbps, 1Mbps and 2Mbps. The resultant eighteen recorded sessions from the wireless transmitter and receiver nodes revealed the data percentage received in each sensor location and for each data rate value used during sprinting.

Figure 3-25 shows the acceleration data transmitted and received for each wireless node location and data rate value during running. These recorded data are acceleration measurements for a rhythmic running action for all sensor locations. The received data percentages are directly related to the channel data rate and the highest data loss occurred when a data rate of 2Mbps was used.

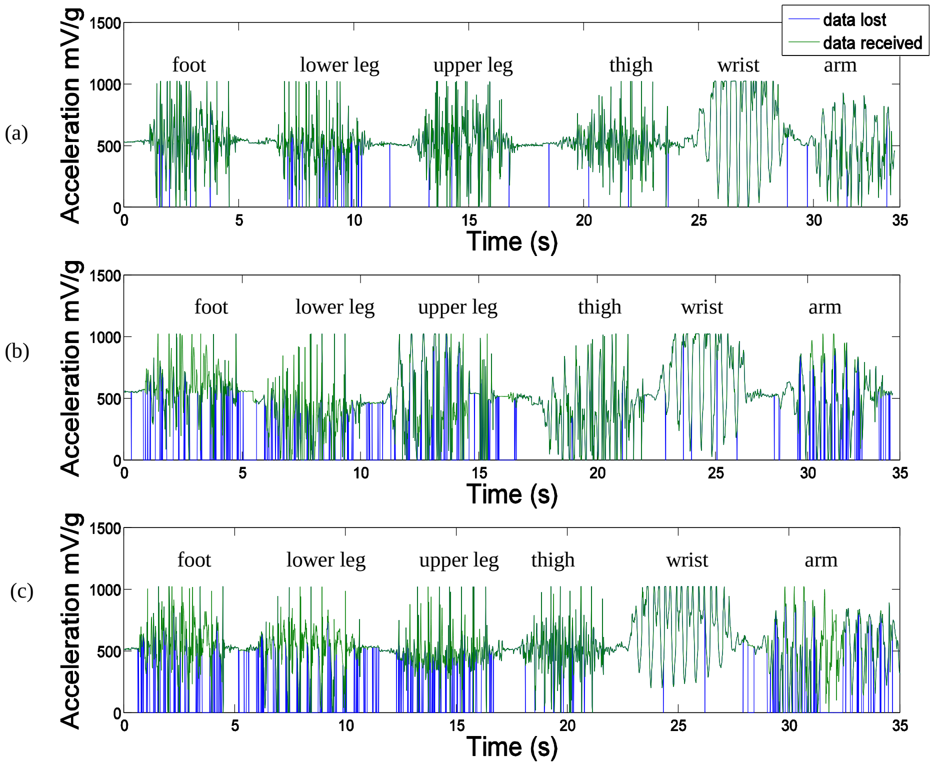


Figure 3-24 Lost data (Blue) and received data (Green) during running for multiple body locations using a data rate of (a) 250kbps (b) 1Mbps (c) 2Mbps. The same sensor was placed on the foot, lower leg, upper leg, thigh, wrist and arm sequentially while the participant ran along 20m, starting and ending at a stationary position.

The swing speed at each sensor location was determined by applying the swing time algorithm. Using the algorithm, checking for the maximum instead of the minimum values gives more accurate swing time calculations for acceleration signals with more categorical peak patterns.

Figure 3-26 shows the link reliability obtained for each sensor location and the data rate. The lower the data rate, the higher the link reliability, which seems logical since higher sensitivity values are required when increasing the data rate (-82dBm for 2Mbps, -85dBm for 1Mbps and -94dBm for 250kbps). The wrist provides the most consistent results over the data rate changes with a reliability of 98%. This is obviously because the sensor on the wrist during movement continuously provides a received signal level above the lowest sensitivity threshold for each different data rate. Lower leg was the most critical location since reliability values of 25%, 42% and 94% were obtained for data rates of 2Mbps, 1Mbps and 250kbps, respectively.

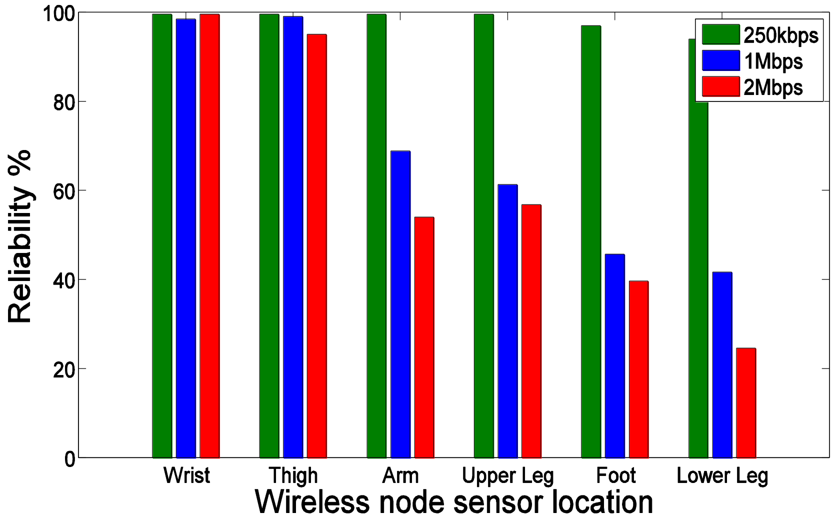


Figure 3-25 Link reliability for all sensor locations and three data rates for participant A.

Table 3-6 gives a general idea of the wireless body area network performance with respect to data rate. The highest reliability value (98%) was obtained when using a channel data rate of 250kbps compared with channel rates of 1 and 2Mbps which produced a network reliability of 70% and 62%, respectively.

Table 3-6 Aggregate percentage of the reliability of participant A for data rates of 250kbps, 1Mbps and 2Mbps.

Data Rate	Aggregate successfully received sample rate %
250kbps	98
1Mbps	70
2Mbps	62

3.9 Conclusions

This chapter outlined the hardware design of the modified sensor, along with the node settings, data collections and analytical techniques. Wireless link reliability values obtained from air and body connectivity tests carried out at different distances were also compared. It discussed the results obtained from static measurements conducted on different wireless body channels, the effect of different test subjects, sensor location and data rate on the wireless link quality during running. It showed the design of the swing measurements algorithm that adopts the acceleration data in calculating the swing time and steps count.

Bearing in mind the results obtained from transmission losses during the athletes' movements and the corresponding reliable point of communications were explained in this chapter. The following two chapters present the methods and smart algorithm designs that were adopted to increase the network reliability.

Some of this work was partly published in the author's publications [129-133].

CHAPTER 4

SELF-CALIBRATING AND GESTURE BASED TRANSMISSION ALGORITHMS

4.1 *Introduction*

This chapter discusses the operations of different embedded algorithms on the body sensor network described in Chapter 3. The effect of the acknowledgement protocol on the wireless channel reliability used for both link to link and multi-link to link communications in a sensor network is reported, and the disadvantages of this technique related to the largely increased generated traffic and reduced channel efficiency are discussed. A reliable BSN based on smart transceivers and the hub capacity of connectivity with multiple sensors through different logical addresses is included. Basic and smart algorithms (such as standard TDM, self-calibrating, dynamic TDM, and gesture triggered communication) were developed to instantly find reliable communication points for different body channels and different subjects during the course of running. These algorithms were implemented and validated through practical experiments.

4.2 *Acknowledgement and retransmission*

The auto acknowledgement feature was enabled and the enhancement of the network success rate was measured. The auto acknowledgement protocol enables the transmitter to wait for the receiver acknowledgement to record when the packets were received successfully. The acknowledgment packet handling is as follows [134]:

- The transmitter starts the operation by sending a data packet to the receiver. The auto acknowledgement protocol automatically switches the transmitter to receive mode to wait for the ACK packet;
- If the packet is received by the receiver, the receiver auto acknowledgement protocol automatically assembles and transmits the ACK packet to the transmitter and then returns to receive mode;

- If the ACK packet is not received; the transmitter sends again the original data packet after a programmable delay and switches to receive mode to wait for the ACK packet;
- The process continues for a pre-programmed number of retransmissions until acknowledging successful delivery or until reaching the predefined maximum number of retransmissions;
- If the transmitter does not receive the ACK packet and after reaching the maximum number of retransmission attempts, the data is discarded and the transmitter starts the transmission of a new data packet. This reduces stack overflow problems when the line-of-sight communications is delayed for a significant time period.

Different maximum numbers of retransmission attempts were programmed into the system and tested with a sampling rate (100 samples per second). Changes in the number of retransmission attempts involved different time delays to ensure successful packet delivery, and therefore, had influence on the observed sampling rate. To ensure a constant 100Hz sampling rate, the time delay of retransmissions was differentiated from the sampling rate at each packet transmission (i.e. acceleration samples are sent every 10ms, even if retransmissions were involved). The test was conducted on participant F with the transmitter node placed at the arm position and the receiver at the chest. The data rate was set at 1Mbps.

Figure 4-1 shows the total number of packets transmitted through the link and received sample rate. The reception rate rises from 69% to 88% with an increasing number of retransmissions (Figure 3-23 shows the swing time calculation algorithm). While a success rate of nearly 69% have 200 original packets with no retransmission, a success rate of nearly 88% requires more than four times that number. The retransmissions increase the traffic load on the wireless link

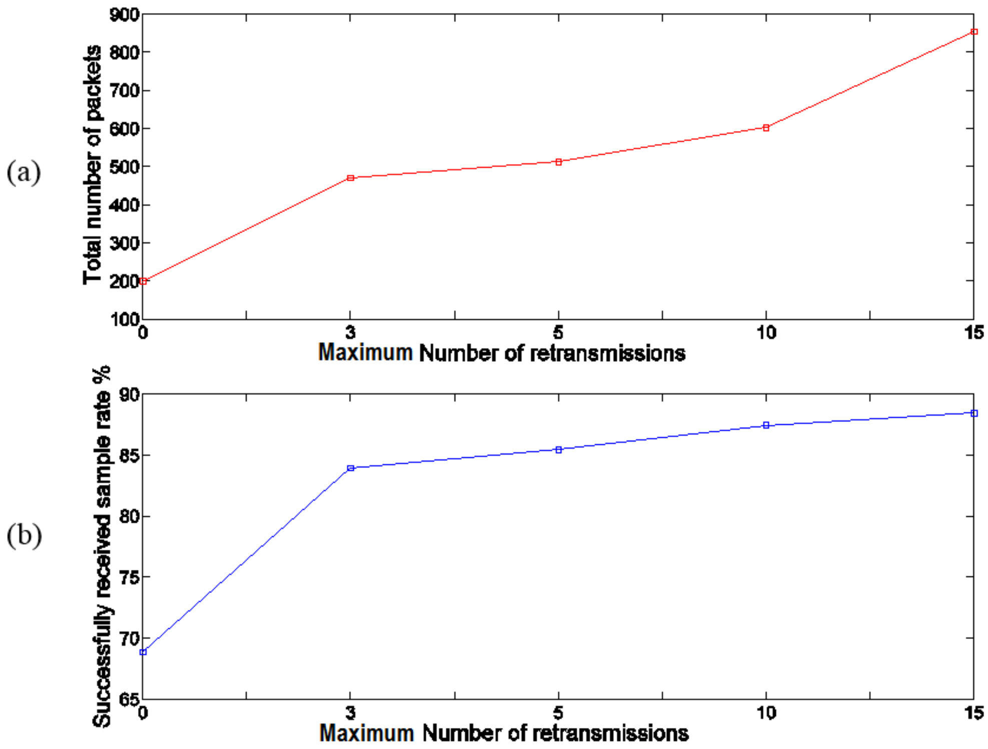


Figure 4-1 Maximum number of packet retransmissions with respect to the: (a) total number of packets transmitted and retransmitted (b) the successfully received sample rate.

Table 4-1 provides an indication of the traffic generation for different retransmission settings. For the 200 packets, the ideal goodput (i.e. the number of information bits delivered by the network to a certain destination, excluding protocol overheads and retransmission packets) is 2.4kbps, which can be calculated by multiplying the sample payload (24 bits long) and the sampling rate (100Hz). From Table 4.1, it can be seen that the goodput is higher when the number of packet retransmissions increases. Nevertheless, for the 15 retransmissions setting in Table 4.1, this small improvement required more than 3.4 times the original traffic speed (i.e. that delivered in one second). This data loss reduces the link capacity and is not practical for applications where continuous data are sent. This is more of a problem in large networks where more than two wireless nodes are communicating, but it would be valuable for applications in which a single or a small number of transmissions happens in one second and link reliability is important.

In this experiment, only one sensor location (on the arm) with one data rate (1Mbps) was chosen, to show clearly the wireless link enhancement for different retransmission attempts. It showed improvement in the link reliability on the account of traffic lost and channel capacity. Other sensor location will show the same behavior as that obtained from the arm location. Therefore, it can be said that different running styles, sensor locations, and data rates would have differing effects on the link reliability and would require different retransmission schemes.

Table 4-1 Wireless link performance for different retransmission settings at Arm sensor location of participant F during running at moderate speed.

Maximum retransmission	Total traffic kbps	Delivered goodput kbps	Lost (retransmitted) traffic kbps
0	2.40	1.64	0.76
3	5.64	2.00	3.64
5	6.14	2.04	4.10
10	7.22	2.09	5.13
15	10.24	2.11	8.13

4.3 BSN of smart transceivers

The network uses a hub node positioned on the chest which communicates with the four nodes located on the wrists and thighs, as shown in Figure 4-2 (b). Data received at the hub node can be identified by the unique node transmission address. As described in Chapter 3, Section 3.2, the UART interface (CP2102) at each node allows wired data transfers for real-time monitoring and further processing during or after the completion of the sport activity. The tests involved the participation of three human volunteers (G, H and I as shown in Chapter 1, Table 1-1) and were conducted under the project code for responsible conduct of research (Ethics approval ENG/20/13/HREC). The nodes were configured to transmit and receive data at the same power level, frequency and data rate (0dBm, 2.4GHz and 2Mbps).

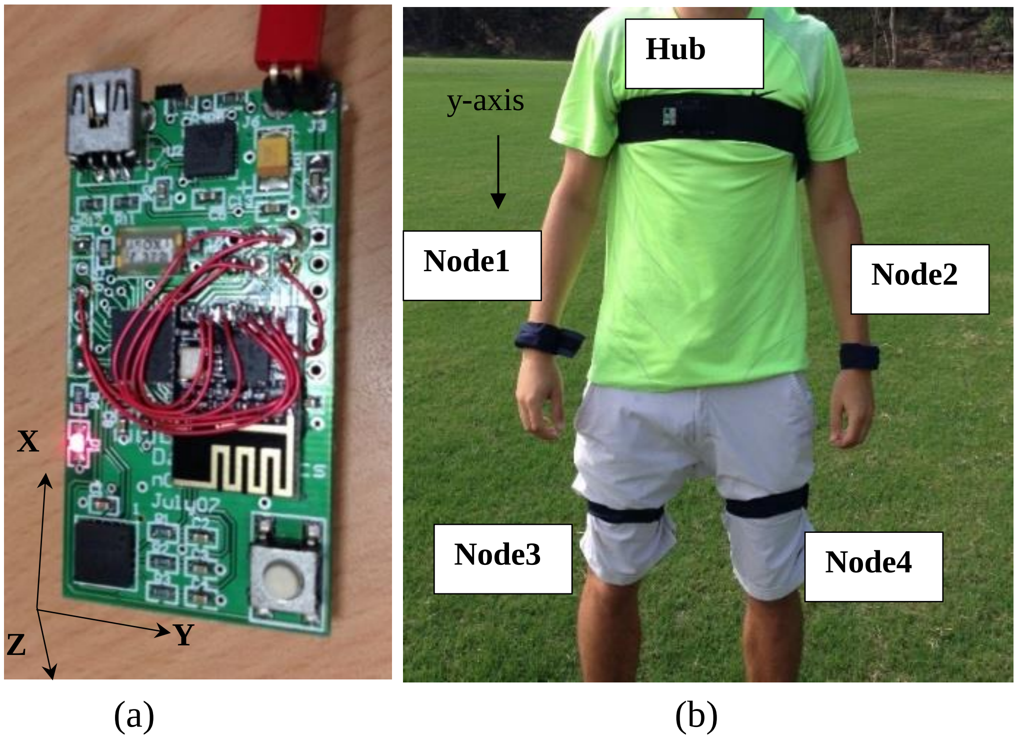


Figure 4-2 (a) Hardware structure of the wireless sensor node. The meander line antenna is clearly evident. (b) Position of wireless nodes on the human body.

This section describes the operational steps of three algorithms that were employed in tests using the system shown in Figure 4-2. All nodes were configured to record acceleration data and to transmit those data to the hub node whenever they receive a data request from the hub. As shown in Figure 4-3, the hub sends an acceleration sample request to each node at a given time burst, and immediately switches to the receive mode and saves the received data. The transmission address is changed to communicate with the next node. The communication sequence of the hub is repeated starting at Node 1 and ending at Node 4. Each hub node has a set of six data pipes with unique addresses: a data pipe is a logical channel in the physical RF channel. Therefore, a hub node can communicate with up to six wireless nodes. Only one data pipe can receive a packet at a time. Each pipe can have up to a 5-byte configurable address. In the wireless node configuration, data pipe 0 should have a unique address, while data pipes 1-5 can share the four most significant address bytes. For identification purposes, the least significant byte of each pipe must be unique. Figure 4-4 shows the details of how the data pipes are addressed.

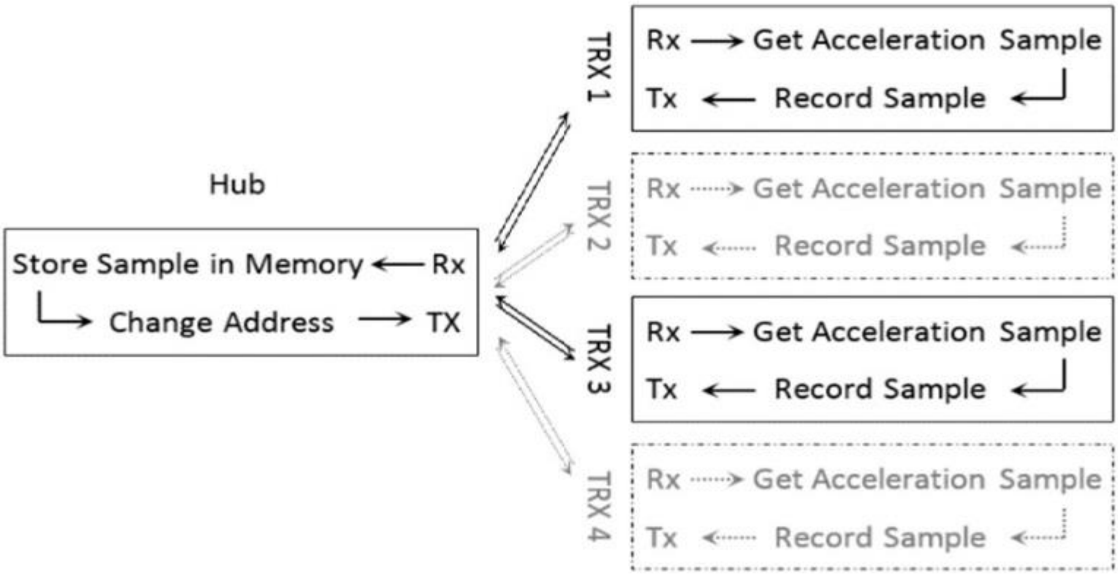
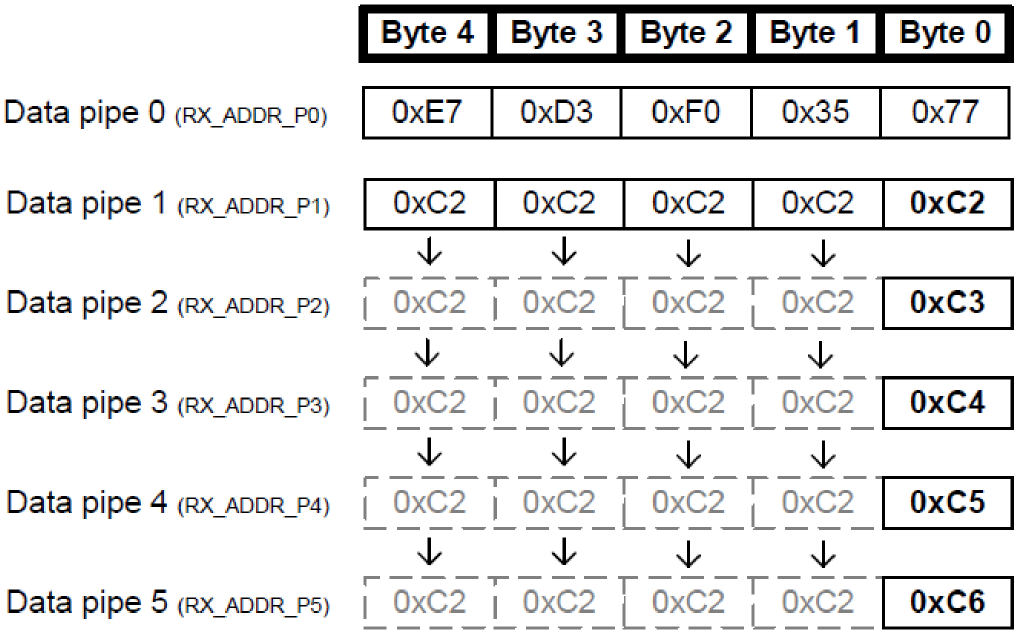

Hub
Store Sample in Memory ← Rx
Change Address → TX
TRX 1
TRX 2
TRX 3
TRX 4
Rx → Get Acceleration Sample
Tx ← Record Sample
Rx → Get Acceleration Sample
Tx ← Record Sample
Rx → Get Acceleration Sample
Tx ← Record Sample
Rx → Get Acceleration Sample
Tx ← Record Sample

Figure 4-3 The network timing multiplexing between the hub and other nodes.



	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Data pipe 0 (RX_ADDR_P0)	0xE7	0xD3	0xF0	0x35	0x77
Data pipe 1 (RX_ADDR_P1)	0xC2	0xC2	0xC2	0xC2	**0xC2**
Data pipe 2 (RX_ADDR_P2)	0xC2	0xC2	0xC2	0xC2	**0xC3**
Data pipe 3 (RX_ADDR_P3)	0xC2	0xC2	0xC2	0xC2	**0xC4**
Data pipe 4 (RX_ADDR_P4)	0xC2	0xC2	0xC2	0xC2	**0xC5**
Data pipe 5 (RX_ADDR_P5)	0xC2	0xC2	0xC2	0xC2	**0xC6**

Figure 4-4 Data pipes 0-5 unique address code example. The four Most Significant Bytes (MSB) of the data pipe address are the same for all pipes, and the Least Significant Byte (LSB) of each data pipe address is changed for identification purposes.

In the receive mode, the hub copies the logical transmitter address indicated in the received packet and uses it as a transmission address to send requests and information to the original sender. This is used when applying the dynamic timing multiplexing algorithm to allow each transceiver unit to have a unique communication path with the hub. Figure 4-5 shows the link configuration of a hub and six wireless nodes with respect to their unique addresses.

The reason for the two-way communication (represented by the hub communication request) is that each node has its own clock system and if they are programmed to transmit at different times, eventually their clocks will drift apart. This will result in timing inaccuracy and data loss. The transceiver capability of each node allows it to transmit data and switch back to the receiving mode for reception at the defined sampling rate. When the hub node sends a sample request, a counter is incremented (location counter) and logged in the hub memory. This counter helps identify the sample number of each acceleration value. The locations at which samples were lost for each node can be identified. In all trials the sample rate was set to 100 samples/s.

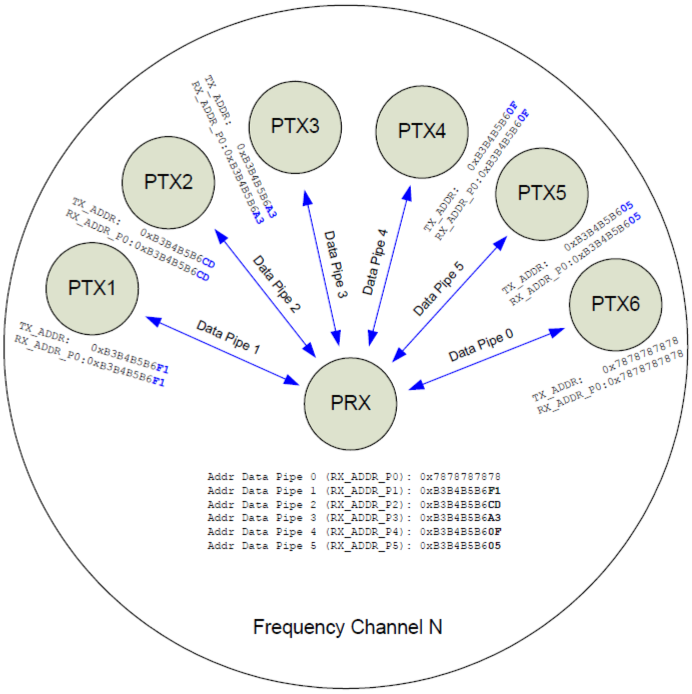


Figure 4-5 A hub using six channels for the communications sequence with six different transceivers.

4.4 Standard TDM algorithm

Before carrying outdoor experiments, the network was tested indoors to validate the standard time multiplexing algorithm. The hub was set to transmit 1000 requests for each node by using 10ms length timing burst. This implies a test time of 10s (resulted from multiplying the 1000 samples by 10ms timing burst), which is more than the time required for following running tests. As expected, there were neither data losses nor shift in the timing sequence of the hub. Figure 4-6 shows the sequence in which acceleration samples were received and saved in the hub memory along with its sample number and node number for 10 consecutive samples. It shows that data coming from the nodes were always logged according to the number of the node (first, data of node 1, next, data of node 2 and so on).

The 10ms timing burst was selected based on timing measurements. This included the total time of data on air (ToA) for both transmitting and receiving operations, sensor switching time from transmitting to receiving modes and the timing requirements for the microcontroller communications process with the transceiver chip, memory and accelerometer sensor. If the hub is communicating with only one node, this will result in 100 samples per second for that node. However, since the hub in the network is controlling four nodes then each node will provide 25 samples per second (the accumulated number of samples for the four nodes is 100). Therefore, the calculated time difference between the samples is 40ms for each node, given that the swing time in running for normal humans is around 400ms. The result will be 10 samples in a swing which is not sufficient to capture fast movement. A simple solution is given by logging the acceleration from only two nodes (Node 1 & Node 3) or (Node 2 & Node 4). In running, these two nodes move opposite to each other with an alternating line of site with the chest. Since running is a periodic movement, it can be assumed that the acceleration data for the first pair of nodes are the mirror of the second pair. This technique duplicates the number of samples (that is, 50 samples per second per node for the 100Hz sampling), and provides more details for the acceleration pattern which helps in gesture identification.

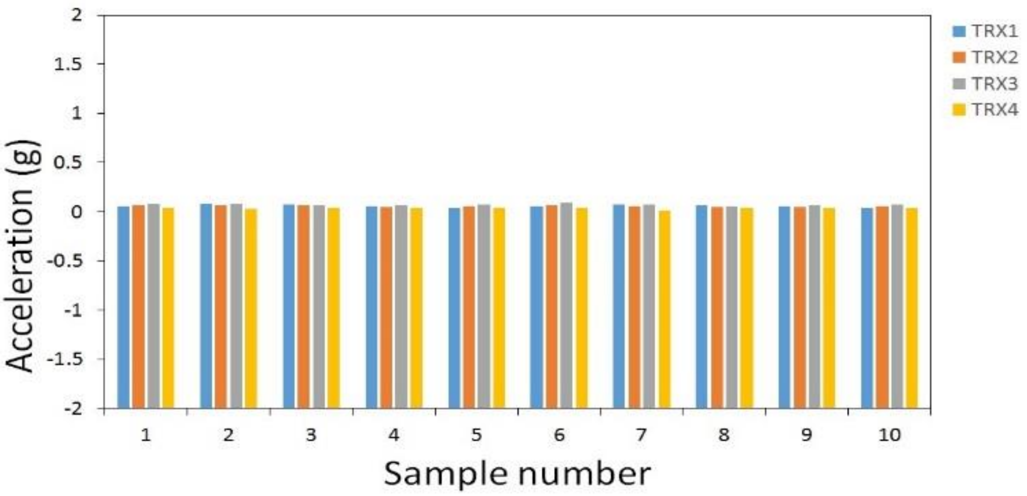
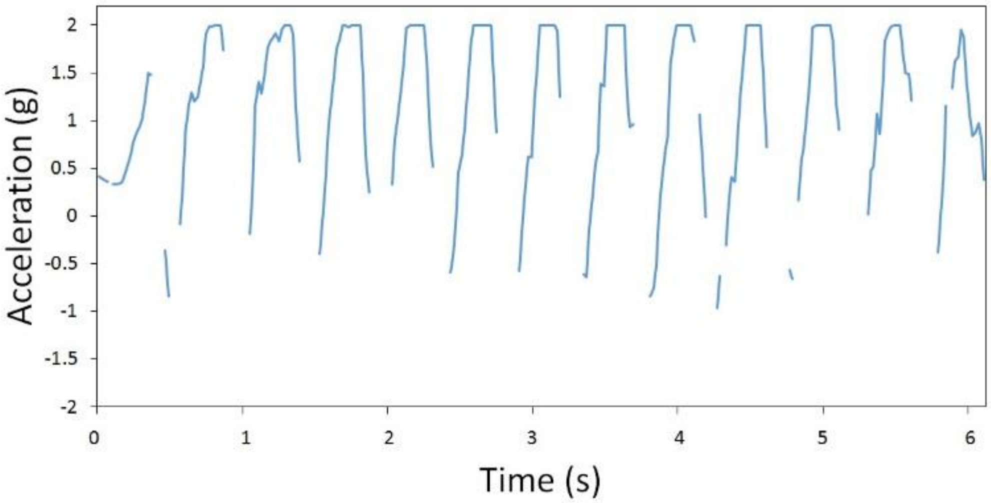
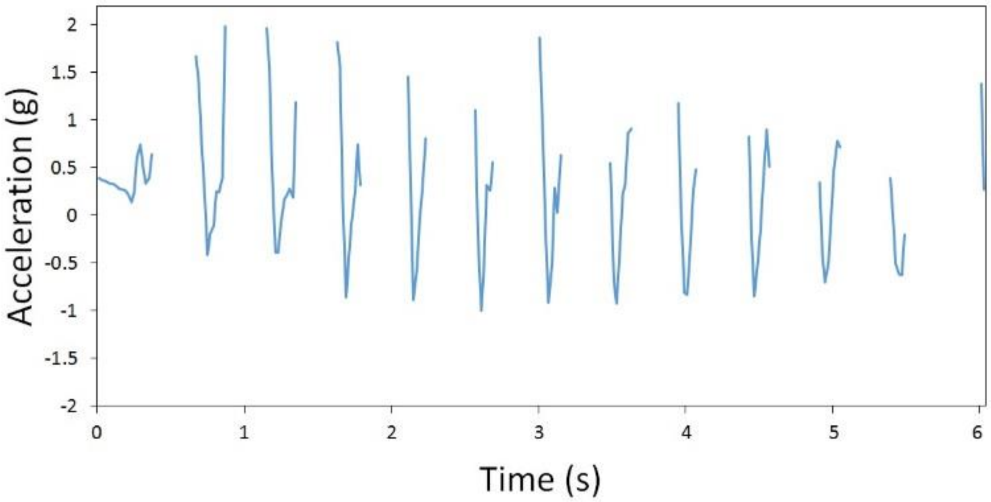


Figure 4-6 The hub order sequence of received acceleration samples for Node 1, 2, 3 and 4 in laboratory, the nodes transmission are shown in different colours in each sequence. These are the transmissions received over a total of 0.4s.

To increase the acceleration resolution during the outdoor tests, the standard time division algorithm was modified to consider opposite nodes and the test was repeated twice for the other two nodes. Figure 4-7 shows the acceleration samples recorded and lost (missed) along with the location counter for Node 1 and Node 3 A pattern of locations for reliable data communication was deduced from the acceleration values received by the hub, as shown in Figure 4-7 (a) and (b). For the wrist locations, the samples are delivered to the hub when the y-axis acceleration is high especially around a maximum value of 2g and data are lost when acceleration values decrease. On the contrary for the thigh locations, samples are delivered to the hub when the y-axis acceleration is low especially around a minimum value of -0.7g and data are lost when acceleration values increase. These patterns show the high reliability time slots for successful communication with the hub node for the wrist and thigh during running. The acceleration values can be used to determine the best locations for reliable data communication, as they are interrelated with the tilt angles (see Figure 3-6). The empty space between acceleration points represent the location at which data were lost in either direction (data request from the hub was not received or the acceleration sample from the node was not delivered).



(a)



(b)

Figure 4-7 The recorded y-axis acceleration data (g) at the hub for participant A of (a) Node 1- right wrist (b) Node 3 - right thigh. The missing data occurs when the communications path fails. Note that the accelerometer has a maximum range of +/- 2g.

The average running cycle time and the average link availability within one running cycle for three runners' wrist position are shown in Table 4-2. In all the cases, the communication between the hub and the node mounted on the wrist was reliable, more than 60% of the average running cycle (which means that nearly 40% of the data is lost when adopting a standard TDM algorithm). It also shows that different running patterns (represented by the wrist cycle time) for the runners affect the wireless link reliability.

Table 4-2 Participant pool and average successful data transmission rate during one running cycle (for wrist position).

Participant ID	Average Cycle Time (ms)	Link availability / Cycle
G	460	62%
H	440	64%
I	500	61%

The acceleration data for the right wrist and thigh (Node 1 and Node 3), left wrist and thigh (Node 2 and Node 4) of participant G were collected and merged in Figure 4-8 to fully illustrate the body sensor network readings during running. It shows two running cycles for right and left body limbs of in one second, and it can be calculated that the cycle for the left/right wrist and thigh are alternating with an average cycle time of around 450ms (which is very close to the time measured in Table 4-1). Furthermore, if the network is programmed to send the information at the time slots that ensure reliable communication between nodes, the link capacity will be optimized and the data delivery will be ensured.

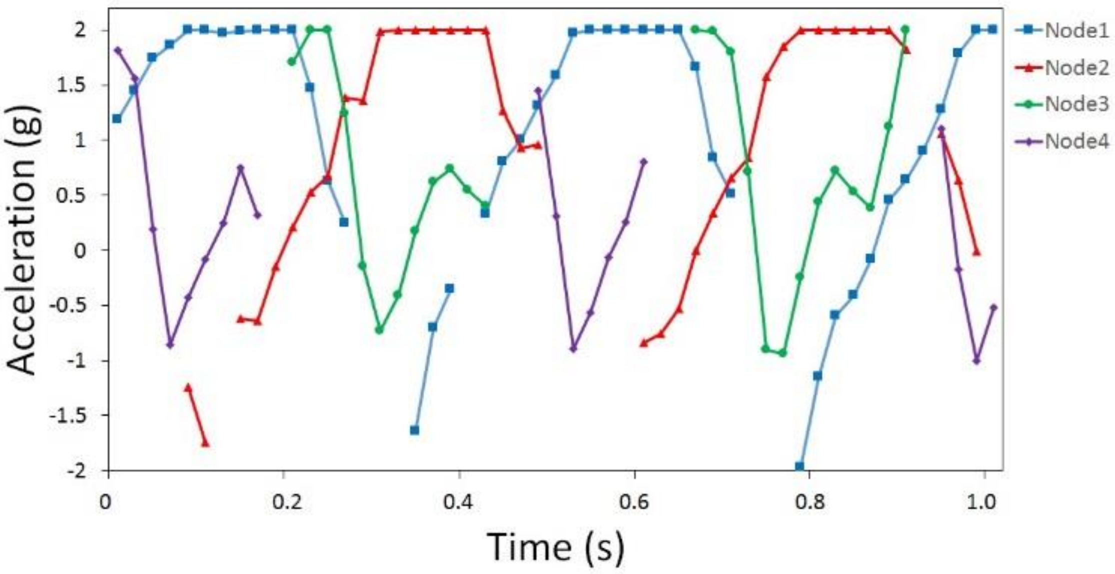


Figure 4-8 The received acceleration data (g) at the hub for all nodes during running.

Continuing with the description of the tests listed on Section 4-1, another experiment was conducted on the five wireless transceiver nodes shown in Figure 4-2. The network was configured to run a typical time division multiplexing algorithm, with the auto acknowledgement feature enabled. The data is retransmitted after a programmable delay of (250µs) and the process continues for up to 15 retransmissions until successful delivery or the maximum number of retransmissions was reached.

The transmitters use different time slots to send their data to the receiver in periodic manner and consecutive sequence. The time difference between each transmission was calculated to allow four data transmissions within the defined sampling rate, which is more than the time required to transmit and acknowledge the data packet. The unique transmitter address allows the hub to identify the origin of the packet. Table 4-3 shows the total traffic generated, the throughput and link reliability for each transmitter node. From the Table, it can be seen that the generated traffic is much bigger than the node throughput. This traffic was due to the retransmission attempts of each transmitter and resulted in a high aggregated body network reliability of 95%. Nevertheless, it reduces the link capacity and restricts the number of transmissions, nodes and information size in large networks. Thus, channel efficiency is low when the link reliability is important. (The channel efficiency is defined by the utilization of wireless transmission channel with respect to its data rate). In this scenario, knowing the accumulated network traffic is 91.22kbps and for 2Mbps channel data rate, the network total channel efficiency is 4.5% (that means the channel is free for 95.5% of the time). Therefore, the large amount of traffic generated would only affect the network life time and have little effect on the channel capacity, represented by the additional retransmissions power consumed in favor of robustness.

Table 4-3 Wireless link evaluation for each node during running shows a high reliability (>93%) for normal time division multiplexing with the AK feature enabled.

Node	Total traffic kbps	Throughput kbps	Reliability
TX1	23.81	1.55	97%
TX2	23.79	1.53	96%
TX3	19.89	1.51	94%
TX4	23.73	1.49	93%

4.5 Self-calibrating algorithm

Different humans have different running styles and speed. As a result, they would generate different running cycle lengths and will have different reliable points/window to establish communication (Chapter 3, Section 3.8.1). This would require different program settings for each runner to make sure the data is sent at a specific timing event. A self-calibrating algorithm was designed to ensure a reliable communication between the hub and the nodes by finding the best transmission points for each node and for different runners. There are two important parameters in determining the best communication time windows based on the acceleration value (actual reading of the sensor) and its location. Controlling one parameter alone would not result in the best performance. For example, if the hub node requests data only at specific locations, then it would be unknown if these locations lie within the pattern where consistent communication exist for that specific node (the results will be similar to those obtained when applying the standard TDM algorithm explained in Section 4.4 with longer time bursts). Requesting the nodes to send their data at certain acceleration values will force the nodes to transmit whenever that condition is true and it may result in overlapped and multi-transmission in one running cycle. Thus, a combination of both is required.

Figure 4-9 shows in-detail the self-calibrating algorithm; the programming code is given in Appendix C. The hub node is responsible for performing this algorithm and controlling the time division multiplexing and address changing (as shown in Figure 4-3). For every 10ms (which is the timing burst), the hub enables the Tx Mode by making a register value (sample=1). This allows it to send a sample data request before switching back to the Rx Mode. After sending the request, the location counter is incremented for every data request for each node (there are four location counters, one for each node). In the Rx Mode, for 10ms the hub keeps monitoring the node reply for the request that already been sent, if the time is runs out and the hub does not receive any reply. It will send another request for the next node and so on. For each acceleration sample received and saved in acc[i], the count counter is incremented (there are four count counters, one for each node), and a condition check is applied on the two counters (location and count). If they are not the same and the count counter holds a value more than 10 (this means there are ten or more samples received consequently before the connection was lost) then an arithmetic operation is performed to find the average acceleration value and the average location of these samples. Adopting the middle point is the best way to ensure that transmission occurs reliably.

The resultant acceleration value is stored in the middle[k] array, while the acceleration location is saved in the point[k] array. Finding the difference between the average location (point[k-1]) and the next average location calculated from the next 10 or more samples, which is obtained through requesting data from the same node (point[k]), will result in finding the running cycle and this is saved in cycle[k]. Reaching a predefined counter value of samples (window =w) the hub calculates the average acceleration value and average cycle and sends it (for given attempts to ensure delivery) to each node to identify best transmission locations. The nodes depend on these values in their transmission and they follow two conditions to ensure that the data is sent only when the calculated acceleration value is reached and the sample numbers are equal or more than the average cycle length (as shown in Figure 4-10).

The application of this algorithm to periodic wrist and thigh movements derived from running activities involves the transmission of one sample burst per limb cycle. In Figure 4-10, one sensor location was tested at a time with the hub. It shows the time slots [Red bars] at which data were received by the hub along with the acceleration samples [Blue line] saved in the nodes attached to the wrist and thigh of participant H. The nodes send data only at the maximum and minimum acceleration values separated by the running cycle length of participant H (440ms) that was obtained in Table 4-2. Figure 4-10 shows that all communication attempts between the hub and wrist/thigh were successful.

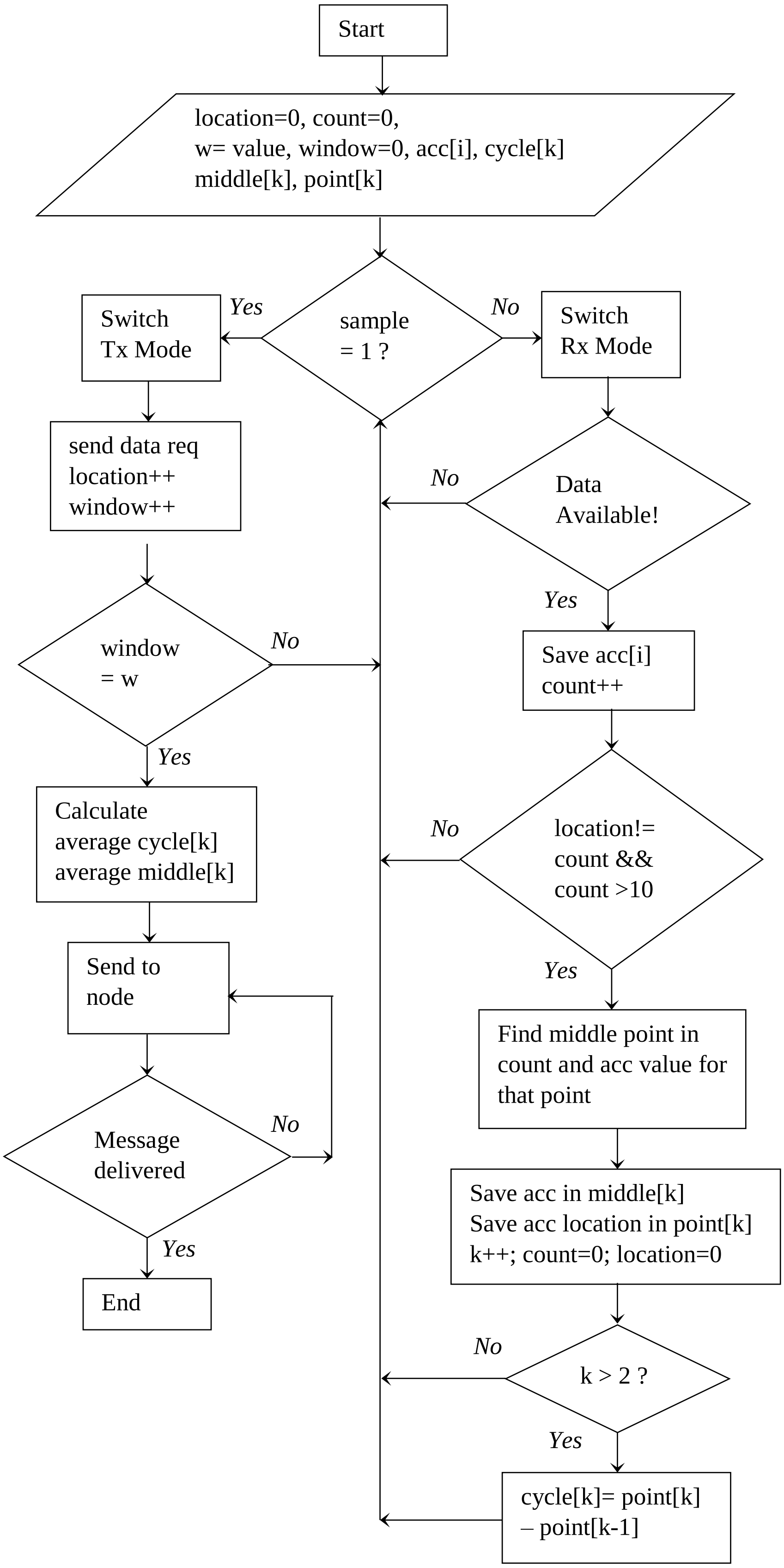
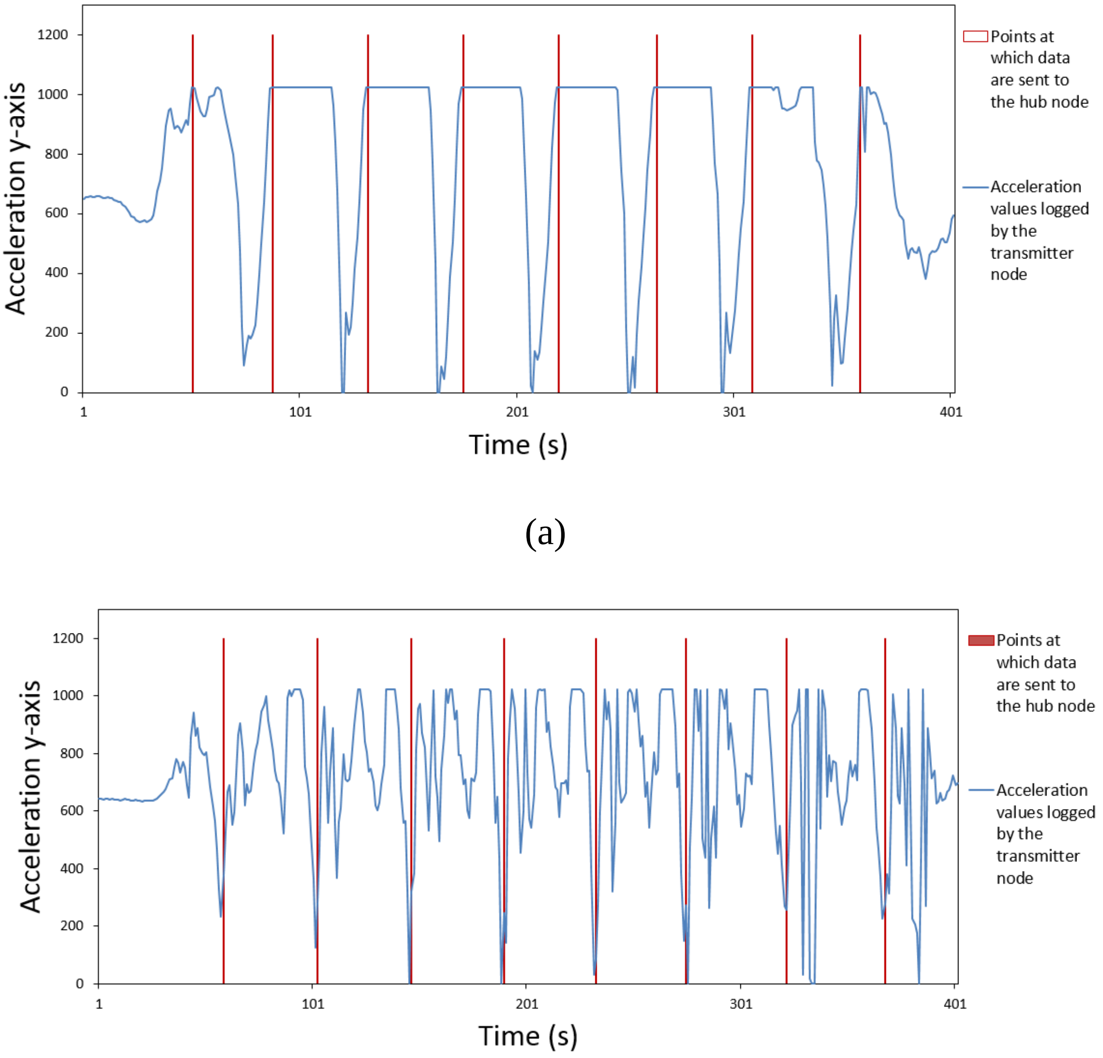


Figure 4-9 Flowchart describing the self-calibrating algorithm performed by the hub. The loop continues for a defined calibration window of samples.



(a)

(b)

Figure 4-10 Acceleration profile in running showing the time slots at which data are received by the hub along with the recorded acceleration samples at each node for (a) wrist (b) thigh.

4.6 Dynamic TDM algorithm

To do these tests, the nodes were calibrated by the hub using the self-calibrating algorithm previously explained, and thus, they learned to send information at the allocated time slots. Figure 4-11 shows the sequence of data transmission from the nodes to the hub performed according to a typical running movement: Node 1 (right wrist), Node 4 (left thigh), Node 2 (left wrist), and finally, Node 3 (right thigh). Table 4-4 shows the transmission attempts for all nodes. This dynamic time division allows nodes to send data depending on the human running gesture, and ensures that only 4 transmissions are allowed in a cycle. Small changes in the limb cycle time would not affect the reliability as the two conditions have to be met before transmission occurs. Nevertheless, two nodes may meet the transmission conditions and attempt to send data at the same time.

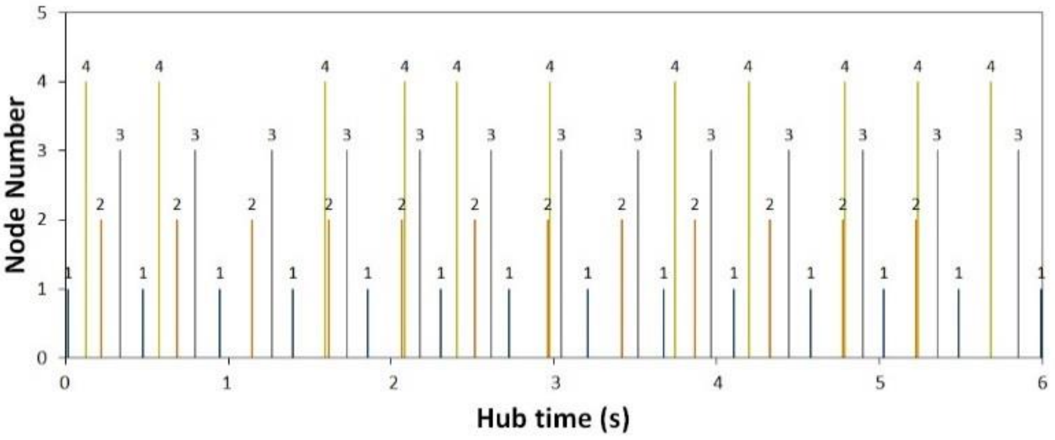


Figure 4-11 Timing diagram showing the time slots at which data is received by the hub from all four nodes. Note that the initial transmission sequence is 1-4-2-3 but near the end the sequence is 1-4-3, where slot 2 is missing.

Table 4-4 Network performance demonstrated by successful wireless transmission using the dynamic time division for all four nodes of participant H. In Figure 4-11 it is clear that there is some timing conflict between 2 and 4 which results in the lower reliability for these two nodes.

Node Number	Average cycle time for each node in (ms)	Aggregated link reliability
1	450	98%
2	450	86%
3	460	95%
4	470	81%

To prevent packet collision, a carrier sensing algorithm was adopted at each node to sense the transmission frequency band before sending its information. The node checks the occupancy of transmission frequencies and, if a signal is detected, waits for 10ms (which is the calculated time burst) and then monitors for a signal again before using the medium. Unsuccessful transmission attempts for Node 4 (near t=1s) and Node 2 (near t=6s) shown in Figure 4-11 can be explained as follows; after the node waiting time is finished, the conditions are not true anymore and the nodes have missed their transmission windows. This issue only appeared when there are multiple sensor nodes in the network simultaneously talking to the hub.

The results in Table 4-4 shows the average limb cycle time (ms) for all nodes attached to participant H for successful transmission only. The limb cycle time is very close with small differences due to the 10ms time burst resolution. It also shows an aggregated network reliability of 90% achieved using dynamic transmission. The 10% losses are the result of transmission overlap between two nodes leading to an unsuccessful transmission attempt. The Table shows that losses resulted from Nodes 1 & 3 (right wrist and thigh) are lower than the losses from Nodes 2 & 4 (left wrist and thigh). This is because Nodes 4 & 2 attempted to communicate with the hub when Nodes 1 & 3 were transmitting their data.

4.7 Gesture transmission algorithm

The control software of the nodes was modified to store the complete acceleration cycle and send the corresponding data when the link was available. Thus, all information was stored during the movement cycle and delivered successfully at these predictable reliable time windows for achieving reliable communication. As running cycle lengths vary for the same runner depending on his/her limb movements, the length of a running cycle was determined from the acceleration samples. Figure 4-12 shows the time corresponding to 10 consecutive wrist cycles of the participant J. The measured cycle length changes slightly (in the range of 400ms-500ms). The average time was 460ms; which means that on average the nodes are required to send 23 acceleration samples from each wrist every cycle (sampled every 20ms). Each acceleration sample is 2-bytes long (excluding the transmission overhead) and requires 13.5mA to transmit in the 2.45GHz frequency channel. The 240mA battery capacity gives a node lifetime of around three hours for continuous transmission (Chapter 3, Section 3-2). The network life time can be increased significantly by selecting one burst transmission point every cycle while forcing additional sleep time for the radio transmitter to avoid unnecessary transmissions.

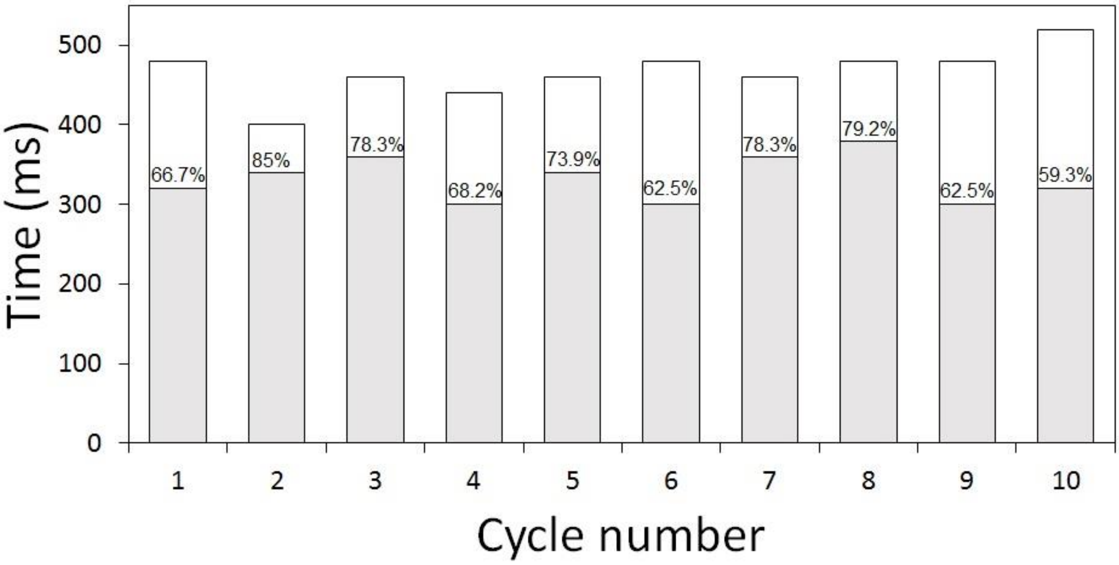


Figure 4-12 Classical continuous transmission with 50Hz sampling rate. The time of ten consecutive running cycles (y axis) for a novice runner and successful information delivery in each cycle (shaded) for 10 running cycles using the gesture algorithm.

During these trails, the sensor node data were stored and sent continuously to the Hub. By comparing the stored data at the node and the hub, the link reliability was calculated. The average reliability was 70% so that 30% of the data were lost when the connection fails at wrist position. Other tests for a different test subject showed higher data losses (see Section 4.4).

To avoid data losses and collisions, a gesture transmission method was adopted. Although data were continuously stored, transmissions were only done when a specific acceleration condition was met. This condition defines the beginning and end of a reliable transmission window and their locations within the running cycle. This varies between different runners (as shown in Section 4.5). The nodes save all samples locally in a block array and start sending these samples one by one at the appropriate time. Link failures will be minor and do not affect the transmission of other samples for the same node (the block array is received with 1 or 2 samples missing). The time required for sending these samples depends on many parameters, including sample size, array size and data rate. In this configuration, each sample takes around 5ms to be delivered to the Hub, and a total average of 23 samples array will require a maximum time period of 115ms to transfer. Figure 4-13 shows a small part of the acceleration data for two nodes stored at the Hub node. Both Node1 (right wrist) and Node2 (left wrist) store data samples in a block array and forward all their information to the Hub at the right time to ensure successful delivery. This was done by setting a minimum acceleration value condition to trigger the transmission. The Hub node receives data blocks and stores them in memory. The block numbers help to keep the received data organized and classified with the sender identity (blocks with odd numbers are used by Node 1 and even numbers by Node 2). The black bars identify the locations at which wrist nodes forward their data blocks of acceleration samples. For example, at 230ms of the Hub time the first block of information from Node 1 is received. After the same period of time, the first block from Node2 follows, and so on.

During these measurements, more than 1000 samples were sent from Node1 and Node2 during running, and only a few samples were lost at the Hub. Specifically, 1000 out of 1007 samples were received by the Hub from Node1, and 1015 out of 1024 samples were received by the Hub from Node2. These lost samples are spread over more than 50 burst transmissions for both nodes, and therefore, does not affect the analysis on the total received acceleration values. Overall, the link reliability was 99.30% and 98.92%

for the Hub-Node1 and Hub-Node2 connections; this is a significant improvement on the continuous transmission reliabilities.

More sensors can be included in the network reliably by carefully considering their transmission burst locations to overcome multiple transmissions at the same time, and by selecting the acceleration sampling rate with respect to the specific runner cycle length. By reducing the data resolution, there will be less time required to forward the data blocks, thus allowing more time in each running cycle for the other nodes. In our current example, each node requires 115ms to forward their data blocks to the Hub. If two more nodes are added to the network, the total nodes transmission time required will be 460ms, which is exactly the length of the participant's running cycle, which makes the network crowded. Even if the transmission conditions for these additional nodes were defined so that a burst communication occurs exactly after the triggered nodes communication with the Hub, the assumption that the length of each running cycle is exactly 460ms (or any other specific value) is wrong. Therefore, adding a margin of tolerance is essential for the network design and conditions setting.

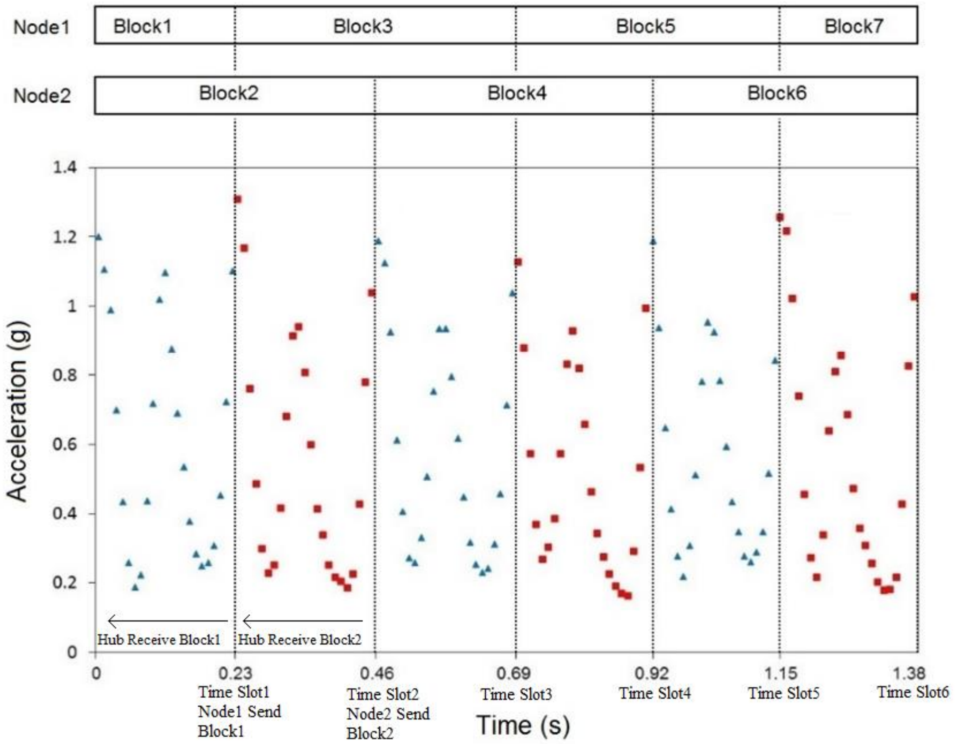


Figure 4-13 The acceleration data points (Node 1 triangles, Node 2 squares) are transmitted in blocks. The transmission is triggered by the detection of an acceleration maximum value [black bars] in both Node1, and Node2.

4.8 Conclusions

The experimental results of the programmed smart algorithms showed minimal or no losses were obtained compared to the traditional TMD technique. The experiments proved that the self -calibrating and gesture burst communication fits the demands of reliable communications with respect to the different running styles of subjects. The network calibration process is dependent on the predefined window of samples gathered for that purpose, during which transmission conditions are set. The smart algorithm design was based on recognizing the limb positions from the instantly captured acceleration samples to predict reliable points. A transmission widow reduction of 75% was achieved, calculated as the 115ms time required to forward data blocks out of the 460ms average swing time. This can increase the sensor life time by a factor of 4, while maintaining a high reliability of more than 98%.

Some of this work was partly published in the author's publications [131, 135-137].

CHAPTER 5

LOCATION TRACKING SMART SENSORS

5.1 Introduction

Prior chapters have addressed the many challenges of body sensor networks, including: high reliability requirements; multipath fading and data losses due to the human body presence; inhomogeneous wireless sensor nodes distribution and change in network topology. An energy-efficient time multiplexing transmission method developed for on-body wireless communication and defined from the human rhythmic movement of running was discussed in Chapter 4. This chapter focuses on the off-body wireless communications between moving human bodies through a sink node. Experiments were conducted to test the sink node coverage at various locations on the body to ensure reliable off-body communications. A neighbour identification algorithm was implemented and validated through experimental results. The nodes adopt the CSMA method to increase their awareness of surrounding neighbouring nodes in dynamic conditions. The final network design employs two frequency channels for on and off-body transmissions, where the processed information are transmitted in a real-time format to the sink node and then presented through an interactive monitoring software that was designed in a MatLab program.

5.2 Sink node coverage

There is no obligation to select specific locations on the human body to install wireless sensor nodes, as this depends on the type of application and the signals to be monitored. For a heart rate monitoring application example, it is logical to place the sensor node at a location closer to the heart area [91]. In most applications, these on-body sensors, and other (nearby) sensors that are spread in the environment, shape the network topology and limit its ability to function reliably over a definite coverage area. The 2.45GHz wireless body network described in Chapter 4 - Section 4.3 was used for obtaining the results presented in this chapter. In this case, the hub acted as the sink node, and thus, it controlled the transmissions of other nodes attached to the body [129, 136]. Additional details with regards to the node hardware design and functionality were presented in Chapter 3, Section 3.2. All measurements were conducted in an outdoor environment and the nodes were configured to transmit and receive data at different power levels and body locations to identify coverage area. Figure 5-1 shows the sink nodes attached to participant J’s chest, shoulder and head. It also shows another node located at the sport field ground that acts as an access point and transfers the captured information in real-time to a laptop through the SPI peripherals where the results are displayed. During the coverage tests, the Tx node was programmed to transmit information at a fixed rate of 100Hz while changing the node locations and transmitting power. The Rx node recorded the information and calculated the percentage of packets lost for every second of transmission. These percentages were used to measure the link reliability between the transmitting and receiving nodes. During these tests, the transmitting node was attached to a human subject who was asked to walk past the Rx node, which was either attached to another participant body or fixed on the ground. As shown in Figure 5-2, the distance between the nodes ranged from a negative value of -6m when the two participants were facing each other, to a positive value of +6m when they were back to back, being zero when the participants were in-line with each other.

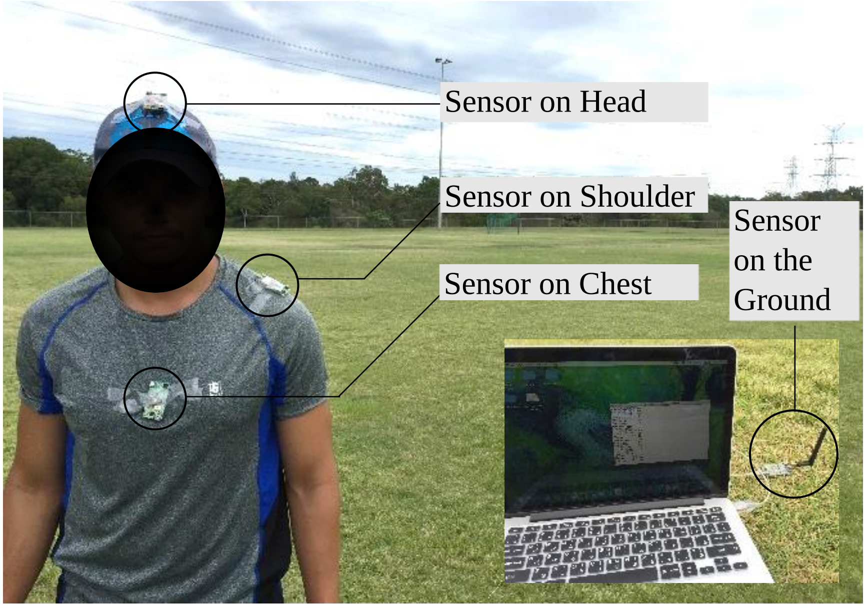
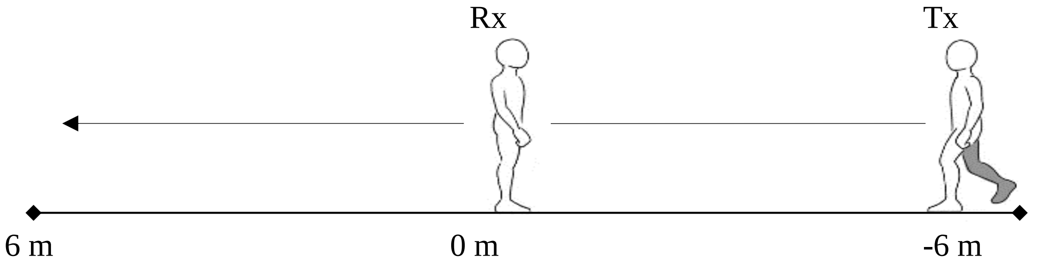


Figure 5-1 Sink nodes location on a human’s participant and ground sink node located in a sport field.



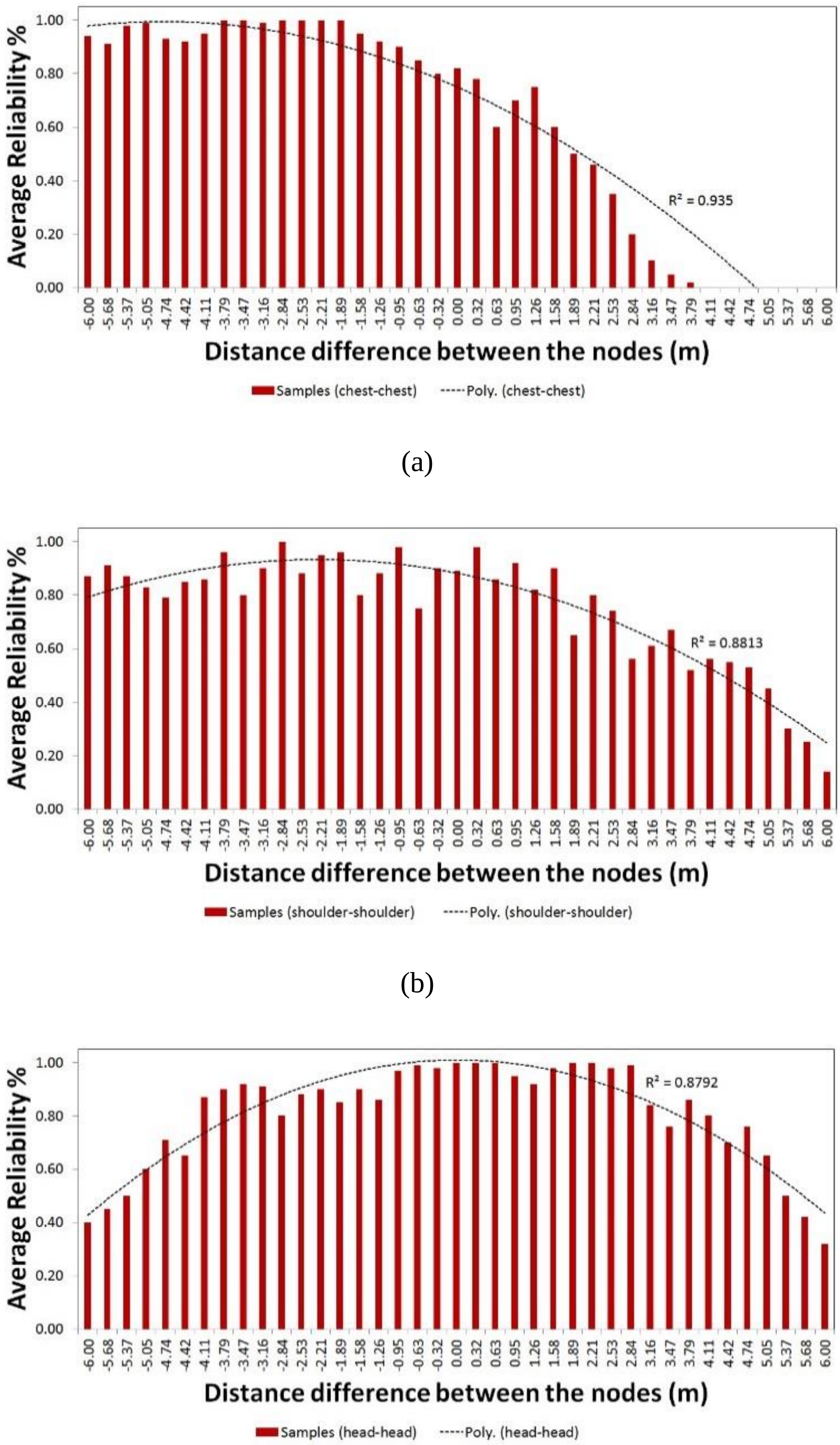
(b)

Figure 5-2 Experiment walk path and difference in distance between the transmitting and receiving nodes.

5.3 Human-human communication

Both the transmitter and receiver nodes were attached to human bodies to validate the off-body connection reliability with respect to the nodes coverage as shown in Figure 5-3. A reliability ratio of “1” indicates no losses in the transmission and all packets were successfully received, while reliability average ratio of “0” indicates 100% data loss and no packets have been received.

The results of placing the nodes on the chest are shown in Figure 5-3 (a). A connection reliability ratio of 90-100% was recorded when the two participants face each other, however this ratio falls to around 75% when the distance between nodes becomes zero. The reliability ratio continues to fall until reaching a value lower than 10% under NLoS conditions when the subjects are back to back at a distance of around 4m. Figure 5-3 (b) shows the reliability ratios recorded when the nodes where attached to participants' shoulders. Similarly, to the chest location, the reliability ratio falls when passing the zero point to reach a value of around 35% at a distance of 6 meters. Placing the sink nodes on the head shows a bell shape results where the reliability ratio is almost evenly distributed around the zero-point distance, with high reliability values closer to 100% when the distance difference between the sink nodes is minimal. The reliability values fall to around 40% when there is a difference of 6 meters as shown in Figure 5-3 (c). All of the recorded samples in Figure 5-3 were fit to polynomial functions of a 2nd order to show the correlation between the reliability and distance between sink nodes.



(a)

(b)

(c)

Figure 5-3 The average samples connection reliability and polynomial curve fit of a 2nd order with respect to the difference in distance between the sink node when attached to the participants (a) chest (b) shoulder (c) head.

The coefficient of determination (R2) showed a good fit for all modules of Figure 5-3, with values of 0.93, 0.88 and 0.87 for chest to chest, shoulder to shoulder and head to head connection modules, respectively. By comparing the curve fittings shown in Figure 5-4, the sink nodes communication placed on the heads shows the most promising location when the athletes are moving. This is mainly because the average reliability pattern is similar to a normal distribution, which ensures normal operation of wireless transmission within the coverage area even when the nodes are not facing each other. Sink nodes placed on the chest or shoulders can result in wider coverage when the players are facing each other. The node coverage can be defined as a function of the wireless reliability, and therefore different coverage can be obtained for different reliability. Choosing a link reliability of 80% or higher results in a node coverage of around 5 meters for chest locations, 7.5 meters for the shoulder location, and 8 for head location.

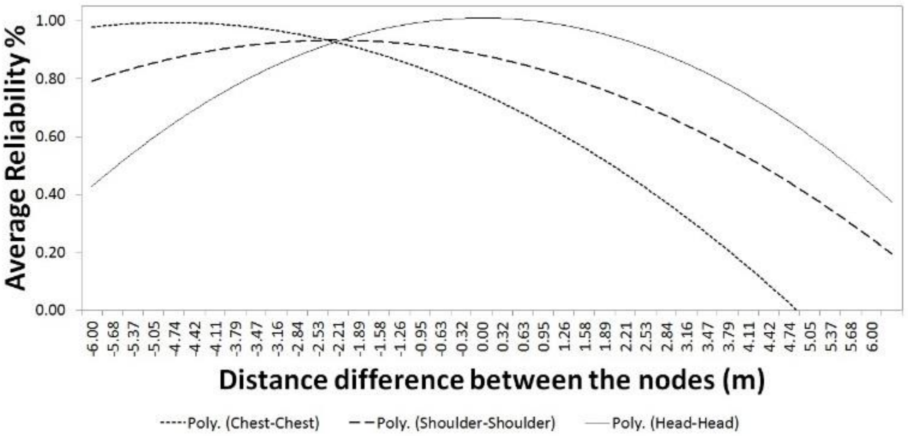


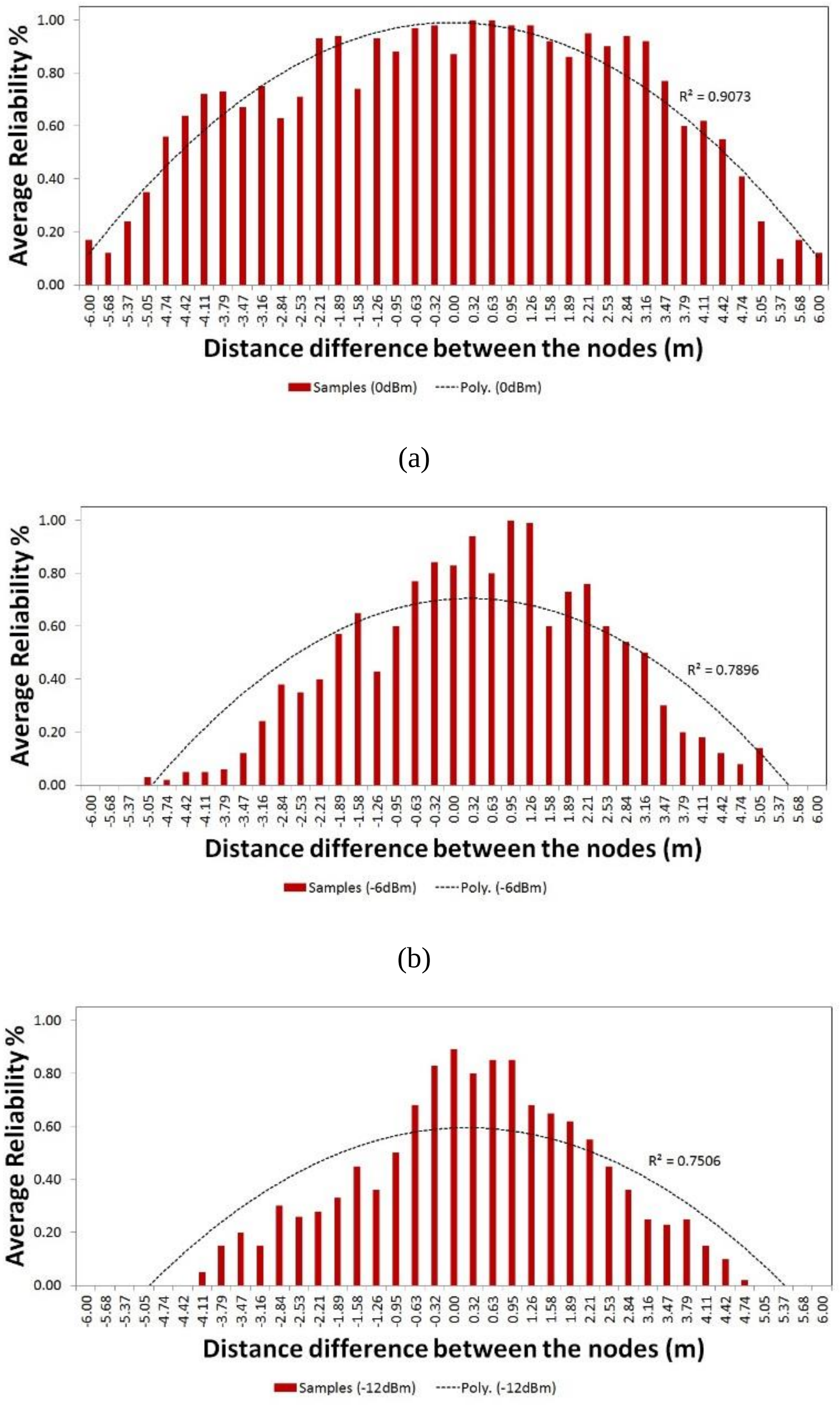
Figure 5-4 A comparison for the polynomial curve fit of a 2nd order with respect to the difference in distance between all locations of the sink node.

5.4 Human-environment communication

Different transmission power levels were applied when the Tx node was attached to the participant head and the Rx node was placed on the ground to verify the off-body connection reliability as shown in Figure 5-5. Similar to the previous results, polynomial functions of a 2nd order were used as curve matching lines to approximate the real data points. Three tests were conducted and the Tx node was programmed to work at different power modes; 0dBm, -6dBm and -12dBm, respectively. Because of the height difference between the transmitting and receiving ground node (this was negligible value in the human to human communication tests), the Rx node with an external antenna and power amplifier as shown in Figure 5-1 was used to establish and test the connection for different power levels.

All plots show an approximate even distribution around the 0 point of the average reliability pattern when the Tx node was attached to the head. The size of change in link reliability and distance with respect to the power levels is shown in Figure 5-5 (a, b & c). A comparison between the trend curves obtained from the reliability ratios previously calculated can be seen in Figure 5-6. It shows that a transmitter power of 0dBm provided a wider node coverage compared to utilizing a -12dBm power. Moreover, the link reliability was effected greatly when transmitting at low powers. For example, -12dBm power level resulted in 50% maximum reliability around zero-point distance compared to a 95% maximum reliability when transmitting at higher power of 0dBm. This means that, in the case of transmitting at -12dBm, there is 50% chance of losing the information even when the player is standing at the point where the Rx node is placed.

The number of nodes required to establish a network covering a whole sport field depends on the transmission power of these nodes (i.e. node coverage). There are many advantages of transmitting at lower power levels; it can save power consumption of the nodes and provide a longer network life time, and increase the resolution in detecting player position on the field. However, more nodes are required when the coverage area is reduced. Furthermore, the network reliability is reduced at lower power levels and this can decrease the probability of data delivery.



(a)

(b)

(c)

Figure 5-5 The average samples connection reliability and polynomial curve fit of a 2nd order with respect to the difference in distance between the ground sink node and sink node attached to a participant’s head while transmitting at maximum power of (a) 0dBm (b) -6dBm (c) -12dBm.

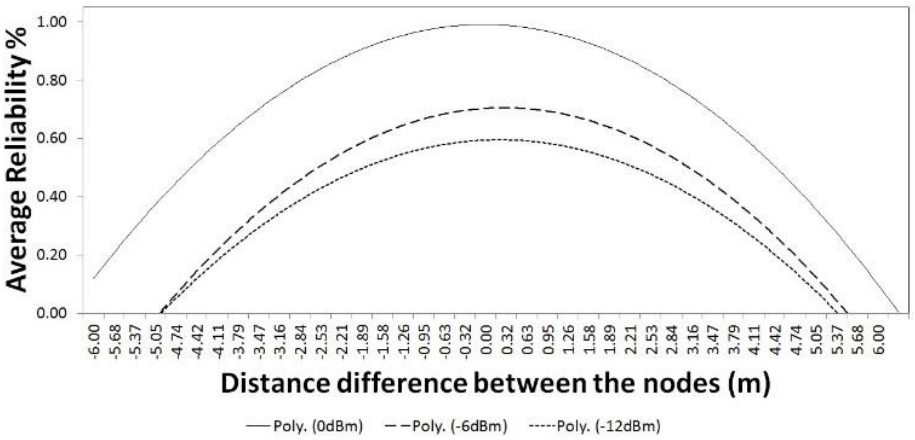


Figure 5-6 A comparison for the polynomial curve fit of a 2nd order with respect to the difference in distance between all transmissions power of the sink node.

5.5 Neighbour identification in changing environment

Tracking human motion for rehabilitation process requires continuous monitoring of biological and physical parameters. Thanks to the promising features of Body Sensor Networks (BSNs), an increasing number of researchers and industry providers have started to utilize this technology to collect such information for different body parts at different times [10, 49]. In this section, all experiments involved placing the nodes on the head position of five participants. The nodes were programmed to send information with a transmission power of 0dBm and a sample rate of 100Hz. This was based on the results of the previous tests which involved selection of the best location of a sink node (as shown in Sections 5.3 and 5.4).

To simulate the movements done by the participants during a training session or actual game on the sport field, the five participants were positioned over a small area in an outdoor sports field and were asked to move around to preconfigured locations as shown in Figure 5-7. The movement of these players results in constant change in their locations with respect to each other, and therefore, the coverage area of new nodes will intersect with some but not others. These dynamic changes result in an adjusted neighbour list that can cause interference problems. To avoid co-channel interference that happens when nodes transmit at the same time, especially when they become within each other's coverage after a lost connection, a self-awareness algorithm was designed to identify the new and old neighbours for each node, such as that:

- Each sink node is a smart node that can identify neighbouring nodes in the changing environment. If a sink node goes off the network grid, it will know that other nodes are out of wireless range;
- The nodes use the CSMA (carrier sensing multiple access) technique, where each node has a different wait time set up in their initial receive mode before they can switch to transmit their information;
- The initial receive time for the transceiver nodes are set to be the multipliers of the time required for a single transmission to avoid network conjunction;
- When transmitting, the nodes send out their new neighbour lists and time of existence of all neighbours, as well as their unique node number to identify where the transmission come from;
- When listening, the nodes identify where packets come from and save this information. They subtract the preconfigured waiting period of received nodes from their initial waiting period before transmitting, to modify and reduce the total waiting time of all nodes;
- As the nodes are moving continuously, neighbours are changing for each node. Every change defines a new overall coverage and the number of nodes within range. For every change, the nodes record the times and node information.
- The nodes were programmed to spend 100ms to complete carriers sensing and transmitting cycle. This period can increase when increasing the sink nodes in the network, as there may be more carriers to be monitored.

The participants were asked to move in a specific order to three positions, see Figure 5-7. For instance, participant with sink Node3 started within the coverage range of sink Nodes1 & 2, then went to a location that was out of the coverage area of the rest of the nodes, and finally moved to the right part of the field in order to join Node4. While the participant with sink Node1 stayed in his position during the whole test, Nodes2 & 3 moved away from his coverage. During this test, all nodes were configured with the neighbour's identification algorithm. Table 5-1 shows neighbouring node lists for each sink node and position. These results confirm the functionality of the algorithm, as it shows the expected neighbours for each scenario. In the same way, when comparing the movement of sink Node4 from Figure 5-6 with the neighbour lists in Table 5-1, shows exactly the anticipated change in the neighbour list as the nodes dynamically change their locations.

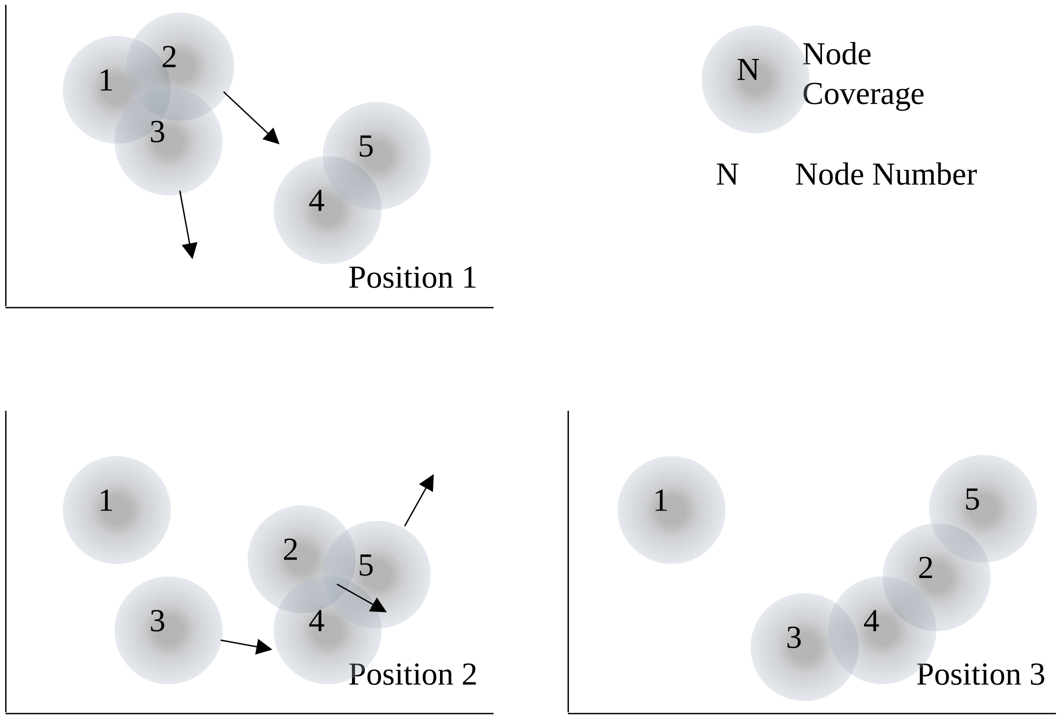


Figure 5-7 Top plan view of a sport field shows the dynamic movements of all five nodes for each stage with respect to each other. Subjects move in sequence from position 1 to position 2 and position 3.

Table 5-1 Updated list of neighbours identified in all nodes at the three positions.

No.	Neighbours at position 1	Neighbours at position 2	Neighbours at position 3
Node 1	2,3	0	0
Node 2	1,3	4,5	4,5
Node 3	1,2	0	4
Node 4	5	2,5	2,3
Node 5	4	2,4	2

Figure 5-8 shows the total time of neighbouring contact for all sink nodes during the participant movement through all of the predefined positions in the dynamic neighbour identification experiment. Throughout the test time, results have showed that Node2 has the highest number of neighbours (4) and time of existence of those neighbours (≈8sec.) in its list, while Node1 had the least number of neighbours (2) and time of existence (≈3.5sec.). All of these successful communications between the participants at each stage of the test and the total time they spent within each other coverage, can help the coach to draw the conclusion that the participant with Node2 had more interaction with the other nodes than participant with Node1.

The test also involved the use of a sink node with an external antenna and power amplifier which was placed on the ground to retrieve network information and mark the position of the players on the field. The ground node was connected to a laptop computer to display real-time updates of the neighbours and time of existence in the neighbour list for each node as shown in Figure 5-9. Once a player is within the ground node coverage all information can be retrieved by the ground node. Employing more ground nodes that act as access points will support identifying player position on the field. This can help the coach to differentiate players that are running and playing together from those stand in a fixed position with respect to movement and time. It is not an excluding case. That is, the nodes can utilize the algorithm to identify nearby neighbours even when access points are used.

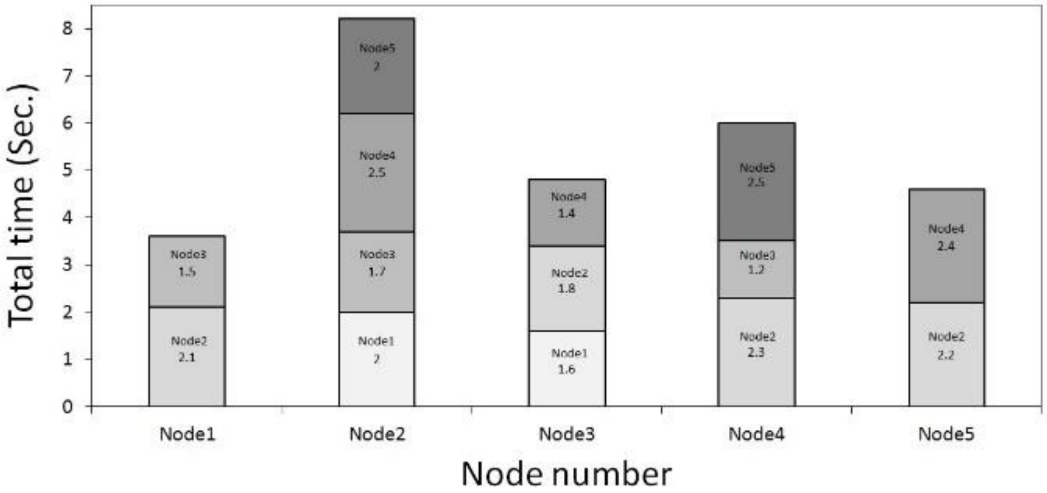
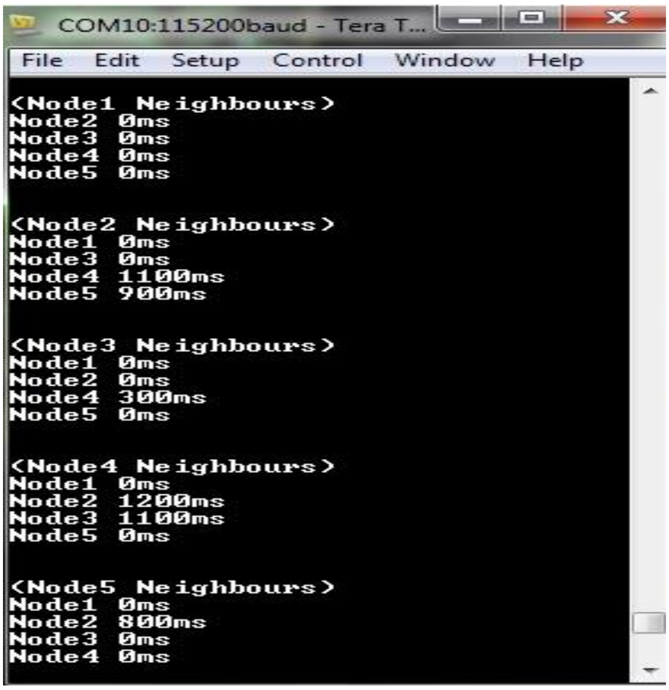


Figure 5-8 The total time in the neighbour list for each node during the dynamic movement identification experiment.



COM10:115200baud - Tera T...
File Edit Setup Control Window Help

```
<Node1 Neighbours>
Node2 0ms
Node3 0ms
Node4 0ms
Node5 0ms

<Node2 Neighbours>
Node1 0ms
Node3 0ms
Node4 1100ms
Node5 900ms

<Node3 Neighbours>
Node1 0ms
Node2 0ms
Node4 300ms
Node5 0ms

<Node4 Neighbours>
Node1 0ms
Node2 1200ms
Node3 1100ms
Node5 0ms

<Node5 Neighbours>
Node1 0ms
Node2 800ms
Node3 0ms
Node4 0ms
```

(b)

Figure 5-9 Screenshot of the serial interface at the ground sink node shows the neighbours and time of existence in the neighbour list for each node.

5.6 Interactive monitoring software design

The test involved the use of a two-tiered network that uses two frequencies to establish on-body and off-body communications. The on-body communication is represented by the central node (Hub) communicating with the two wrist nodes using channel 1 frequency. The Hub node on the chest acts as a repeater that forwards all gathered information to the remote unit through channel 2. Figure 5-10 shows the sensor network layout; these sensor nodes were hardware and software modified accordingly.

When measuring acceleration data, it is important to detect gestures or actions that are dependent on the direction in which the sensor is moved. It was experimentally determined that the best acceleration axis to use in this calculation is that perpendicular to the wrist movements (Chapter 3, Section 3.4).

The time required for sending data blocks depends on many parameters, including sample size, array size and data rate. In this configuration, the time required for the Hub node to control the on-body communication of two nodes exceeds the running cycle period and so a second channel was required for off-body communication, to avoid interference between the transmitted signals.

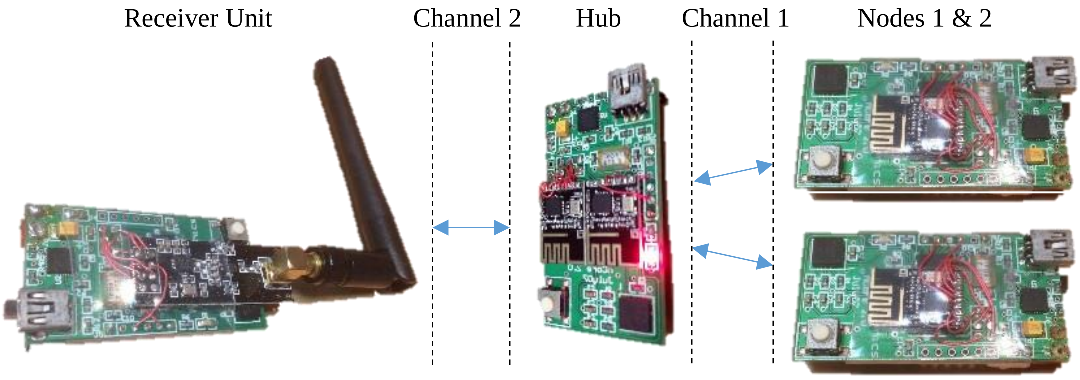


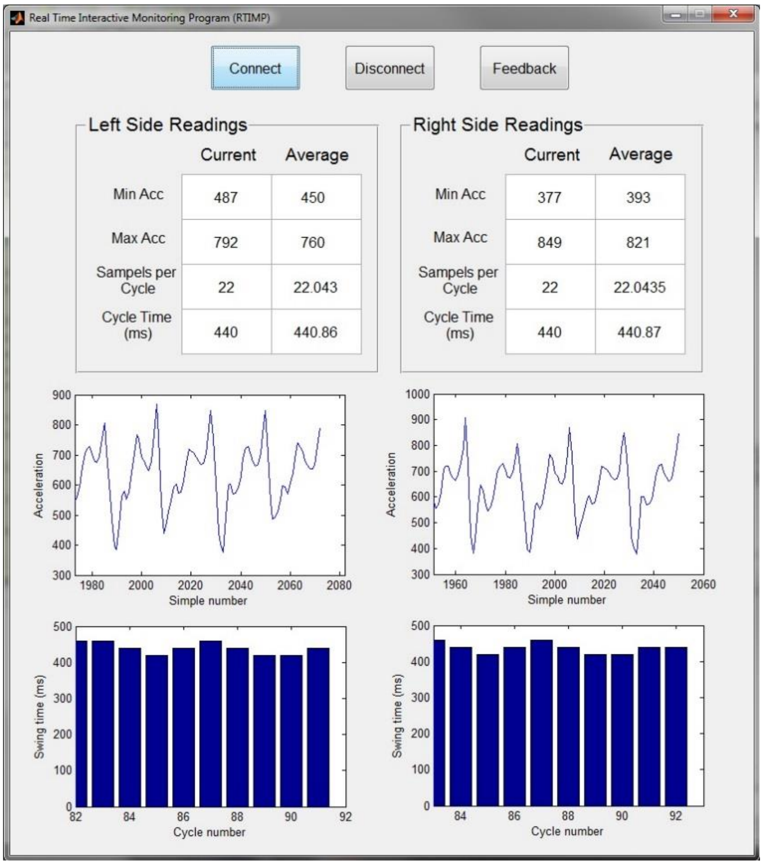
Figure 5-10 Shows the receiver unit is on the ground with a vertical omnidirectional monopole. The hub was located on the chest of each participant and the nodes were located on the left and right wrists.

In competitive sports, the measurements and calculations of the running cycle lengths and symmetry in body performance can provide important information to the athlete. Following the previous work on increasing the link reliability using gesture triggered transmissions, data provided by on-body nodes are stored in blocks and sent once every running cycle (Chapter 4 -Section 4.7).

Data blocks are transferred from Node1 and Node2 through the network tiers to the receiver unit attached to a local computer or Tablet. The later checks the data block number to identify the transmitter (left or right wrist node) and to highlight important features in the athlete performance in each running cycle. A different block number enables the receiver to certify that all samples from the old block were received and new information. Once all information is received, signal analysis and presentation operations are performed. For example, the time duration and samples recorded in each running cycle are used to observe the symmetry between right and left wrist readings (see Figure 5-11). Other relevant parameters, such as maximum and minimum acceleration values, are calculated for each running cycle to show limb movement. The information was plotted to help the coaches instantly identify movement with respect to time and to measure the balance in the running activity.

This real-time interactive monitoring program consists of a single software module that can be run in any MS Windows operating system. It was configured using the MatLab® application support tool (GUI). It is specifically designed to measure the difference between the right and left limb movements of athletes, which could help coaches and professionals to understand the athlete performance, monitor injury recovery and to identify movement restrictions of a limb. The interactive nature of the program allows the receiver to send feedback to the runner in order to change the cycle rate following coach advice (for example, to increase or decrease the cycle rhythm until the desired rate is achieved). All information is collected via the serial communication between the UART interface of the sensor node and the USB connection of the computer device.

To be able to create a serial port object associated with the serial port specified, a serial function was used. If the port does not exist, or if it is in use, a try and catch statement was implemented to allow override of the default error behaviour for a try statements. If any statement in a try block generates an error, the program control goes immediately to the catch block, which contains the error handling statements. This ensures that there is an established serial port connection with the sensor node. The program reads the next line of the serial connection, including the newline characters and uses the white-space character as a delimiter, to separate and store the streamed data. Once the block of data is received and arrays for the specified information have been created, the program runs the calculations on the stored information by using the implemented function, as shown in Appendix D.



Left Side Readings

	Current	Average
Min Acc	487	450
Max Acc	792	760
Sampels per Cycle	22	22.043
Cycle Time (ms)	440	440.86

Right Side Readings

	Current	Average
Min Acc	377	393
Max Acc	849	821
Sampels per Cycle	22	22.0435
Cycle Time (ms)	440	440.87

Figure 5-11 The real-time interactive monitoring program displays the maximum and minimum acceleration values, the raw acceleration data and the variation over time. The interface is highly flexible and can be programmed for many different monitoring applications.

5.7 Conclusions

The wireless connection reliability was measured at different locations on the human body, namely, chest, shoulder and head. These measurements were used to identify best sink node position for human to human/ground communications. A smart adaptation algorithm employed the carrier sensing multiple access (CSMA) technique to detect moving neighbours in the vicinity of sink nodes. The results present an effective operation of the designed algorithm that can provide significant feedback to the coaches during training sessions.

Some of this work was partly published in the author’s publications [138].

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Research summary

The development and use of 2.45GHz wireless inertial sensors in a body network for sport applications was investigated. The designed BSN utilizes various division multiplexing techniques for capturing real-time human running movement and extract important features for athletes and coaches.

Multiple algorithms and codes were developed and tested for the body-centric star shape network at six different places, located on the arms and legs and a central node (hub) on the chest for a number of volunteers during running. The proposed network allows a coordinated approach to gait analysis. The time sequence for communications between the wireless nodes and the hub is set during the first few steps, as runners have different styles and can run at different speeds. Different experiments were conducted on the designed network, based on static and dynamic measurements to monitor the effect of human running movement on the wireless body sensor network and to enable on-body communication with high reliability. This research also discussed the electronics requirements for the wireless nodes that were designed, implemented and tested. The wireless communication settings and the accelerometer sampling rates were thoroughly investigated.

The effect of human position was explored and the received signal level was recorded at 108 static body positions to simulate the human running and walking actions and for six different transmitter locations on the human body. The results showed that the sensor on the wrist provided the most reliable communications with a central node attached to the chest. The receiver angles for all sensor positions were identified as the best LOS positions that can provide the most reliable wireless communications link. This work has been published in [132]. The time division technique and the swing time of runners were used to set the transmission time windows for the sensors distributed on the human body. The accelerometer on each sensor node was used to identify these time windows during the action of running. The synchronization process is carried out by the central node to ensure that the data collected were sampled at the same time for each wireless node. This work has been published in [130].

A reliable measurement sensory tool for capturing the limb rotation and tilt angle from sensor orientation was presented in [133], and a comparison between the continuous and discrete acceleration measurements were made. The angles for stationary measurements were calculated with a 5° accuracy for stationary limbs. The tilt angle was calculated as function of the normalized and mapped gravitational acceleration. This type of measurements can play a significant role in the process of athlete rehabilitation, such as restoring the full range of movement following sprain injuries.

Dynamic measurements were conducted to determine the wireless gait analysis using accelerometer sensors with a sample rate of 100 samples per second. Each sensor recorded the acceleration samples per swing action and sent these samples during the appropriate time window to the central unit, which in turn sends them off the body (locally or to the cloud) for further analysis. The dynamic test results confirmed that installing the sensor on the wrist provides the best among all selected locations for reliable communications. It also showed that the swing time of different runners are strongly correlated with the percentage of successful transmissions. This study [131] reported the analysis of the wireless link performance for different transmitter locations and explored the effect of using the auto acknowledgment protocol implementation for different applications. High aggregated network reliability of 95% was achieved by retransmitting the lost packets to increase the chance of information delivery. The drawback is the network lifetime will become shorter as more power is consumed compared to its normal operation. Measurements of the channel occupancy and traffic were generated to provide high reliability for a time multiplexed-star shape body network were reported in [137].

An investigation of the dynamic time division multiplexing technique coordinated by the human movement to ensure high communication reliability led to the design of a self-calibrating algorithm [135], which was employed to find these time slots and the running cycle of the participants. After calibration, the central unit sends a synchronization pulse every running step and sets a unique transmission time window for each individual node. The time windows are scheduled when there is reliable communications between the hub and the sensor nodes around the body. The results showed that all the transmissions attempts were successful in a node to node communication, and there were few data packet losses (10%) in a multi-node network which can occur when two nodes try to transmit at the same time.

A gesture analysis algorithm was developed [136] and tested in real-time on the collected acceleration samples from the participant running movement for multiple locations on the human body (foot, leg, and arm). This has shown a significant improvement over the wireless channel reliability and efficiency. The test objective was to identify high link quality positions to assist the successful delivery of saved data blocks in the distributed nodes around the body to the central node on the chest. This has been achieved by controlling the transmission time in a dynamic and periodic manner. The technique allows the sensed acceleration data to trigger information broadcast at specific actions, so that the energy cost for multiple transmissions can be minimized and the lifetime of the network can be prolonged.

A two-tiered wireless sensor network was designed and configured for real-time feedback applications. The acceleration readings obtained from the on-body athlete sensors were transmitted off-body throughout the network and processed to be presented in preferable pattern to the involved personals, showing the movement symmetry and stride rate between the right and left body limbs during running cycles. This provides an interactive real-time feedback monitoring system between the coach and the athlete for rapid improvement in player performance during training/rehabilitation sessions. Outdoor tests were made to find the best data sink location on a human body to overcome coverage limitations. A self-awareness algorithm that uses the Carrier Sensing Multiple Access (CSMA) technique was developed to discover neighbouring nodes during dynamic movements in the network.

Overall, the research underpinning this thesis helped to modify and develop a small wireless inertial sensor and software tools which are capable of extracting features of interest and provide instant feedback. Even as the focus of this research was based on running, the algorithms and codes developed could be implemented in other sports. The data was presented in a real-time format on a nearby computer through wireless transmission. However, the growing sector of Internet of Things (IoT), cloud based services and mobile internet ready devices, such as smart phones and tablets, can support the presentation of vast amounts of new real-time data generated by these devices from anywhere in the world. This could be a general goal for future research.

6.2 Future work

Future research might be focused on developing comparable IEEE 802.15.6 channel models, improving the hardware and software design of the body sensors network to be more user friendly. This can be done by optimizing the electronic components requirements of wireless sensor node, by replacing or adding extra components to the circuit. There is an increasing computational power of microcontrollers and accuracy in sensory devices. A software platform for the nodes in particular and the body network in general might satisfy the user-level operation. This helps eliminate the need of professional/engineering support during the measurement sessions and allows the coaches and athletes to work more closely.

More research can be done to reduce the overall network power consumption, to develop an instant data analysis and transfer functions, data classification and prioritization of information transmission, and to provide larger memory capacities for sensed information including data fusion.

Further research can be undertaken to develop sport-based algorithms to be used to calculate the distance travelled and to calculate the energy expenditure from the recorded acceleration and rotation to help improve training sessions. Furthermore, the vast availability of sensor technology (e.g. using gyroscopes and magnetometers) will help to further remove the sensor orientation and allow a better quantification of these measurements. This would show the capacity of an athlete to maintain the same high level of repeatability during the running or any other sport activity. It can lead to identify fatigue influences on the athlete’s performance and can point to ways of achieving an improved personal training program.

The development and implementation of complex filtering techniques will increase the ability to remove unwanted effects, such the effect of gravity and rotational acceleration, from the signals under study (e.g. forward acceleration). With the resulting clearer acceleration signal, the identification of biomechanical activities, such as the different swing or stance phases, will be significantly increased.

Future work can involve the design of more advanced and complexed machine learning algorithms that automatically enhance the wireless link reliability by changing one or more parameters for each different sensor position and user. The optimization of the central node functions in accordance to its characteristics can provide both on and off body communications and the ability to include multi-hop communication between the hub nodes of different players to increase system reliability and to assist in information delivery to its required destination.

Ten subjects participated in this research during the testing of the wireless body sensor network performance and smart algorithms for enhancing reliability. However, more participants with different experiences and their coaches would be used to test the system for commercial and professional use. Furthermore, more effective methods of data processing and collection techniques are considered to be important issues. Time-stamping and ordering of events, the synchronization of different sensors are open problems for study and research.

The measured signal level was randomly distributed across the movement axis and the same signal level can be obtained at different distances for different limbs. This is probably due to constructive and destructive contributions caused by the multipath effect. Future work may involve an analysis of this effect.

Data compression algorithms at the node level are required to reduce the amount of raw data transmitted during an activity, thus reducing the requirements for long wireless transmissions which can help to minimize the power consumption and channel occupancy. Another important issue is the security of the entire system and privacy provided to ensure data integrity. Further work can be undertaken on the communication key establishment, authentication and data integrity of sensors.

The smart algorithms presented in this thesis showed significant improvement on the network reliability by conducting real network deployments to measure physiological signals and can improve the proposed system. Nevertheless, data transmission should be investigated for low power body sensor networks which is still a challenge.

The issue of integration of multiple sensory devices that operate at different frequencies increases the compatibility problems. Communication between devices operates in multiple bands and use different protocols. This can lead to interference between different devices, especially in the unlicensed industrial, scientific and medical band radio. Tracking human motion for rehabilitation requires continuous monitoring of biological and physical parameters. Future algorithms could also estimate the distance between players on the field by using a triangulation technique based on the connection reliabilities measured by means of subsurface sink nodes.

REFERENCES

[1] "802.15.6-2012 - IEEE Standard for Local and metropolitan area networks - Part 15.6: Wireless Body Area Networks," 2017, Avilable: https://standards.ieee.org/findstds/standard/802.15.6-2012.html, [Accessed: 04 August 2017].

[2] B. Latré, B. Braem, I. Moerman, C. Blondia, and P. Demeester, "A survey on wireless body area networks," Wireless Network, vol. 17, pp. 1-18, 2011.

[3] B. Malik and V. Singh, "A survey of research in WBAN for biomedical and scientific applications," Health and Technology, pp. 1-9, 2013.

[4] C. Otto, E. Jovanov, and A. Milenkovic, "A WBAN-based system for health monitoring at home," 3rd IEEE/EMBS International Summer School on Medical Devices and Biosensors, pp. 20-23, 2006.

[5] R. Marin-Perianu, M. Marin-Perianu, P. Havinga, S. Taylor, R. Begg, M. Palaniswami, et al., "A performance analysis of a wireless body-area network monitoring system for professional cycling," Personal and Ubiquitous Computing, vol. 17, pp. 197-209, 2013.

[6] D. Thiel, H. Espinosa, G. Davis, E. Dylke, N. Foroughi, and S. Kilbreath, "Arm Movement: The Effect of Obesity on Active Lifestyles," Procedia Engineering, vol. 60, pp. 182-187, 2013.

[7] H. Espinosa, D. James, S. Kelly, and A. Wixted, "Sports Monitoring Data and Video Interface Using a GUI Auto Generation Matlab Tool," Procedia Engineering, vol. 60, pp. 243-248, 2013.

[8] J. Wu, L. Dong, and W. Xiao, "Real-time physical activity classification and tracking using wearble sensors," 6th International Conference on Information, Communications & Signal Processing, pp. 1-6, 2007.

[9] S. Milanese, S. Gordon, P. Buettner, C. Flavell, S. Ruston, D. Coe, et al., "Reliability and concurrent validity of knee angle measurement: smart phone app versus universal goniometer used by experienced and novice clinicians," Manual Therapy, vol. 19, pp. 569-74, 2014.

[10] L. Xiong, L. Soon, and T. Taher, "Unrestrained measurement of arm motion based on a wearable wireless sensor network," IEEE Transactions on Instrumentation and Measurement, vol. 59, pp. 1309-1317, 2010.

[11] S. Cotton and W. Scanlon, "A statistical analysis of indoor multipath fading for a narrowband wireless body area network," 17th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1-5, 2006.

[12] A. Alomainy, A. Sani, A. Rahman, J. Santas, and H. Yang, "Transient characteristics of wearable antennas and radio propagation channels for ultrawideband body-centric wireless communications," IEEE Transactions on Antennas and Propagation, vol. 57, pp. 875-884, 2009.

[13] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," 1st International Conference on Embedded Networked Sensor Systems, pp. 1-13, 2003.

[14] X. Liang, X. Li, Q. Shen, R. Lu, X. Lin, X. Shen, et al., "Exploiting prediction to enable secure and reliable routing in wireless body area networks," IEEE Conference on Computer Communications, pp. 388-396, 2012.

[15] M. Habib, G. Nicholas, I. G. William, J. Matthew, M. Bruce, and Y. David, "Review of applications of wireless sensor networks," CRC Press Publisher, pp. 3-32, 2012.

[16] M. Fouad, N. El-Bendary, R. Ramadan, and A. Hassanien, "Wireless sensor networks: A medical perspective," CRC Press Publisher, pp. 713-732, 2012.

[17] P. Hall and Y. Hao, "Antennas and propagation for body centric communications," First European Conference on Antennas and Propagation, pp. 1-7, 2006.

[18] E. Miluzzo, X. Zheng, K. Fodor, and A. Campbell, "Radio characterization of 802.15. 4 and its impact on the design of mobile sensor networks," Wireless Sensor Networks, Springer, pp. 171-188, 2008.

[19] P. Hall and Y. Hao, "Antennas and propagation for body-centric wireless communications," United States: Artech House, 2012.

[20] S. Konstantina, "Design considerations of biomedical telemetry devices," Handbook of Biomedical Telemetry, Wiley-IEEE Press, 2014.

[21] K. Ahmavaara, H. Haverinen, and R. Pichna, "Interworking architecture between 3GPP and WLAN systems," IEEE Communications Magazine, vol. 41, pp. 74-81, 2003.

[22] M. Akhlaq and T. Sheltami, "The recursive time synchronization protocol for wireless sensor networks," IEEE Sensors Applications Symposium, pp. 1-6, 2012.

[23] I. Akyildiz and E. Stuntebeck, "Wireless underground sensor networks: Research challenges," Ad Hoc Networks, vol. 4, pp. 669-686, 2006.

[24] I. Akyildiz and M. Vuran, "Wireless sensor networks," John Wiley & Sons, 2010.

[25] G. Merret and Y. Tan, "Wireless sensor networks: Application centric design," InTech Publisher, 2010.

[26] E. Jovanov, A. Milenkovic, C. Otto, and P. Groen, "A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation," Journal of NeuroEngineering and rehabilitation, vol. 2, p. 6, 2005.

[27] A. Mahapatro, and P. Khilar, "Fault Diagnosis in body sensor networks," International Journal of Computer Information Systems and Industrial Management Applications," vol. 5, pp. 252-259, 2012.

[28] R. Horst, D. Jewett, and D. Lenoski, "The risk of data corruption in microprocessor-based systems," 23th International Symposium on Fault-Tolerant Computing, pp. 576-585, 1993.

[29] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," 2nd international conference on Embedded networked sensor systems, pp. 95-107, 2004.

[30] L. Gu and J. A. Stankovic, "Radio-triggered wake-up for wireless sensor networks," Real-Time Systems, vol. 29, pp. 157-182, 2005.

[31] D. Jiang and L. Delgrossi, "IEEE 802.11 p: Towards an international standard for wireless access in vehicular environments," IEEE Conference on Vehicular Technology, pp. 2036-2040, 2008.

[32] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," IEEE Proceedings of Computer and Communications Societies, pp. 1567-1576, 2002.

[33] I. Lamprinos, A. Prentza, E. Sakka, and D. Koutsouris, "Energy-efficient MAC protocol for patient personal area networks," 27th IEEE International Conference of Engineering in Medicine and Biology Society, pp. 3799-3802, 2005.

[34] W. Ye and J. Heidemann, "Medium access control in wireless sensor networks," Springer, Chapter: Wireless sensor networks, pp. 73-91, 2004.

[35] T. Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," 1st international conference on Embedded networked sensor systems, pp. 171-180, 2003.

[36] G. Lu, B. Krishnamachari, and C. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," 18th International Parallel and Distributed Processing Symposium, pp. 224, 2004.

[37] S. Marinkovic, C. Spagnol, and E. Popovici, "Energy-efficient TDMA-based MAC protocol for wireless body area networks," 3rd International Conference on Sensor Technologies and Applications, pp. 604-609, 2009.

[38] D. Bhatia, L. Estevez, and S. Rao, "Energy efficient contextual sensing for elderly care," IEEE International Conference of Engineering in Medicine and Biology Society, pp. 4052-4055, 2007.

[39] M. Cardei, M. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," IEEE Annual Joint Conference of Computer and Communications Societies, pp. 1976-1984, 2005.

[40] A. Merentitis, N. Kranitis, A. Paschalis, and D. Gizopoulos, "Low energy online self-test of embedded processors in dependable WSN nodes," IEEE Transactions on Dependable and Secure Computing, vol. 9, pp. 86-100, 2012.

[41] M. Renaud, K. Karakaya, T. Sterken, P. Fiorini, C. Hoof, and R. Puers, "Fabrication, modelling and characterization of MEMS piezoelectric vibration harvesters," Sensors and Actuators A: Physical, vol. 145, pp. 380-386, 2008.

[42] V. Leonov, P. Fiorini, S. Sedky, T. Torfs, and C. Hoof, "Thermoelectric MEMS generators as a power supply for a body area network," 13th International Conference on Solid-State Sensors, Actuators and Microsystems, pp. 291-294, 2005.

[43] H. Cheng-Chan and H. Shiow-Yuan, "On the study of a ubiquitous healthcare network with Security and QoS," IET International Conference on Frontier Computing, Theory, Technologies and Applications, pp. 139-144, 2010.

[44] Q. Abbasi, A. Sani, A. Alomainy, and Y. Hao, "On-body radio channel characterization and system-level modeling for multiband OFDM ultra-wideband body-centric wireless network," IEEE Transactions on Microwave Theory and Techniques, vol. 58, pp. 3485-3492, 2010.

[45] E. Monsef, T. Gonnot, Y. Won-Jae, and J. Saniie, "An application-agnostic quality of service framework for wireless body area networks," IEEE Innovation Conference in Healthcare, pp. 30-33, 2014.

[46] K. Ali, J. Sarker, and H. Mouftah, "QoS-based MAC protocol for medical wireless body area sensor networks," IEEE Symposium on Computers and Communications, pp. 216-221, 2010.

[47] C. Yunxia and C. Tellambura, "Distribution functions of selection combiner output in equally correlated Rayleigh, Rician, and Nakagami-m fading channels," IEEE Transactions on Communications, vol. 52, pp. 1948-1956, 2004.

[48] J. Misic, "Enforcing patient privacy in healthcare WSNs using ECC implemented on 802.15. 4 beacon enabled clusters," Sixth IEEE International Conference on Pervasive Computing and Communications, pp. 686-691, 2008.

[49] B. Shu-Di, C. Poon, Z. Yuan-Ting, and S. Lian-feng, "Using the timing information of heartbeats as an entity identifier to secure body sensor network," IEEE Transactions on Information Technology in Biomedicine, vol. 12, pp. 772-779, 2008.

[50] S. Bao, Y. Zhang, and L. Shen, "Physiological signal based entity authentication for body area sensor networks and mobile healthcare systems," International Conference of Engineering in Medicine and Biology Society, pp. 2455-2458, 2005.

[51] V. Srinivasan, J. Stankovic, and K. Whitehouse, "Protecting your daily in-home activity information from a wireless snooping attack," 10th international conference on Ubiquitous computing, pp. 202-211, 2008.

[52] G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski, and R. Jafari, "Enabling effective programming and flexible management of efficient body sensor network applications," IEEE Transactions on Human-Machine Systems, vol. 43, pp. 115-133, 2013.

[53] Y. Jasemian, "Elderly comfort and compliance to modern telemedicine system at home," Second International Conference on Pervasive Computing Technologies for Healthcare, pp. 60-63, 2008.

[54] A. Khandakar and A. Mark, "Data-centric storage in wireless sensor networks," CRC Press Publisher, pp. 33-60, 2012.

[55] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks," Ad Hoc Networks, vol. 1, pp. 215-233, 2003.

[56] W. Liao and C. Chen, "Data storage and range query mechanism for multi-dimensional attributes in wireless sensor networks," IET Communications, vol. 4, pp. 1799-1808, 2010.

[57] H. Shen, L. Zhao, and Z. Li, "A distributed spatial-temporal similarity data storage scheme in wireless sensor networks," IEEE Transactions on Mobile Computing, vol. 10, pp. 982-996, 2011.

[58] B. Fong, A. Fong, and C. Li, "Telemedicine Technologies : Information Technologies in Medicine and Telehealth," Wiley, 2010

[59] S. Armstrong, "Wireless connectivity for health and sports monitoring: a review," British Journal of Sports Medicine, vol. 41, pp. 285-9, 2007.

[60] R. Garcia-Serna, C. Garcia-Pardo, and J. Molina-Garcia-Pardo, "Influence of the posture in body surface to external UWB body area networks channels," IEEE Antennas and Propagation Society International Symposium, pp.1950-1951, 2013.

[61] Y. Hao and R. Foster, "Wireless body sensor networks for health-monitoring applications," Physiological measurement, vol. 29, pp. 27-56, 2008.

[62] V. Bai and S. Srivatsa, "Design and implementation of mobile telecardiac system," Journal of Scientific & Industrial Research, vol. 67, pp. 1059-1063, 2008.

[63] D. Zois, M. Levorato, and U. Mitra, "Energy-efficient, heterogeneous sensor selection for physical activity detection in wireless body area networks," IEEE Transactions on Signal Processing, vol. 61, pp. 1581-1594, 2013.

[64] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour, "Wireless body area networks: A survey," IEEE Communications Surveys & Tutorials, vol. 16, pp. 1658-1686, 2014.

[65] K. Neles, W. Malvezzi, and G. Bressan, "Dealing with uncertainties in the monitoring of patients through sensors networks," Pan American Health Care Exchanges, pp. 50-57, 2012.

[66] "What is Bluetooth: How it works," 2017, Available: https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works, [Accessed: 10 March 2017].

[67] C. Dethe, D. Wakde, and C. Jaybhaye, "Bluetooth Based Sensor Networks Issues and Techniques," in Modelling & Simulation, 2007. AMS '07. First Asia International Conference on, 2007, pp. 145-147.

[68] S. Ergen, "ZigBee/IEEE 802.15. 4 Summary," University of California Berkeley, 2004.

[69] "Low-power, low-cost, low-complexity networking for the Internet of Things," 2017, Available: http://www.zigbee.org/zigbee-for-developers/network-specifications, [Accessed: 10 March 2017].

[70] M. Huang, J. Huang, J. You, and G. Jong, "The wireless sensor network for home-care system using ZigBee," Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 643-646, 2007.

[71] L. Prinslin and V. Janani, "Efficient data delivery in mobility aware ZigBee wireless networks," International Conference on Information Communication and Embedded Systems, pp. 1-5, 2014.

[72] B. Kang and H. Uhm, "Electrical discharge characteristics in water and shockwaves," IEEE Conference Record on Pulsed Power Plasma Science, pp. 547, 2001.

[73] Y. Li, G. Peng, X. Qi, G. Zhou, D. Xiao, S. Deng, et al., "Towards energy optimization using joint data rate adaptation for BSN and WiFi networks," 7th IEEE International Conference on Networking, Architecture and Storage, pp. 235-244, 2012.

[74] Y. Li, X. Qi, Z. Ren, G. Zhou, D. Xiao, and S. Deng, "Energy modeling and optimization through joint packet size analysis of BSN and WiFi networks," 30th IEEE International Performance Computing and Communications Conference, pp. 1-8, 2011.

[75] J. Yu, W. Liao, and C. Lee, "A MT-CDMA based wireless body area network for ubiquitous healthcare monitoring," IEEE Conference on Biomedical Circuits and Systems, pp. 98-101, 2006.

[76] C. Hsiao-Hwa and M. Guizani, "Multiple access technologies for B3G wireless communications," IEEE Communications Magazine, vol. 43, pp. 65-67, 2005.

[77] V. Cionca, T. Newe, and V. Dadrlat, "TDMA Protocol Requirements for Wireless Sensor Networks," Second International Conference on Sensor Technologies and Applications, pp. 30-35, 2008.

[78] A. Rahim, N. Javaid, M. Aslam, Z. Rahman, U. Qasim, and Z. Khan, "A comprehensive survey of MAC protocols for wireless body area networks," Seventh International Conference on Broadband, Wireless Computing, Communication and Applications, pp. 434-439, 2012.

[79] W. Zhiqi, Y. Fengqi, T. Liqiang, and Z. Zusheng, "A fairness spatial TDMA scheduling algorithm for wireless sensor network," 12th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 348-353, 2011.

[80] N. Benjamin and S. Sankaranarayanan, "Performance of hierarchical agent based Wireless Sensor Mesh Network for patient health monitoring," World Congress on Nature & Biologically Inspired Computing, pp. 1653-1656, 2009.

[81] D. Dallas and L. Hanlen, "Optimal transmission range and node degree for multi-hop routing in wireless sensor networks," 4th Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks, pp. 167-174, 2009.

[82] J. Gomez and A. Campbell, "A case for variable-range transmission power control in wireless multihop networks," Twenty-third IEEE Conference of Computer and Communications Societies, pp. 1425-1436, 2004.

[83] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocols for self-organization of a wireless sensor network," IEEE Personal Communications, vol. 7, pp. 16-27, 2000.

[84] A. Krohn, M. Beigl, C. Decker, T. Riedel, T. Zimmer, and D. Garces, "Increasing connectivity in wireless sensor network using cooperative transmission," 3rd International Conference on Networked Sensing Systems, pp. 1-8, 2006.

[85] S. Yousaf, N. Javaid, U. Qasim, N. Alrajeh, Z. Khan, and M. Ahmed, "Towards reliable and energy-efficient incremental cooperative communication for wireless body area networks," Sensors, vol. 16, pp. 284, 2016.

[86] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," IEEE Network, vol. 18, pp. 45-50, 2004.

[87] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," Operating Systems Review, vol. 36, pp. 147-163, 2002.

[88] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," 1st international conference on Embedded networked sensor systems, pp. 138-149, 2003

[89] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," 2nd international conference on Embedded networked sensor systems, pp. 39-49, 2004.

[90] M. Maggs, S. O'Keefe, and D. Thiel, "Consensus clock synchronization for wireless sensor networks," IEEE Sensors Journal, vol. 12, pp. 2269-2277, 2012.

[91] L. Huaming and T. Jindong, "Heartbeat-driven medium-access control for body sensor networks," IEEE Transactions on Information Technology in Biomedicine, vol. 14, pp. 44-51, 2010.

[92] "OMNeT++ Discrete Event Simulator," 2017, Avilable: https://omnetpp.org/, [Accessed: 04 August 2017].

[93] J. Kirkup, D. Rowlands, and D. Thiel, "Indoor propagation investigation from a 2.4GHz waist mounted beacon," Procedia Engineering, vol. 60, pp. 188-194, 2013.

[94] T. Daniel, S. Braveena, K. Rezaul, and P. Marimuthu, "Sensor networks in healthcare in Healthcare Sensor Networks," CRC Press Publisher, pp. 1-20, 2011.

[95] " Rules & Regulations for Title 47," 2017, Avilable: https://www.fcc.gov/encyclopedia/rules-regulations-title-47, [Accessed: 04 August 2017].

[96] " Impact of industrial, scientific and medical (ISM) equipment on radiocommunication services," 2017, Avilable: http://www.itu.int/pub/R-REP-SM.2180, [Accessed: 04 August 2017].

[97] H. Cao, V. Leung, C. Chow, and H. Chan, "Enabling technologies for wireless body area networks: A survey and outlook," IEEE Communications Magazine, vol. 47, pp. 84-93, 2009.

[98] L. Shipeng, G. Yu, J. Shengming, and T. Pink, "A lightweight and robust interference mitigation scheme for wireless body sensor networks in realistic environments," Wireless Communications and Networking Conference, pp. 1697-1702, 2014.

[99] C. Bin, G. Yu, K. Wee, F. Gang, H. Tan, and L. Yun, "An experimental study for inter-user interference mitigation in wireless body sensor networks," IEEE Sensors Journal, vol. 13, pp. 3585-3595, 2013.

[100] S. Wen, G. Yu, and W. Wai-Choong, "Inter-user interference in body sensor networks: A case study in moderate-scale deployment in hospital environment," IEEE 14th International Conference on e-Health Networking, Applications and Services, pp. 447-450, 2012.

[101] J. Choi, H. Kang, and Y. Choi, "A study on the wireless body area network applications and channel models," International Conference on Future Generation Communication and Networking, pp. 263-266, 2008.

[102] A. Wood, G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, et al., "ALARM-NET: Wireless sensor networks for assisted-living and residential monitoring," University of Virginia Computer Science Department Technical Report, vol. 2, 2006.

[103] T. Gao, D. Greenspan, M. Welsh, R. Juang, and A. Alm, "Vital signs monitoring and patient tracking over a wireless network," 27th International Conference of the Engineering in Medicine and Biology Society, pp. 102-105, 2005.

[104] P. Shaltis, A. Reisner, and H. Asada, "Wearable, cuff-less PPG-based blood pressure monitor with novel height sensor," 28th Annual International Conference of the Engineering in Medicine and Biology Society, pp. 908-911, 2006.

[105] H. Ghasemzadeh, V. Loseu, E. Guenterberg, and R. Jafari, "Sport training using body sensor networks: A statistical approach to measure wrist rotation for golf swing," Fourth International Conference on Body Area Networks, pp. 1-8, 2009.

[106] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton, "Codeblue: An ad hoc sensor network infrastructure for emergency medical care," International workshop on wearable and implantable body sensor networks, pp. 1-4,2004.

[107] J. Ko, J. Lim, Y. Chen, R. Musvaloiu, A. Terzis, G. Masson, et al., "MEDiSN: medical emergency detection in sensor networks," ACM Transactions on Embedded Computing Systems, vol. 10, pp. 1-25, 2010.

[108] J. Ng, B. Lo, O. Wells, M. Sloman, N. Peters, A. Darzi, et al., "Ubiquitous monitoring environment for wearable and implantable sensors (UbiMon)," International Conference on Ubiquitous Computing, pp. 1-2, 2004.

[109] R. Chakravorty, "A programmable service architecture for mobile medical care," IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 1-5, 2006.

[110] M. Munoz, R. Foster, and Y. Hao, "Exploring Physiological Parameters in Dynamic WBAN Channels," IEEE Transactions on Antennas and Propagation, vol. 62, pp. 5268-5281, 2014.

[111] T. Novacheck, "The biomechanics of running," Gait Posture, vol. 7, pp. 77-95, Jan 1998.

[112] P. Weyand, D. Sternlight, M. Bellizzi, and S. Wright, "Faster top running speeds are achieved with greater ground forces not more rapid leg movements," Journal of Applied Physiology, vol. 89, pp. 1991-1999, 2000.

[113] S. Sarkar, P. Phillips, Z. Liu, I. Vega, P. Grother, and K. Bowyer, "The humanID gait challenge problem: data sets, performance, and analysis," IEEE Transaction Pattern Analysis Machine Intelligence , vol. 27, pp. 162-77, 2005.

[114] R. Mayagoitia, A. Nene, and P. Veltink, "Accelerometer and rate gyroscope measurement of kinematics: an inexpensive alternative to optical motion analysis systems," Journal of Biomechanics, vol. 35, pp. 537-42, 2002.

[115] E. Scilingo, F. Lorussi, A. Mazzoldi, and D. Rossi, "Strain-sensing fabrics for wearable kinaesthetic-like systems," IEEE Sensors Journal, vol. 3, pp. 460-467, 2003.

[116] C. Strohrmann, H. Harms, C. Kappeler-Setz, and G. Troster, "Monitoring kinematic changes with fatigue in running using body-worn sensors," IEEE Transactions on Information Technology in Biomedicine, vol. 16, pp. 983-990, 2012.

[117] D. Brunelli, E. Farella, L. Rocchi, M. Dozza, L. Chiari, and L. Benini, "Bio-feedback system for rehabilitation based on a wireless body area network," Fourth IEEE International Conference on Pervasive Computing and Communications Workshops, pp. 1-5, 2006.

[118] J. Neville, A. Wixted, D. Rowlands, and D. James, "Accelerometers: An underutilized resource in sports monitoring," Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, pp. 287-290, 2010.

[119] "CardioNet," 2017, Avilable: https://www.cardionet.com/, [Accessed: 04 August 2017].

[120] A. Karlstädt, D. Fliegner, G. Kararigas, H. Ruderisch, V. Regitz-Zagrosek, and H. Holzhütter, "CardioNet: a human metabolic network suited for the study of cardiomyocyte metabolism," BMC systems biology, vol. 6, pp. 1-20, 2012.

[121] P. Kulkarni and Y. Öztürk, "Requirements and design spaces of mobile medical care," Mobile Computing and Communications Review, vol. 11, pp. 12-30, 2007.

[122] B. Rogers, L. Severe, and P. Eggers, "Method and apparatus for monitoring and communicating with an implanted medical device," Patent US 6957107 B2 , 2005.

[123] Y. Lee, D. Lee, G. Walia, R. Myllylae, and W. Chung, "Query based duplex vital signal monitoring system using wireless sensor network for ubiquitous healthcare," World Congress on Medical Physics and Biomedical Engineering vol. 14, pp. 396-397, 2007.

[124] D. James, N. Davey, and T. Rice, "An accelerometer based sensor platform for insitu elite athlete performance analysis," Proceedings of IEEE,Sensors, vol. 3, pp. 1373-1376, 2004.

[125] R. Takeda, S. Tadano, M. Todoh, M. Morikawa, M. Nakayasu, and S. Yoshinari, "Gait analysis using gravitational acceleration measured by wearable sensors," Journal of biomechanics, vol. 42, pp. 223-233, 2009.

[126] C. Vaughan, B. Davis, and J. Connor, "Dynamics of human gait," Second Edition ed., Kiboho Publishers, 1999.

[127] M. Vidojkovic, X. Huang, P. Harpe, S. Rampu, C. Zhou, L. Huang, et al., "A 2.4GHz ULP OOK single-chip transceiver for healthcare applications," IEEE Transactions on Biomedical Circuits and Systems, vol. 5, pp. 523-534, 2011.

[128] P. Kalyanasundaram, L. Huang, K. Imamura, and G. Dolmans, "Autonomous operation of super-regenerative receiver in BAN," International Symposium on Medical Information and Communication Technology, pp. 1-4, 2012.

[129] R. Hagem, H. Sabti, and D. Thiel, "Coach-swimmer communications based on wrist mounted 2.4GHz accelerometer sensor," Procedia Engineering, vol. 112, pp. 512-516, 2015.

[130] H. Sabti and D. Thiel, "Movement based time division multiplexing for near real-time feedback body area network applications," International Workshop on Antenna Technology: Small Antennas, Novel EM Structures and Materials, and Applications, pp. 22-24, 2014.

[131] H. Sabti and D. Thiel, "Node position effect on link reliability for body centric wireless network running applications," IEEE Sensors Journal, vol. 14, pp. 2687-2691, 2014.

[132] H. Sabti and D. Thiel, "A study of wireless communication links on a body centric network during running," Procedia Engineering, vol. 72, pp. 3-8, 2014.

[133] H. Sabti and D. Thiel, "Wearable wireless sensor for real-time angle measurements," Journal of Fitness Research, vol. 5, pp. 28-30, 2016.

[134] N. Semiconductor, "nRF24L01 Ultra low power 2.4GHz RF Transceiver IC," Available: http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01, [Accessed 10 March 2017].

[135] H. Sabti and D. Thiel, "Self-calibrating body sensor network based on periodic human movements," IEEE Sensors Journal, vol. 15, pp. 1552-1558, 2015.

[136] H. Sabti and D. Thiel, "High reliability body sensor network using gesture triggered burst transmission," Procedia Engineering, vol. 112, pp. 507-511, 2015.

[137] H. Sabti and D. Thiel, "Time multiplexing-star shape body sensor network for sports applications," IEEE Antennas and Propagation Society International Symposium, pp. 969-970, 2014.

[138] H. Sabti and D. Thiel, "CSMA Identification of Moving Neighbours in Body Sensor Networks,", Submitted to International Sports Engineering Association 2018.

APPENDICES

7.1 Appendix A – Wireless Node Design

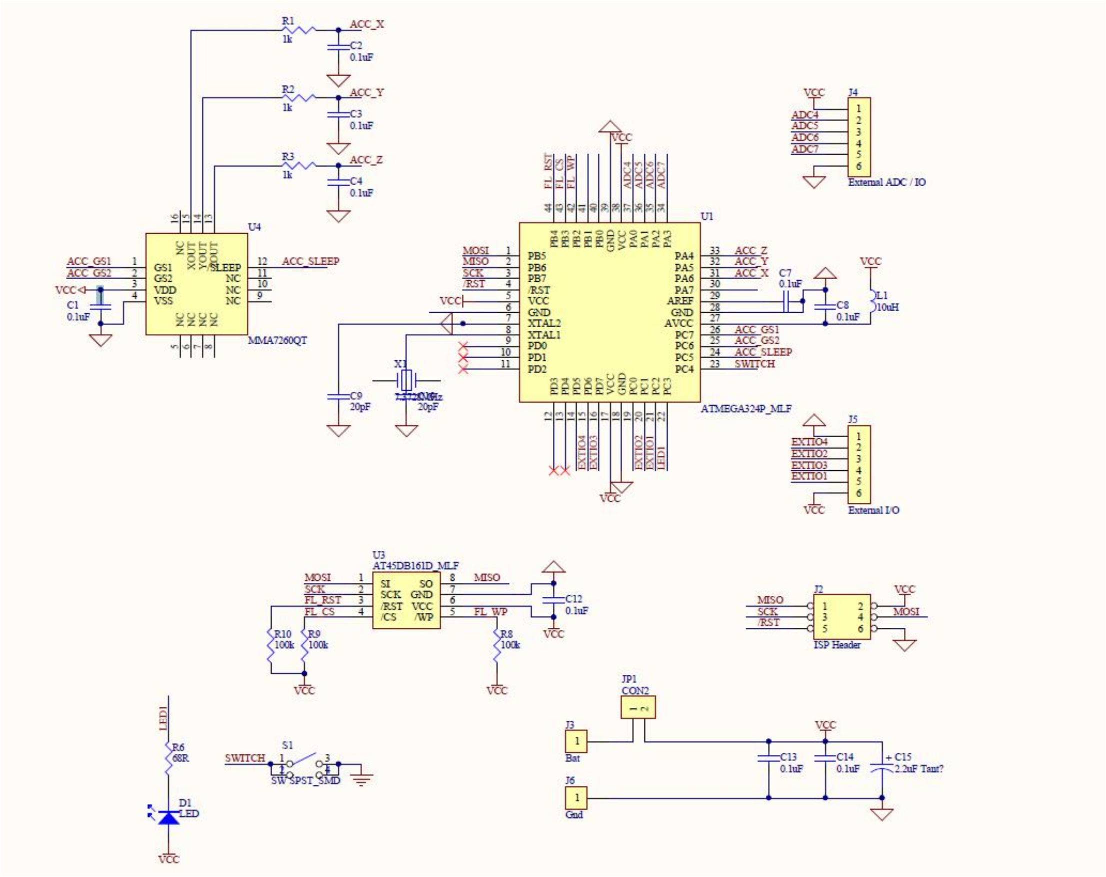


Figure 7-1 The schematic of the developed sensor node.

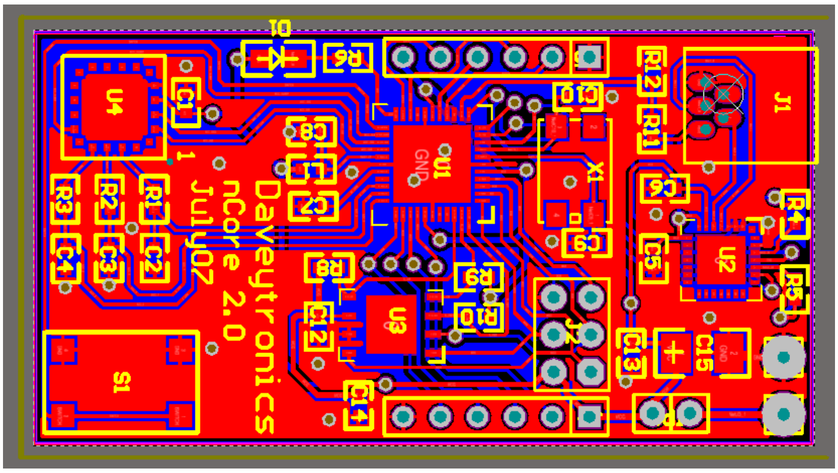


Figure 7-2 The PCB upper layer of the developed sensor node.

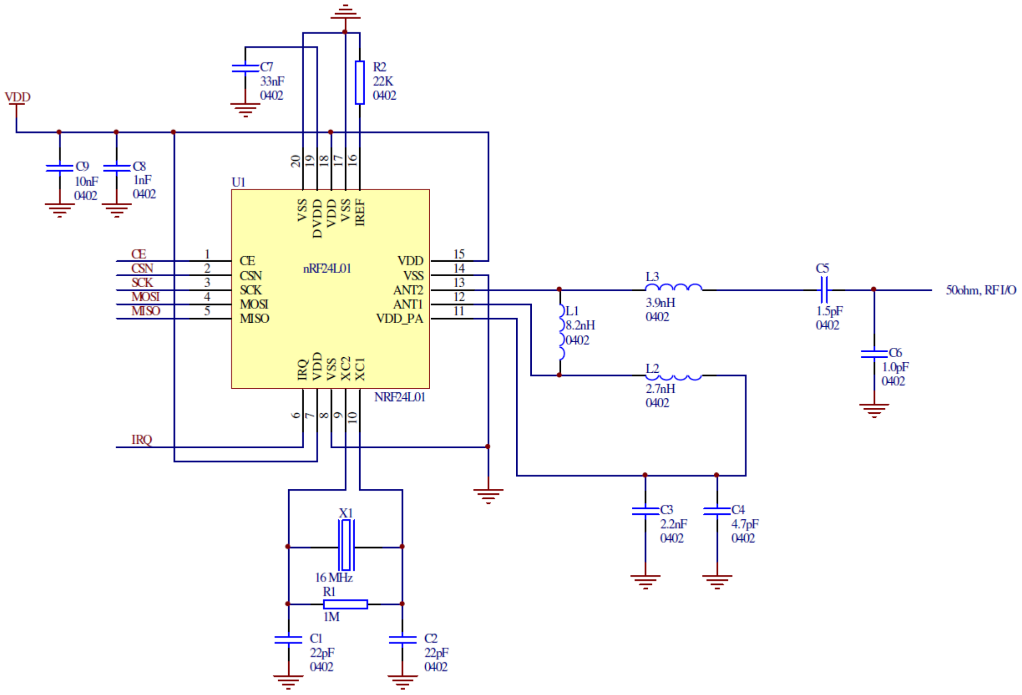


Figure 7-3 The nRF24L01 schematic for RF layouts with single ended matching network crystal, bias resistor, and decoupling capacitors.

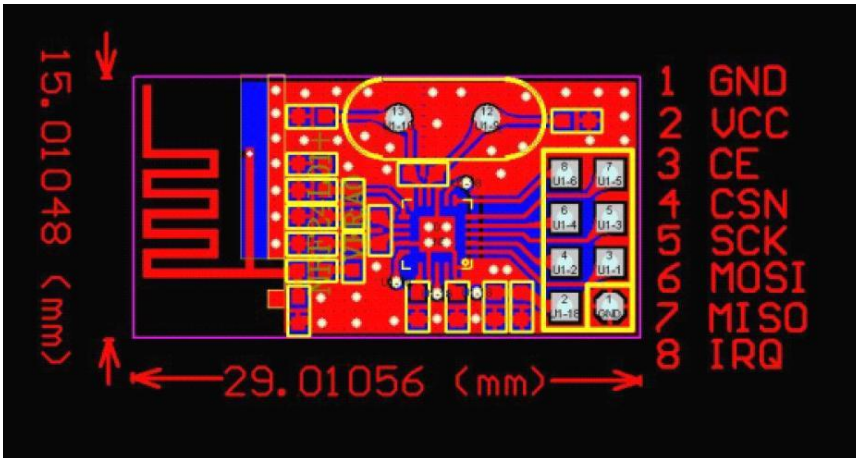


Figure 7-4 The PCB upper layer of the nRF24L01 wireless transceiver.

7.2 Appendix B - MatLab Swing Time Calculation Algorithm

```
%% swing time calculations

value = transpose(accx);
len = length(value);
threshold=100;                      % threshold of amplitude
difference between the lowest vaules in two swings
holder=1;                           % to hold the counter value of
the last minimum until reaching the count level
count =1;                           % count up to time window
counter=1;                          % counter for values
window=40;                          % window of values to find min
r=round(len/window);
swing_count=1;                      % swing counter (usually count 2
swings to compare minimum values)
swing_time(r+1)=0;                  % time difference between the two
minimum values
minimum (len)=0;                    % find the  minimums values -->
lowest(r)=0;                        % find the lowest values amoung
the minimum values
pointer(r)=0;                       % point to the lowest value
% two_swing=0;                      % counter for two swings
% even_counter=1;                   % count each two swings

while (counter<len)
    count =count+1;
    counter =counter+1;
    if (counter < len)
    if value(counter) < value(counter-1) && value(counter) <
value(counter+1)  % compare the three values and register only
minimums --> V
        minimum(counter) = value(counter);
        holder=counter;
    end
    if (count==window)  % after half a sec find lowest minimum
        lowest(swing_count) = minimum(holder);  % put last recorded
minimum in lowest
        for j=(1+((swing_count-1)*window)):counter %compare all
minimum values for each swing to find the lowest between them
            if ((minimum(j) <= lowest(swing_count)) &&
minimum(j)~=0)  % compare minimum values are not equal to zero
                lowest(swing_count) = minimum(j);
                pointer(swing_count)= j;
            end
        end
        count =0;
        swing_count=swing_count+1;  % carfull swing_count is going
to be equal 3 after the first two swings
        swing_time(swing_count)= abs((pointer(swing_count-1))-
(pointer(swing_count)));
    end
    end
end
```

7.3 Appendix C - Transceiver Sensor Embedded C Code

7.3.1 Self-Calibrating Code for the Hub

```
#define F_CPU 7372800UL  // 8MHz
#include <avr/io.h>
#include <avr/signal.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <avr/eeprom.h>
#include <inttypes.h>
#include <stdlib.h>
#include <util/delay.h>
#include "util.h"
#include "rf24.h"
//-----------------------------------------
// Definitions
//-----------------------------------------
#define LED_ON      PORTC &= ~_BV(PC3)  // turn led on
#define LED_OFF     PORTC |= _BV(PC3)   // turn led off
#define DEVICE_MODE_UNKNOWN         0
#define DEVICE_MODE_USB_CONNECTED   1
#define DEVICE_MODE_STAND_ALONE     2
#define DEVICE_OPERATION_NONE               0
#define DEVICE_OPERATION_LOGGING_CONTINUE   1
#define DEVICE_OPERATION_LOGGING_STOP       2
#define DEVICE_OPERATION_CLEAR_FLASH        3
#define DEVICE_OPERATION_DETAILS_DOWNLOAD   4
#define DEVICE_OPERATION_DATA_M_DOWNLOAD    5
#define DEVICE_OPERATION_LOGGING_SET_MARK   6
#define DEVICE_OPERATION_LOGGING_START_MARK 7
#define FLASH_BUFFER_1      1
#define FLASH_BUFFER_2      2
#define LOGGING_DISABLED    0
#define LOGGING_ENABLED     1
#define COMMAND_RECEIVE_DISABLED    0
#define COMMAND_RECEIVE_ENABLED     1
#define COMMAND_LENGTH            255
#define LED_CNT_LIMIT   30
#define LED_ON      PORTC &= ~_BV(PC3)  // turn led on
#define LED_OFF     PORTC |= _BV(PC3)   // turn led off
unsigned char gDevice_Mode = DEVICE_MODE_UNKNOWN; // used to
determine if the usb is connected or not
unsigned char gDevice_Operation = DEVICE_OPERATION_NONE; // used to
determine operation mode, based on conneciton mode
unsigned char led_pattern = 0; // use this to control led pattern?
unsigned char led_mode = 1;
unsigned char gFlash_Buffer = FLASH_BUFFER_1;
unsigned int flash_page = 0;
unsigned int gFlash_Buffer_Cnt = 0;
volatile unsigned char gLogging_State = LOGGING_DISABLED;
volatile unsigned char gCommand_Receive_Mode =
COMMAND_RECEIVE_DISABLED;
unsigned char status, pipe; //register to hold letter sent and
received
// unsigned int adc_val_x_p0, adc_val_x_p1, adc_val_x_p2,
adc_val_x_p3, indicator_p0, indicator_p1, indicator_p2,
indicator_p3;
```

```
short int counter1=0;
short int saveone;
short int savetwo;
short int savethree;
unsigned long int location=0;
short int location_count=0;
short int i =0;         // for button toggling (strat with button 
not pressed)
short int swith_flag =0;
short int transmit =0;
short int learn=0;
short int sample =0;
short int tensamples =0;
short int ten_tensamples =0;
short int arc_cnt= 0;
short int pols_cnt= 0;
short int data= 0;
short int rt_count=0;
short int rx_ind = 0;
short int indicator = 0; // 1 byte
short int data1,data2;
short int k =101;
short int j =0;
short int threshold=100;                      // threshold of 
amplitude difference between the lowest vaules in two swings
short int count =0;
unsigned long int counter=0;                           // count up 
to time window
short int window=50;                          // window of values to 
find min
int accelration_x[50];
short int swing_count=0;                     // swing counter 
(usually count 2 swings to compare minimum values)
short int ten_swing_count=0;
short int total_swings=0;
int swing_time[100];                   // time difference between 
the two minimum values
int lowest[100];                       // find the lowest values 
amoung the minimum values
int pointer[100];                      // point to the lowest value
long int txdata= 0; // 4 byte
long int rxdata= 0; // 4 byte
// short int count_one; count_two; count_three; count_four;
int accelration_one[50];
int accelration_two[50];
int accelration_three[50];
int accelration_four[50];
unsigned int rx_accx =0;
unsigned int rx_accy =0;
unsigned int rx_accz =0;
unsigned int adc_val_x = 0; // 2 byte
unsigned int adc_val_y = 0;
unsigned int adc_val_z = 0;
unsigned int led_timecnt = LED_CNT_LIMIT;
unsigned int timecnt = 0;               //Testing value
//-----------------------------------------
// ButtonPressed
//-----------------------------------------
int Pressed;
int Pressed_Confidence_Level; //Measure button press confidence
int Released_Confidence_Level; //Measure button release confidence
```

```
char ButtonPressed(unsigned char pinOfButton, int portBit, int
confidenceLevel)
{
    if (bit_is_clear(pinOfButton, portBit))
    {
        Pressed_Confidence_Level ++; //Increase Pressed Confidence
        Released_Confidence_Level = 0; //Reset released button
confidence since there is a button press
        if (Pressed_Confidence_Level > confidenceLevel) //Indicator
of good button press
        {
            if (Pressed == 0)
            {
                Pressed = 1;
                return 1;
            }
            Pressed_Confidence_Level = 0;   //Zero it so a new
pressed condition can be evaluated
        }
    }
    else
    {
        Released_Confidence_Level ++; //This works just like the
pressed
        Pressed_Confidence_Level = 0; //Reset pressed button
confidence since the button is released
        if (Released_Confidence_Level > confidenceLevel)
        {
            Pressed = 0;
            Released_Confidence_Level = 0;
        }
    }
    return 0;
}
//------------------------------------------
// Get Led State
//------------------------------------------
unsigned char Get_Led_State(void)
{
    if(bit_is_clear(PINC,PC3))
    {
        return(TRUE);
    }
    else
    {
        return(FALSE);
    }
}
//------------------------------------------
// Clear Led Mode
//------------------------------------------
void Clear_Led_Mode(void)
{
    led_mode = 0;
}
//------------------------------------------
// Inc Led Mode
//------------------------------------------
void Inc_Led_Mode(void)
{
    // if flash led equals 3 after this inc then loop back to zero
(so only 2 modes really..)
    led_mode++;

    if(led_mode >= 3)   //7 Modes are the maximum
    led_mode = 0;

}
//-----------------------------------------
// Get Led Mode
//-----------------------------------------
unsigned char Get_Led_Mode(void)
{
    return(led_mode);
}
//--------------------------------------------
// Clear Command
//--------------------------------------------
void Clear_Command(unsigned char* command_array)
{
    int cnt;
    for(cnt = 0;cnt<COMMAND_LENGTH;cnt++)
    command_array[cnt] = ' ';
}
//--------------------------------------------
// Read Command
//--------------------------------------------
void Read_Command(unsigned char* command_array)
{
    int cnt;
    USART_Flush();
    //USART_PutString("Read Command exicuted\r\n");     //This line
is just a indicator for function usage
    for(cnt = 0;cnt<COMMAND_LENGTH;cnt++)
    {
        command_array[cnt] = USART_Receive();
        if(command_array[cnt] == 13)
        break;
    }
}
//--------------------------------------------
// Get Command
//--------------------------------------------
void Get_Command(unsigned char* command_array)
{
    int cnt;
    for(cnt=0;cnt<COMMAND_LENGTH;cnt++)
    {
        USART_Transmit(command_array[cnt]);
        if(command_array[cnt] == 13)
        break;
    }
    USART_PutString("\r\n");
}
//--------------------------------------------
// Check Command
//--------------------------------------------
unsigned char Check_Command(unsigned char* command_array,const char*
command_reference)
{
    int cnt;
    unsigned char command_test_value = FALSE;

    if(*command_array == 13)
    return FALSE;
    for(cnt=0;cnt<COMMAND_LENGTH;cnt++)
    {
        if(command_array[cnt] == 13)
        break;
        //  USART_Number_Transmit(command_test_value);
        if(command_array[cnt] != (*command_reference++))
        return command_test_value;
    }

    return TRUE;
}
//-----------------------------------------
// Task Button Processing
//-----------------------------------------
void Task_Button_Processing(void)
{
    unsigned char command_array[COMMAND_LENGTH];
    unsigned char command_controll = FALSE; //controle the command
legality
    //unsigned char command_reference[] = {"delete"};
    //unsigned char session;
    if(gDevice_Mode == DEVICE_MODE_USB_CONNECTED)
    {
        if(ButtonPressed(PINC, 4, 5))
        {
            //USART_Number_Transmit(timecnt);
            USART_PutString("Device connected to USB.\r\n");
            USART_PutString("To see all commands press 'h'+
Enter\r\n");
            USART_PutString("DeviceNr:         ");
            USART_Number_Transmit(DEVICE_NUMBER);
            USART_PutString("\r\n");
            USART_PutString("Recorded sessions:");

USART_Number_Transmit(Get_EEPROM_2Bytes(EEPROM_ADDRESS_SESSION));
            USART_PutString("\r\n\n");

            gCommand_Receive_Mode = COMMAND_RECEIVE_ENABLED;
        }
        else
        {
            if( gCommand_Receive_Mode == COMMAND_RECEIVE_ENABLED)
            {

                USART_PutString("Enter command...\r\n");
                Clear_Command(command_array);
                LED_OFF;
                Read_Command(command_array);
                //USART_Number_Transmit(timecnt);
                if(command_array[0] != 'm')
                Get_Command(command_array);

                if(Check_Command(command_array,"e") &&
command_controll == FALSE)
                {
                    command_controll = TRUE;
                    USART_PutString("You are going to erase all 
recorded data.\r\nDo you want to continue?? y/n\r\n");
                    USART_PutString("Enter command...\r\n");
                    Clear_Command(command_array);
                    Read_Command(command_array);
                    Get_Command(command_array);
                    if(Check_Command(command_array,"y"))
                    //USART_PutString("clear Flash command\r\n");
                    gDevice_Operation = 
DEVICE_OPERATION_CLEAR_FLASH;
                    else
                    {
                        command_controll = 3;
                        USART_PutString("Erase canceled.\r\n");
                    }
                }
                if(Check_Command(command_array,"s") && 
command_controll == FALSE)
                {
                    command_controll = TRUE;
                    //USART_PutString("Download details\r\n");
                    gDevice_Operation = 
DEVICE_OPERATION_DETAILS_DOWNLOAD;
                }

                if(Check_Command(command_array,"d") && 
command_controll == FALSE)
                {
                    command_controll = TRUE;
                    led_timecnt = 10;  // command indicator
                    //USART_PutString("Download data\r\n");
                    gDevice_Operation = 
DEVICE_OPERATION_DATA_M_DOWNLOAD;
                }
                if(Check_Command(command_array,"h") && 
command_controll == FALSE)
                {
                    command_controll = 3;
                    USART_PutString("HELP:\r\n\ne  erase memory\r\ns  
download details\r\nd  download data\r\nq  quit\r\nh  help\r\n");
                    gDevice_Operation = DEVICE_OPERATION_NONE;
                }
                if(Check_Command(command_array,"q") && 
command_controll == FALSE)
                {
                    command_controll = TRUE;
                    USART_PutString("Have a nice Day...\r\n");
                    gDevice_Operation = DEVICE_OPERATION_NONE;
                    led_mode = 2;
                }

                if(Check_Command(command_array,"hello") && 
command_controll == FALSE)
                {
                    command_controll = 3;
                    USART_PutString("Hi,\r\nhow are you?\r\n");
                    gDevice_Operation = DEVICE_OPERATION_NONE;
                }
                //case of unknown command
                if(command_controll == TRUE)
                gCommand_Receive_Mode = COMMAND_RECEIVE_DISABLED;
                else
                {
                    if(command_controll == FALSE)
                    USART_PutString("unknown command\r\n");
                }

            }
        }
    }
    else
    if(gDevice_Mode == DEVICE_MODE_STAND_ALONE)
    {
        //if(ButtonPressed(PINC, 4, 5))
        {
            // do something if we have a short button press
            if(gLogging_State == LOGGING_DISABLED)
            {
                Clear_Led_Mode();
                gDevice_Operation =
DEVICE_OPERATION_LOGGING_CONTINUE;
            }
            /*else
                                {
                                    gDevice_Operation =
DEVICE_OPERATION_LOGGING_SET_MARK;

                                }*/
        }
    }
}
//-------------------------------------------
// Task Flash Led
//-------------------------------------------
void Task_Flash_Led(void)
{
    if((gLogging_State == LOGGING_ENABLED) && (led_mode == 0)) //
toggle the led slowly when logging
    {
        if(led_pattern == 0)
        {
            if(Get_Led_State())
            {
                LED_OFF;
            }
            else
            {
                LED_ON;
            }
        }
    }
    else
    if((led_mode != 0)&&(led_mode <= 5)) // else toggle the led when
indicating a button press
    {
        // enable toggle led
        if(led_pattern <= (led_mode*2))
        {
            if(led_pattern%2)
            {
                LED_ON;
            }
            else
            {
                LED_OFF;
            }
        }
        else
        {
            LED_OFF;
        }
    }
    else
    if(led_mode == 6)
    {
        LED_ON;
    }
    else
    {
        LED_OFF;
    }

    led_pattern++;
    if(led_timecnt < LED_CNT_LIMIT)
    {
        LED_ON;
        led_timecnt++;
    }
    if(led_pattern == 10)
    led_pattern = 0;
}
//-----------------------------------------
// BG Task Data Download
//-----------------------------------------
void BG_Task_Data_Download(void)
{
    unsigned int cnt;
    unsigned char data1;
    unsigned char data2;
    unsigned char data_cnt;
    unsigned int fcnt;
    unsigned int data;
    //USART_PutString("X,Y,Z,\r\n");
    for(fcnt=0;fcnt<4096;fcnt++)
    {
        // read array
        PORTB &= ~_BV(PB3); // flash cs low
        _delay_ms(1.0);
        SPI_Write(0x03);
        SPI_Write((((unsigned long)fcnt << 10) >> 16) & 0xff);
        SPI_Write((((unsigned long)fcnt << 10) >>  8) & 0xff);
        SPI_Write((((unsigned long)fcnt << 10)      ) & 0xff);
        data1 = SPI_Read();
        data2 = SPI_Read();
        if((data1 == 'A') && (data2 == 'P'))
        {
            data_cnt = 0;
            for(cnt=0;cnt<258;cnt++)
            {
                data = SPI_Read() << 8;
                data += SPI_Read();

                USART_Number_Transmit(data);
                data_cnt++;

                if(data_cnt == 3)
                {
                    data_cnt = 0;
                    USART_PutString("\r\n");
                }
                else
                {
                    USART_Transmit(',');
                }
            }
        }
        else
        {
            PORTB |= _BV(PB3); // flash cs high
            _delay_ms(1.0);
            break;
        }
        PORTB |= _BV(PB3); // flash cs high
        _delay_ms(1.0);

    }

    USART_PutString("Download complete.\r\n");
    //to enter new command
    gCommand_Receive_Mode = COMMAND_RECEIVE_ENABLED;
}
//-----------------------------------------
// Task Data Download
//-----------------------------------------
void Task_Data_Download(void)
{
    unsigned int session;
    unsigned int scnt;
    /*  USART_Number_Transmit(timecnt);
    USART_PutString("gDevice_Operation =");
    USART_Number_Transmit(gDevice_Operation); //Test
    USART_PutString("\r\n");
*/
    if(gDevice_Operation == DEVICE_OPERATION_CLEAR_FLASH)
    {
        USART_PutString("Flash clearing is in proccess.\r\n");

        Init_EEPROM();
        gDevice_Operation = DEVICE_OPERATION_NONE;
        USART_PutString("Flash clearing done\r\n\n");
        //to enter new command
        gCommand_Receive_Mode = COMMAND_RECEIVE_ENABLED;
    }
    if(gDevice_Operation == DEVICE_OPERATION_DETAILS_DOWNLOAD)
    {
        //Get number of writen flash pages
        flash_page = Get_EEPROM_2Bytes(EEPROM_ADDRESS_FLASH_PAGE);
        //flash_page = eeprom_read_byte(1);
        USART_PutString("Flash_page  = ");
        USART_Number_Transmit(flash_page);
        USART_PutString("\r\n");

        //Get Sessions count
        USART_PutString("Sessions    = ");
        session = Get_EEPROM_2Bytes(EEPROM_ADDRESS_SESSION);
        USART_Number_Transmit(session);
        USART_PutString("\r\n\n");
        USART_PutString("Session Nr. = page || line\r\n");
        for(scnt = 1;scnt <= session;scnt++)
        {
            USART_PutString("Session");
            USART_Number_Transmit(scnt);
            USART_PutString(" = ");
            //submit flashpage of session start

USART_Number_Transmit(Get_EEPROM_2Bytes(scnt*2+EEPROM_ADDRESS_SESSIO
N));
            USART_PutString(" || ");
            //submit line of session start in hyperterminal

USART_Number_Transmit(Get_EEPROM_2Bytes(scnt*2+EEPROM_ADDRESS_SESSIO
N)*86);

            USART_PutString("\r\n");
        }
        USART_PutString("\r\n");
        gDevice_Operation = DEVICE_OPERATION_NONE;
        //to enter new command
        gCommand_Receive_Mode = COMMAND_RECEIVE_ENABLED;
    }

    if(gDevice_Operation == DEVICE_OPERATION_DATA_M_DOWNLOAD)
    {
        // add a background task to download flash pages..
        USART_Number_Transmit(DEVICE_NUMBER);
        USART_PutString("\r\n");
        USART_Number_Transmit(TIMER0_COMPARE);
        USART_PutString("\r\n");
        BG_Task_Data_Download();
        gDevice_Operation = DEVICE_OPERATION_NONE;
    }
}
//-----------------------------------------
// Task Data Logging Start
//-----------------------------------------
void Task_Data_Logging_Start(void)
{
    unsigned int session;
    // handle the logging start task if that mode is set
    if(gDevice_Operation == DEVICE_OPERATION_LOGGING_CONTINUE)
    {
        if(gLogging_State == LOGGING_DISABLED)
        {
            //#warning This is bad.. it should really be in a
background task..
            // erase entire flash.. takes 15 seconds..
            // turn led on solid to indicate

            //Set_Accel_Gain(ACCEL_GAIN_2G0); // set gain to 2.0G
            //Set_Accel_Sleep(ACCEL_SLEEP_DISABLED); // wake up
accel
            flash_page =
Get_EEPROM_2Bytes(EEPROM_ADDRESS_FLASH_PAGE);
            session = Get_EEPROM_2Bytes(EEPROM_ADDRESS_SESSION);
            //EEPROM protokoll update

            session++;
            Set_EEPROM_2Bytes(EEPROM_ADDRESS_SESSION,session);
            //write session address into EEPROM

Set_EEPROM_2Bytes((session*2)+EEPROM_ADDRESS_SESSION,flash_page);
            // lets clear the sram1
            Flash_Clear_SRAM(FLASH_BUFFER_1,0x00);
            // lets clear the sram2
            Flash_Clear_SRAM(FLASH_BUFFER_2,0x00);
            gDevice_Operation = DEVICE_OPERATION_LOGGING_START_MARK;
            gLogging_State = LOGGING_ENABLED;
            // reset buffer flag and flash page..
            gFlash_Buffer = FLASH_BUFFER_1;
            Flash_Fill_Buffer_UCHAR(FLASH_BUFFER_1,0,'A');
            Flash_Fill_Buffer_UCHAR(FLASH_BUFFER_1,1,'P');
            Flash_Fill_Buffer_UCHAR(FLASH_BUFFER_2,0,'A');
            Flash_Fill_Buffer_UCHAR(FLASH_BUFFER_2,1,'P');
            gFlash_Buffer_Cnt = 2;
        }
    }
}
//-------------------------------------------
// Task Data Logging Stop
//-------------------------------------------
void Task_Data_Logging_Stop(void)
{
    // handle the logging stop task if that mode is set
    if(gDevice_Operation == DEVICE_OPERATION_LOGGING_STOP)
    {
        if(gLogging_State == LOGGING_ENABLED)
        {
            gDevice_Operation = DEVICE_OPERATION_NONE;
            gLogging_State = LOGGING_DISABLED;
            //Set_Accel_Sleep(ACCEL_SLEEP_ENABLED); // put accel to
sleep
            //wirte current flashpage to EEPROM for logging continue
        }
    }
}
//-------------------------------------------
// Task Data Logging
//-------------------------------------------
void Task_Data_Logging(void)
{
    // handle the logging of data..
    if(gLogging_State == LOGGING_ENABLED)
    {

        // get adc values..
        //  adc_val_x = Get_ADC_Sample(6);
        //  adc_val_y = Get_ADC_Sample(5);
        //  adc_val_z = Get_ADC_Sample(4);

        if(gDevice_Operation == DEVICE_OPERATION_LOGGING_START_MARK)
        {

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0x00);
Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0x00);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0x00);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0x00);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0x00);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0x00);


//Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0x00);

//Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0x00);

            gDevice_Operation = DEVICE_OPERATION_NONE;
        }
        else
        {
            if(gDevice_Operation ==
DEVICE_OPERATION_LOGGING_SET_MARK)
            {

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0xff);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0xff);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0xff);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0xff);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0xff);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0xff);

                //
Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0xff);
                //
Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,0xff);

                gDevice_Operation = DEVICE_OPERATION_NONE;
            }
            else
            {

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,(saveone
>> 8) & 0xff);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,(saveone
) & 0xff);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,(savetwo
>> 8) & 0xff);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,(savetwo
) & 0xff);
                //
Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,(adc_val_z
>> 8) & 0xff);
```

```
                //
Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,(adc_val_z
) & 0xff);


Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,(savethree
>> 8) & 0xff);

Flash_Fill_Buffer_UCHAR(gFlash_Buffer,gFlash_Buffer_Cnt++,(savethree
) & 0xff);

                // USART_Number_Transmit(saveone);
                // USART_Transmit(',');
                // USART_Number_Transmit(savetwo);
                // USART_Transmit(',');
                // USART_Number_Transmit(savethree);
                // USART_Transmit(',');
                // USART_PutString("\r\n");

            }
        }
        //  gDevice_Operation = DEVICE_OPERATION_NONE; //test value
        // we will store  samples per buffer (buffers are 528 bytes)
        // (518/6)*sample rate = time required to fill the buffer
        if(gFlash_Buffer_Cnt == 518)
        {
            if(gFlash_Buffer == FLASH_BUFFER_1)
            {

Flash_Write_Buffer_To_Main(FLASH_BUFFER_1,flash_page++);

                gFlash_Buffer = FLASH_BUFFER_2;
                gFlash_Buffer_Cnt = 0;
                Flash_Fill_Buffer_UCHAR(FLASH_BUFFER_2,0,'A');
                Flash_Fill_Buffer_UCHAR(FLASH_BUFFER_2,1,'P');
                gFlash_Buffer_Cnt = 2;
            }
            else
            {

Flash_Write_Buffer_To_Main(FLASH_BUFFER_2,flash_page++);
                gFlash_Buffer = FLASH_BUFFER_1;
                gFlash_Buffer_Cnt = 0;
                Flash_Fill_Buffer_UCHAR(FLASH_BUFFER_1,0,'A');
                Flash_Fill_Buffer_UCHAR(FLASH_BUFFER_1,1,'P');
                gFlash_Buffer_Cnt = 2;
            }

            //write last flash page to EEPROM
            Set_EEPROM_2Bytes(EEPROM_ADDRESS_FLASH_PAGE,flash_page);

            //Storage protection
            if(flash_page == 4096)
            {
                gDevice_Operation = DEVICE_OPERATION_LOGGING_STOP;
                led_mode = 6;
            }
        }
    }
}
//-----------------------------------------
```

```
// Timers
//------------------------------------------
void Timer_Start(void)
{
    TCCR2B = 0; // disable timer;
    TCNT2 = 0;
    // clock = 8MHz (not sure on acuracy)
    // 16 bit timer

    //TCCR2B |= _BV(CS30);  // start timer, no prescaler (clk/1)
resolution is 1/8000000 -> 125ns    ~max = 65535*125ns  = 8.19ms
    //TCCR2B = 0b00000010;  // start timer, clk/8
resolution is 1/1000000 -> 1us      ~max = 65535*1us    = 65.5ms
    //TCCR2B = 0b00000011;  // start timer, clk/64
resolution is 1/125000 -> 8us       ~max = 65535*8us    = 524.28ms
    TCCR2B = 0b00000101;    // start timer, max prescaler (clk/1024)
resolution is 1/7812 -> 128us       ~max = 65535*128us  = 8.39s

    //TIMSK2 = 0b00000001; //Enable Overflow interrupts
    TIMSK2 = 0b00000000; //Disable Overflow interrupts

}
void Timer_Stop(void)
{
    TCCR2B = 0; // disable timer;
    //test_data[test_data_count++]=TCNT2;
    //fprintf(&USBSerialStream,"\nTime: %u\n", TCNT2);
    USART_Number_Transmit(TCNT2);
    USART_PutString("\r\n");
}
//------------------------------------------
// Toggle LED
//------------------------------------------
void ToggleLED_Fast(void)
{
    TCCR1B |= 1<<CS10 | 1<<CS11 | 1<<WGM12; // prescale 64 + wave
genration mode
    TIMSK1 |= 1<<OCIE1A; //If using atmega324, this regester is
TIMSK1
    OCR1A = 31250; // 1/4 sec
}
void ToggleLED_Slow(void)
{
    TCCR1B |= 1<<CS10 | 1<<CS11 | 1<<WGM12; // prescale 64 + wave
genration mode
    TIMSK1 |= 1<<OCIE1A; //If using atmega324, this regester is
TIMSK1
    OCR1A = 62500; // 1/2 sec
}
//------------------------------------------
// Sampling, RX & TX Mode
//------------------------------------------
void sampling(void)
{
    TCCR1B |= 1<<CS10 | 1<<CS11 | 1<<WGM12; // prescale 64 + wave
genration mode
    TIMSK1 |= 1<<OCIE1A; //If using atmega324, this regester is
TIMSK1
    OCR1A = 625; // 5ms
}
void node_tx(void)
```

```
{
    TCCR1B |= 1<<CS10 | 1<<CS11 | 1<<WGM12; // prescale 64 + wave
genration mode
    TIMSK1 |= 1<<OCIE1A; //If using atmega324, this regester is
TIMSK1
    OCR1A = 1250; // 0.01 sec (10ms)
}
void node_rx(void)
{
    TCCR1B |= 1<<CS10 | 1<<CS12 | 1<<WGM12; // prescale 1024 + wave
genration mode
    TIMSK1 |= 1<<OCIE1B; //If using atmega324, this regester is
TIMSK1
    OCR1B = 3900; // 1/2 sec
}
//-------------------------------------------
// Hardware Setup
//-------------------------------------------
void Hardware_Setup(void)
{
    cli(); // clear global interrupts
    DDRD |= _BV(PD3);   // cts to cp2102
    PORTD &= ~_BV(PD3);
    DDRD &= ~_BV(PD2);  // rts input from cp2102
    DDRD &= ~_BV(PD4);  // usb sense input
    PORTD &= ~_BV(PD4); // no internal pull up
    DDRC |= _BV(PC3);   // led output
    PORTC |= _BV(PC3);  // led is off
    DDRC &= ~_BV(PC4);  // switch input
    PORTC |= _BV(PC4);  // switch is high (NOT Pressed)
    DDRC |= _BV(PC5);   // accel sleep control
    DDRC |= _BV(PC6);   // accel GS2
    DDRC |= _BV(PC7);   // accel GS1
    DDRB |= _BV(PB4);   // flash reset output
    DDRB |= _BV(PB3);   // flash chip select output
    DDRB |= _BV(PB2);   // flash write protect output

    PORTB |= _BV(PB4); // flash reset high
    PORTB |= _BV(PB3); // flash cs high
    PORTB |= _BV(PB2); // flash wp high

    DDRB |= _BV(PB5);   // MOSI output
    DDRB &= ~_BV(PB6); // MISO input
    DDRB |= _BV(PB7);   // SCK output

    DDRC |= _BV(PC1);   // CSN output
    DDRC |= _BV(PC2);   // CE output

    nrf24l01_set_csn(); // CSN high
    nrf24l01_clear_ce(); // CE low (standby mode)

    DDRC &= ~_BV(PD7);  // IRQ input

    //Set_Accel_Gain(ACCEL_GAIN_1G5); // set gain to 1.5G
    Set_Accel_Gain(ACCEL_GAIN_2G0); // set gain to 2.0G
    //Set_Accel_Sleep(ACCEL_SLEEP_DISABLED); // wake up accel
    Set_Accel_Sleep(ACCEL_SLEEP_ENABLED); // and put acccel to sleep
    USART_Init(UART_BAUD_115200);
    Init_ADC();
    SPI_Init();
}
```

```
//-----------------------------------------
// Initialization
//-----------------------------------------
void Initialization(void)
{
    // init flash
    if(Flash_Init() == FALSE)
    {
        Flash_Reset();
        if(Flash_Init() == FALSE)
        {
            led_mode = 5;   // problem initalising flash...
        }
    }
    //-----------------------------------------

    // init EEPROM
    if(Get_EEPROM_2Bytes(EEPROM_ADDRESS_INIT) != 23130) // 23130 =
ZZ
    Init_EEPROM();
    //-----------------------------------------

    // init RF
    nrf24l01_initialize_mod(true, 3, false); // set as (Rx, payload
bytes , auto-ack enabled)

    //-----------------------------------------
}
//-----------------------------------------
// main
//-----------------------------------------
int main(void)
{
    Hardware_Setup();
    Initialization();
    sei();

    while(1)
    {


        // USB connection status determines what OS operation and
function is
        if(USB_Status()) // USB connected
        {
            gDevice_Mode = DEVICE_MODE_USB_CONNECTED; // USB mode
for Task_Button_Processing

            Task_Button_Processing();
            Task_Data_Download();

            ToggleLED_Slow();
        }

        else  // USB not connected
        {
            gDevice_Mode = DEVICE_MODE_STAND_ALONE; // logging data
mode for Task_Button_Processing
            Task_Button_Processing();
            if(nrf24l01_irq_pin_active() &&
nrf24l01_irq_rx_dr_active())
```

```
            {

                nrf24l01_irq_clear_all(); //clear all interrupts in
the 24L01

                nrf24l01_read_rx_payload(&rxdata, 3); //read the
packet into data + 130 uS delay is needed
                // (function clears CE to wirte commands to RF and
then sets CE back to activate RX mode)

                rx_accx = (rxdata >> 8) & 0xffff;  // highest 2
bytes, counting from left: bits 8-23
                rx_ind =  rxdata   & 0xff;  // low-order byte: bits
0-7


                saveone      =   rx_ind;
                savetwo      =   location;
                savethree    =   rx_accx;

                Task_Data_Logging_Start();  // start recoeding after
button pressed
                Task_Data_Logging();        // data that are
recorded
                Task_Data_Logging_Stop(); // stop recoeding when
memory is fill (protection)

                rxdata = 0; //  txdata is zero unless receive
something

                // #node1 if start receving at different time slot
or the array counter exceeds the limits (calculate the highest and
reset counter)
                if (location != check1+1 || counter1 == 50)
                {
                    // if received samples are more than 10 && acc
data lies between(2 sec to 8 sec) calcuate the highest
                    if (counter1 >= 10 && (location >= 400 ||
location <= 9600))
                    {
                        highest1[count1] = accelration_one[0];  //
put last recorded minimum in lowest
                        pointer1[count1]= 0;

                        for (j=1;j<=counter1; j++) //compare all
minimum values for each swing to find the lowest between them
                        {
                            if (accelration_one[j] >=
highest1[count1])  // compare minimum values are not equal to zero
                            {
                                highest1[count1] =
accelration_one[j];

                                pointer1[count1]= j;
                            }
                        }
                        saveone      =   11;
                        savetwo      =   pointer1[count1];
                        savethree    =   highest1[count1];

                        count1++;
                        Task_Data_Logging_Start();  // start
recoeding after button pressed
                        Task_Data_Logging();        // data that are
recorded
                        Task_Data_Logging_Stop(); // stop recoeding
when memory is fill (protection)
                    }
                    counter1 = 0;
                }

                // #node2 if start receving at different time slot
or the array counter exceeds the limits (calculate the highest and
reset counter)
                if (location != check2+1 || counter2 == 50)
                {
                    // if received samples are more than 10 && acc
data lies between(2 sec to 8 sec) calcuate the highest
                    if (counter2 >= 10 && (location >= 400 ||
location <= 9600))
                    {
                        highest2[count2] = accelration_two[0];  //
put last recorded minimum in lowest
                        pointer2[count2]= 0;

                        for (j=1;j<=counter2; j++) //compare all
minimum values for each swing to find the lowest between them
                        {
                            if (accelration_two[j] >=
highest2[count2])  // compare minimum values are not equal to zero
                            {
                                highest2[count2] =
accelration_two[j];

                                pointer2[count2]= j;
                            }
                        }
                        saveone     =   12;
                        savetwo     =   pointer2[count2];
                        savethree   =   highest2[count2];

                        count2++;

                        Task_Data_Logging_Start();  // start
recoeding after button pressed
                        Task_Data_Logging();        // data that are
recorded
                        Task_Data_Logging_Stop(); // stop recoeding
when memory is fill (protection)
                    }
                    counter2 = 0;
                }

                // #node3 if start receving at different time slot
or the array counter exceeds the limits (calculate the highest and
reset counter)
                if (location != check3+1 || counter3 == 50)
                {
                    // if received samples are more than 10 && acc
data lies between(2 sec to 8 sec) calcuate the highest
                    if (counter3 >= 10 && (location >= 400 ||
location <= 9600))
                    {
                        highest3[count3] = accelration_three[0];
// put last recorded minimum in lowest
                        pointer3[count3]= 0;

                        for (j=1;j<=counter3; j++) //compare all
minimum values for each swing to find the lowest between them
                        {
                            if (accelration_three[j] >=
highest3[count3])  // compare minimum values are not equal to zero
                            {
                                highest3[count3] =
accelration_three[j];
                                pointer3[count3]= j;
                            }
                        }
                        saveone     =   13;
                        savetwo     =   pointer3[count3];
                        savethree   =   highest3[count3];

                        count3++;

                        Task_Data_Logging_Start();  // start
recoeding after button pressed
                        Task_Data_Logging();        // data that are
recorded
                        Task_Data_Logging_Stop(); // stop recoeding
when memory is fill (protection)
                    }
                    counter3 = 0;
                }

                // #node4 if start receving at different time slot
or the array counter exceeds the limits (calculate the highest and
reset counter)
                if (location != check4+1 || counter4 == 50)
                {
                    // if received samples are more than 10 && acc
data lies between(2 sec to 8 sec) calcuate the highest
                    if (counter4 >= 10 && (location >= 400 ||
location <= 9600))
                    {
                        for (j=0;j<=counter4; j++) // calculate the
number of elements in the array that not equal to zero
                        {
                            if (accelration_four[j] != 0)  //
compare element values are not equal to zero
                            {
                                element4++;
                            }
                        }

                        middle4 = element4/2; // find the middle
point

                        average4[count4] =
accelration_four[middle4];   // acceleration at that point
                        pointer4[count4] = check4-middle4;
// distance (location) form the 400 sample

                        saveone     =   14;
                        savetwo     =   average4[count4];
                        savethree   =   pointer4[count4];

                        count4++;

                        Task_Data_Logging_Start();  // start
recoeding after button pressed
                        Task_Data_Logging();        // data that are
recorded
                        Task_Data_Logging_Stop(); // stop recoeding
when memory is fill (protection)
                    }
                    counter4 = 0;
                }

                // put the accleration values into array for each
node
                if (counter1 <= 49)
                {
                    if (rx_ind==1)
                    {
                        accelration_one[counter1] = savethree;
                    }
                    if (check1 != location) counter1++;
                    check1=location;
                }

                if (counter2 <= 49)
                {
                    if (rx_ind==1)
                    {
                        accelration_two[counter2] = savethree;
                    }
                    if (check2 != location) counter2++;
                    check2=location;
                }

                if (counter3 <= 49)
                {
                    if (rx_ind==1)
                    {
                        accelration_three[counter3] = savethree;
                    }
                    if (check3 != location) counter3++;
                    check3=location;
                }

                if (counter4 <= 49)
                {
                    if (rx_ind==1)
                    {
                        accelration_four[counter4] = savethree;
                    }
                    if (check4 != location) counter4++;
                    check4=location;
                }

                //Timer_Stop();
            }

            if (sample==1 && !(nrf24l01_irq_pin_active() &&
nrf24l01_irq_rx_dr_active()))
```

```
            {
                //Timer_Start();

                // adc_val_x = Get_ADC_Sample(6);   // getting the x
vaule for the transmission and logging (2 bytes)
                // adc_val_y = Get_ADC_Sample(5);
                // adc_val_z = Get_ADC_Sample(4);

                data1 = ((adc_val_x >> 8) & 0xff); // MSB
                data2 = ((adc_val_x     ) & 0xff);  // LSB

                txdata = (( data1 << 8 ) | data2);  // shifting MSB
& LSB of adc_val_x to fit in txdata
                txdata = (( txdata << 8 ) | (k)); // shifting
adc_val_x one byte to make room for indicator ( total 3 bytes)

                nrf24l01_irq_clear_all();
                nrf24l01_set_as_tx();

                nrf24l01_write_tx_payload(&txdata, 3, true);
//transmit FF char over RF + 130 uS delay is needed
                // (function wirte commands to RF and then sets CE
for 10uS to activate TX mode)


                //wait until the packet has been sent or the maximum
number of retries has been active
                while(!(nrf24l01_irq_pin_active() &&
(nrf24l01_irq_tx_ds_active() || nrf24l01_irq_max_rt_active())));

                nrf24l01_read_register(nrf24l01_OBSERVE_TX, &data,
1);      // read OBSERVE_TX value into data
                arc_cnt = (data & nrf24l01_OBSERVE_TX_ARC_CNT);
//count number of retransmits of a single packet (reset when
transmitting a new packet)
                pols_cnt = ((data & nrf24l01_OBSERVE_TX_PLOS_CNT) >>
4);    //count the number of packets lost after the max number of
retransmits.
                nrf24l01_clear_plos_cnt();

                //check to see if the maximum number of retries has
been hit.  if not, clear tx fifo
                if(!nrf24l01_irq_max_rt_active()) // auto ak
received
                {
                    nrf24l01_irq_clear_all(); //clear all interrupts
in the 24L01
                }

                else // max retransmit reached
                {
                    rt_count++;
                    nrf24l01_flush_tx(); //get the unsent character
out of the TX FIFO
                    nrf24l01_irq_clear_all(); //clear all interrupts
                }

                nrf24l01_set_as_rx(true);

                txdata=0;
                sample=0;
                k++; location_count++;
                if (k==105) k=101;
                if (location_count==5) location_count=1;
                if (location_count==1) location++;

            }
        }

        if (i==0) // button not pressed
        {
            if (swith_flag==0) ToggleLED_Fast();
            swith_flag=1;
        }

        if (i==1) // button pressed
        {
            if (swith_flag==1) sampling();
            swith_flag =0;
        }

        // if (ButtonPressed(PINC, 4, 5)) // (pinport, pinnumber,
confidence level)
        // {
        // if (i==0) indicator++; // increase once each two button
clicks

        // i ^= 1;
        // }
    }
}
ISR(TIMER1_COMPA_vect)
{
    if (i==0) PORTC ^= 1<<PINC3;
// if receiving mode(i==0) toggle led each 1/4 sec
    if (i==1) sample=1;
// sample each 10ms
    if (i==1 && tensamples==10) {PORTC ^= 1<<PINC3; tensamples=0;}
// if transmitting mode(i==1) toggle led each 100ms
}
```

7.3.2 *Self-Calibrating Code for the Transceiver*

The same code for parameter definitions, supporting functions and initialization of the hub sensor node is used in each terminal, however, the main function is different for each one. Below main function is for sensor node 1.

```
//-----------------------------------------
// main
//-----------------------------------------
int main(void)
{
    Hardware_Setup();
    Initialization();
    sei();
    while(1)
    {

        // USB connection status determines what OS operation and
function is
        if(USB_Status()) // USB connected
        {
            gDevice_Mode = DEVICE_MODE_USB_CONNECTED; // USB mode
for Task_Button_Processing

            Task_Button_Processing();
            Task_Data_Download();

            ToggleLED_Slow();
        }

        else  // USB not connected
        {
            gDevice_Mode = DEVICE_MODE_STAND_ALONE; // logging data
mode for Task_Button_Processing
            Task_Button_Processing();
            if(nrf24l01_irq_pin_active() &&
nrf24l01_irq_rx_dr_active())
            {
                nrf24l01_irq_clear_all(); //clear all interrupts in
the 24L01

                nrf24l01_read_rx_payload(&rxdata, 3); //read the
packet into data + 130 uS delay is needed
                // (function clears CE to wirte commands to RF and
then sets CE back to activate RX mode)

                rx_accx = (rxdata >> 8) & 0xffff;  // highest 2
bytes, counting from left: bits 8-23
                rx_ind =  rxdata   & 0xff;  // low-order byte: bits
0-7

                if (rx_ind==101) sample=1;
                rxdata = 0; //  txdata is zero unless receive
something

                // USART_Number_Transmit(rx_ind);
                // USART_PutString("\r\n");


            }

            if (sample==1 && !(nrf24l01_irq_pin_active() &&
nrf24l01_irq_rx_dr_active()))
            {
                // adc_val_x = Get_ADC_Sample(6);   // getting the x
vaule for the transmission and logging (2 bytes)
                adc_val_y = Get_ADC_Sample(5);
                // adc_val_z = Get_ADC_Sample(4);

                data1 = ((adc_val_y >> 8) & 0xff); // MSB
                data2 = ((adc_val_y      ) & 0xff);  // LSB

                txdata = (( data1 << 8 ) | data2);  // shifting MSB
& LSB of adc_val_x to fit in txdata
```

```
                txdata = (( txdata << 8 ) | (1)); // shifting
adc_val_x one byte to make room for indicator ( total 3 bytes)

                nrf24l01_irq_clear_all();
                nrf24l01_set_as_tx();

                nrf24l01_write_tx_payload(&txdata, 3, true);
//transmit FF char over RF + 130 uS delay is needed
                // (function wirte commands to RF and then sets CE
for 10uS to activate TX mode)


                //wait until the packet has been sent or the maximum
number of retries has been active
                while(!(nrf24l01_irq_pin_active() &&
(nrf24l01_irq_tx_ds_active() || nrf24l01_irq_max_rt_active())));
                //{ USART_Transmit('L');}
                nrf24l01_read_register(nrf24l01_OBSERVE_TX, &data,
1);      // read OBSERVE_TX value into data
                arc_cnt = (data & nrf24l01_OBSERVE_TX_ARC_CNT);
//count number of retransmits of a single packet (reset when
transmitting a new packet)
                pols_cnt = ((data & nrf24l01_OBSERVE_TX_PLOS_CNT) >>
4);    //count the number of packets lost after the max number of
retransmits.
                nrf24l01_clear_plos_cnt();

                //check to see if the maximum number of retries has
been hit.  if not, clear tx fifo
                if(!nrf24l01_irq_max_rt_active()) // auto ak
received
                {
                    nrf24l01_irq_clear_all(); //clear all interrupts
in the 24L01
                }

                else // max retransmit reached
                {
                    rt_count++;
                    nrf24l01_flush_tx(); //get the unsent character
out of the TX FIFO
                    nrf24l01_irq_clear_all(); //clear all interrupts
                }

                saveone     =   1;
                savetwo     =   indicator;
                savethree   =   adc_val_y;

                // USART_PutString("TX1");
                // USART_Transmit(',');
                // USART_Number_Transmit(indicator);
                // USART_Transmit(',');
                // USART_Number_Transmit(adc_val_y);
                // USART_PutString("\r\n");

                // Task_Data_Logging_Start();   // start recoeding
after button pressed
                // Task_Data_Logging();     // data that are
recorded
                // Task_Data_Logging_Stop(); // stop recoeding when
memory is fill (protection)
```

```
                nrf24l01_set_as_rx(true);

                txdata=0;
                sample=0;
            }

            if (i==0) // button not pressed
            {
                if (swith_flag==0) ToggleLED_Fast();
                swith_flag=1;
            }

            if (i==1) // button pressed
            {
                if (swith_flag==1) sampling();
                swith_flag =0;
            }

            // if (ButtonPressed(PINC, 4, 5)) // (pinport,
pinnumber, confidence level)
            // {
            // if (i==0) indicator++; // increase once each two
button clicks

            // i ^= 1;
            // }
        }
    }
}
ISR(TIMER1_COMPA_vect)
{
    if (i==0) PORTC ^= 1<<PINC3;
// if receiving mode(i==0) toggle led each 1/4 sec
    if (i==1) sample=1;
// sample each 10ms
    if (i==1 && tensamples==10) {PORTC ^= 1<<PINC3; tensamples=0;}
// if transmitting mode(i==1) toggle led each 100ms
}
```

7.4 *Appendix D - MatLab Real-time Interactive Monitoring Program Code*

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%% Initializing variables
i=1;
h=1;
count=1;
block_number=1;
sampling=20;
cycle=100;
min_acc(cycle)=0;
```

```
max_acc(cycle)=0;
size(cycle)=0;
pointer_max_acc(cycle)=0;
pointer_min_acc(cycle)=0;
swing_time(cycle)=0;

% coneccting the serail port with specified baud ratec
s=serial('COM10', 'BaudRate', 115200);
try
    fopen(s);
catch err
    fclose(instrfind);
    error('Make sure you select the correct COM Port where the 
Sensor is connected.');
end
get(s)

% endless loop
while (1)
    % scanning the serial port for data and allocating part of the 
string to cell arrays
    %tic
    acc = fgets(s),
    [token,remain]=strtok(acc);
    Indicator(i) = str2double(token);
    Accelration(i) = str2double(remain);
    M = [Indicator; Accelration];
    K=M';
    if (block_number ~= Indicator(i))  % after half a sec find 
lowest minimum
        block_number = Indicator(i);
        size(count)= h;
        swing_time(count) = size(count)*sampling;
        min_acc(count)= Accelration(i);
        max_acc(count)= Accelration(i);

        % min_acc acceleration
        for j=i:-1:(i-h+1)
            if ((Accelration(j) <= min_acc(count)) || 
~isnumeric(Accelration(i)))  % compare min_accimum values are not 
equal to zero
                min_acc(count) = Accelration(j);
                pointer_min_acc(count)= j;
            end
        end

        % max_acc acceleration
        for j=i:-1:(i-h+1)
            if ((Accelration(j) >= max_acc(count)) || 
~isnumeric(Accelration(i)))  % compare min_accimum values are not 
equal to zero
                max_acc(count) = Accelration(j);
                pointer_max_acc(count)= j;
            end
        end

        if mod(block_number,2)
            drawnow;
            set(handles.edit1,'String', min_acc(count));
            set(handles.edit2,'String', max_acc(count));
            set(handles.edit3,'String', size(count));
            set(handles.edit4,'String', swing_time(count));
            set(handles.edit9,'String', min_acc_av);
            set(handles.edit10,'String', max_acc_av);
            set(handles.edit11,'String', size_av);
            set(handles.edit12,'String', swing_av);
            axes(handles.axes1);
            plot(Accelration)
            set(get(handles.axes1, 'XLabel'), 'String', 'Simple
number');
            set(get(handles.axes1, 'YLabel'), 'String',
'Acceleration');
%             set(get(handles.axes1, 'Title'), 'String', 'Haider');
            if (i<100) xlim([1 110]);
            else xlim([(i+1-100) (i+10)]);
            end
            axes(handles.axes3);
            bar(swing_time)
            set(get(handles.axes3, 'XLabel'), 'String', 'Cycle
number');
            set(get(handles.axes3, 'YLabel'), 'String', 'Swing time
(ms)');
            if (count<10) xlim([1 11]);
            else xlim([(count+1-10) (count+1)]);
            end

        else
            drawnow;
            set(handles.edit5,'String', min_acc(count));
            set(handles.edit6,'String', max_acc(count));
            set(handles.edit7,'String', size(count));
            set(handles.edit8,'String', swing_time(count));
            set(handles.edit13,'String', min_acc_av);
            set(handles.edit14,'String', max_acc_av);
            set(handles.edit15,'String', size_av);
            set(handles.edit16,'String', swing_av);
            axes(handles.axes2);
            plot(Accelration)
            set(get(handles.axes2, 'XLabel'), 'String', 'Simple
number');
            set(get(handles.axes2, 'YLabel'), 'String',
'Acceleration');
            if (i<100) xlim([1 110]);
            else xlim([(i+1-100) (i+10)]);
            end
                        axes(handles.axes4);
            bar(swing_time)
            set(get(handles.axes4, 'XLabel'), 'String', 'Cycle
number');
            set(get(handles.axes4, 'YLabel'), 'String', 'Swing time
(ms)');
            if (count<10) xlim([1 11]);
            else xlim([(count+1-10) (count+1)]);
            end
        end
        count = count+1;
        h=0;
    end
    h=h+1;
    i=i+1;
    % average calculations
    size_av = (mean(size)*cycle)/count;
    swing_av = (mean(swing_time)*cycle)/count;
    max_acc_av = (mean(max_acc)*cycle)/count;
    min_acc_av = (mean(min_acc)*cycle)/count;
    %toc
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
clear all;
try
    fopen(s);
catch err
    fclose(instrfind);
end
```