

Collage of Engineering and Technology

Electrical Engineering Department

College of IT and Computer Engineering

Department of computer engineering and science



Bachelor Thesis

ZigBee-Based Monitoring System for People with Mental Disabilities

Project Team

Deema Herbawi

Sa'di Al-Tamimi

Luma Sharabati

Yusra Al-Qaisi

Project Supervisor

Dr. Ghandi F. Manasrah

Hebron-Palestine

June, 2013

إهداء؛

للذين استحقوا دعائنا .. ودموعنا .. وصلواتنا .. وابتهالاتنا ..
للذين لهم حق علينا، لعنا بهذا نسترضي الضمير الذي ما فتى يجلدنا ويذكرنا بواجبنا، وبنهر فينا ضعفنا وقلة
حيلتنا ..
اليهم جميعا في غرة والشام؛ حيث أرواحهم طافت كل البلاد وأعلنت طهر العباد .. وكتبت بالأحمر المداد
تاريخ أمة لها من الشهداء العداد .. العداد ..
إلى كل قطرة عرق تساقطت من جبين أبي وصبت في مجرى دمي لتغدو لي لحما ودما وهوية ..
إلى كل دمة حب وخوف وقلق سقطت من عين أُمي لتروي دربي وتجعله أخضرا ياتعا اسير به بعفوية ..
إلى من حملوا مشاعل النور في زمن الظلام و علمونا كيف نصنع نورنا بكل حرفية ..
إلى من شاركونا درب الحياة وملكوا من دمنا وذكرياتنا وطفولتنا .. اخوتنا وأصدقائنا كل التحية ..
إلى جامعة بوليتكنك فلسطين صاحبة الفضل والرفعة الأبية ..
إلى كل عنقود عنب خليلي يُطعم شهدا و يعلم معنى الحرية ..
إلى كل حجر في بيت قديم ما زال يحتفظ برائحة هذه المدينة الأصلية ..
إلى جبالها التي نستمد منها رفعتنا وعزتنا الخليلية ..
إليك يا وطني الذي قُتلت ألف ألف مرة .. وفي كل مرة بُعثت لتعلن لهم .. أنك باقٍ للأبدية .
إلى سمانك المرصعة بالشهداء والأسرى واللاجئين نرفع مشروعا ليكون نجمة تضيء للبرية ..

We would like to dedicate this thesis to our loving parents who gave us everything they had to facilitate our success. We are so proud of you. You trust us, and we hope that we were deserved it.

Acknowledgements

We would like to express our heartfelt gratitude to our advisor Professor Ghandi F. Manasrah for his constant encouragement and guidance throughout the course of our bachelors study and all the stages of the writing of this thesis. Without his instruction, this thesis work could not have reached its present form. His editorial advice and infinite patience were essential for the completion of this thesis. We feel privileged to have had the opportunity to study under him. You have trusted us, you kept challenging us to do our best. And we hope that we made you proud of us as we always proud of you.

We also would like to thank everyone who has helped us along the way. Special thanks to our friend Ibrahim Dwaik, and Mr.Monther Jubran for their help in making some of the hardware components available to us so that the implementation stage becomes possible. At last, we should mention that this graduation project has been supported by the Deanship of Graduate Studies and Scientific Research through "Distinguished Graduation Projects Fund".

Abstarct

People with mental illnesses are group of people who need special care and constant attention. Although their abilities to think and control their actions are limited - or may be disabled, they still can move easily and don't stick to their places where they are assumed to be. Thus, there is a substantial need to be aware of their locations to guarantee their safety. For that, an indoor monitoring system based on location determination was designed especially for hospitals and treatment centers. The system exploits ZigBee technology to enable patient's monitoring. The location of each patient was determined using artificial neural networks through the received signal strength measured from ZigBee transceiver attached to each patient. Results are displayed on a pre-uninstalled map and are stored together with relative important information about each patient in a secured database. The system was able to localize patients at room level in real time with accuracy of 97%.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Overview | 1 |
| 1.2 | Objectives and Requirements | 2 |
| 1.2.1 | Design Objectives | 2 |
| 1.2.2 | System Requirements | 2 |
| 1.3 | Motivations | 3 |
| 1.4 | Related Works | 4 |
| 1.5 | Project Plan | 5 |
| 2 | Background | 6 |
| 2.1 | Indoor Localization | 6 |
| 2.1.1 | Indoor Environment | 6 |
| 2.1.2 | Localization | 8 |
| 2.2 | Artificial Neural Networks | 10 |
| 2.2.1 | Historical Background | 10 |
| 2.2.2 | Biological Neural Networks | 10 |
| 2.2.3 | ANN Terminology | 12 |
| 2.3 | ZigBee Technology | 16 |
| 2.3.1 | ZigBee Stack | 16 |
| 2.3.2 | ZigBee and other competitors | 21 |
| 2.4 | Security Algorithms | 21 |
| 2.4.1 | Advanced Encryption Standard | 22 |
| 2.4.2 | The Secure Hash Algorithm | 24 |
| 2.5 | Software Development Tools | 26 |
| 2.5.1 | C# Programming Language | 26 |
| 2.5.2 | Database | 28 |
| 3 | System Design | 30 |
| 3.1 | Problem Formulation and Modeling | 31 |
| 3.1.1 | Mobile node | 31 |
| 3.1.2 | Anchor Network | 34 |
| 3.1.3 | The Server | 39 |
| 3.2 | Software | 42 |

| | | |
|----------|---|------------|
| 3.2.1 | System users | 43 |
| 3.2.2 | System security | 45 |
| 3.2.3 | Database | 49 |
| 4 | Hardware Implementation | 53 |
| 4.1 | Detailed Circuit Design | 53 |
| 4.2 | Some Modifications on the Localization Algorithm due to Hardware Issues . . . | 58 |
| 4.3 | RSSI Collection (Fingerprinting stage) | 61 |
| 5 | Software Implementation | 66 |
| 5.1 | The main software - GUI | 66 |
| 5.2 | Database and Logger files | 79 |
| 5.3 | MATLAB code | 80 |
| 5.4 | Embedded C Code for Mobile and Fixed Nodes | 84 |
| 5.5 | The Central Localization Engin | 90 |
| 6 | Testing and System's Performance | 93 |
| 6.1 | Test The main Software | 93 |
| 6.2 | Fingerprinting And RSSI Visualization | 97 |
| 6.3 | Neural Network Performance | 103 |
| 7 | Challenges and Future Works | 106 |
| 7.1 | Main challenges | 106 |
| 7.2 | Future Works | 107 |

List of Figures

| | | |
|------|--|----|
| 1.1 | Project time table for this semester. | 5 |
| 2.1 | Fading effects on RSS | 8 |
| 2.2 | General taxonomy of localization techniques. | 9 |
| 2.3 | A biological neuron | 11 |
| 2.4 | A single artificial neuron | 12 |
| 2.5 | A Feed-Forward neural network | 14 |
| 2.6 | Overfitting in supervised learning | 16 |
| 2.7 | Stack of ZigBee protocol. | 17 |
| 2.8 | IEEE802.15.4 Frame format for PHY and MAC layers. | 17 |
| 2.9 | Typical ZigBee network. | 18 |
| 2.10 | AODV routing algorithm. | 20 |
| 2.11 | The overall structure of AES algorithm [26]. | 23 |
| 2.12 | AES encryption round [26]. | 24 |
| 2.13 | The secure hash algorithm [26]. | 25 |
| 2.14 | Illustration shows how powerful hashing algorithms are [27]. | 26 |
| 2.15 | Illustration shows randomized hashing algorithms [27]. | 26 |
| 2.16 | Simplified block diagram of DBMS architecture. | 29 |
| 3.1 | General Block Diagram | 32 |
| 3.2 | XBee-PRO DigiMesh 2.4 module. | 32 |
| 3.3 | The synchronization behavior of sleep compatible nodes. | 35 |
| 3.4 | Localization and tracking procedure. | 37 |
| 3.5 | The One-Hop Concept | 38 |
| 3.6 | Messages exchanged in the L&TP | 39 |
| 3.7 | Illustration: MN's location affect k | 40 |
| 3.8 | The Localization and tracking algorithm. | 42 |
| 3.9 | The graphical user interface of the software. | 43 |
| 3.10 | Software - admin interaction flowchart. | 44 |
| 3.11 | Software - doctor interaction flowchart. | 46 |
| 3.12 | Login flowchart. | 47 |
| 3.13 | The procedure of joining the network. | 48 |
| 3.14 | ER model for database used to train the neural networks. | 50 |
| 3.15 | ER model for system database. | 51 |

| | | |
|------|--|-----|
| 3.16 | Normalized logical models | 52 |
| 4.1 | Coordinator node. | 54 |
| 4.2 | Circuit diagram of the Fixed Node. | 55 |
| 4.3 | Fixed node implementation. | 58 |
| 4.4 | Two Directional Sensors. | 60 |
| 4.5 | Two ADXL335 readings. | 60 |
| 4.6 | Fingerprinting process on the 6th floor of B-Building of Engineering Department. | 65 |
| 5.1 | The main window of the monitoring system. | 67 |
| 5.2 | The login window. | 69 |
| 5.3 | Add patient form | 71 |
| 5.4 | Add doctor form | 73 |
| 5.5 | Nodes form | 73 |
| 5.6 | Patient profile form | 75 |
| 5.7 | Patient notes form | 76 |
| 5.8 | Update patient profile form | 78 |
| 5.9 | Feedforward-Neural Network structure. | 83 |
| 6.1 | A wrong username was entered in the Log in screen. | 94 |
| 6.2 | Main window of the Monitoring System | 94 |
| 6.3 | Strip menu options. | 95 |
| 6.4 | Adding New patient. System will check the data for validity. | 95 |
| 6.5 | Nodes Form. | 96 |
| 6.6 | Attempts to use serial port while coordinator is not connected. | 96 |
| 6.7 | Actions performed during the session are stored in a logger file. | 97 |
| 6.8 | Test drawing functionality. | 98 |
| 6.9 | Mosque RSSI fingerprint. | 99 |
| 6.10 | Required time to complete 1 location update. | 100 |
| 6.11 | The heatmap in the corridor from. | 101 |
| 6.12 | The heatmap room where FN2 is located. | 102 |
| 6.13 | NN performance | 104 |
| 6.14 | Correlation between the target and NN output. | 105 |

List of Tables

| | | |
|-----|---|----|
| 2.1 | Comparison between Wi-Fi, Bluetooth and ZigBee technologies [25]. | 22 |
| 3.1 | Specification of DigiMesh Transceivers | 33 |
| 3.2 | Summary of AT Commands for mobile nodes. | 36 |
| 4.1 | Packets flow to collect RSSI data - Senario 1. | 63 |
| 4.2 | Packets flow to collect RSSI data - Senario 3 (is implemented). | 64 |
| 5.1 | Types of used ZigBee packets. | 85 |

Abbreviations

| | |
|---------|--|
| AES | Advanced Encryption Standard |
| AN | Anchor Node |
| ANN | Artificial Neural Networks |
| AOA | Angle of Arrival |
| CRC | Cyclic Redundancy Check |
| CSMA/CA | Carrier Sense Multiple Access with Collision Avoidance |
| DBMS | Database Management System |
| DDL | Data Definitions Language |
| DML | Data Manipulation Language |
| DOA | Direction of Arrival |
| DSSS | Direct-Sequence Spread-Spectrum |
| FDMA | Frequency Division Multiple Access |
| FIPS | Federal Information Processing Standard |
| GPS | Global Positioning System |
| GUI | Graphical User Interface |
| ISI | Intersymbol Interference |
| L&TC | Localization and Tracking Command |
| L&TE | Location and Tracking Engine |
| L&TP | Location and Tracking Process |
| LAN | Local Area Network |
| LOS | Line of Sight |
| MAC | Medium Access Control |
| MLP | Multilayer Perceptron |
| MN | Mobile Node |
| NIST | National Institute of Standards and Technology |
| PAN | Personal Area Network |
| RSS | Received Signal Strength |
| SGT | Sleep Guard Time |
| SHA | Secure Hash Algorithm |
| SyncMsg | synchronization message |
| TDOA | Time Difference of Arrival |
| TOA | Time of Arrival |

| | |
|-----|---------------------------|
| WHO | World Health Organization |
| WSN | Wireless Sensor Network |
| WW | Wire Wrapping |
| ZC | ZigBee Coordinator |
| ZED | ZigBee End Device |
| ZR | ZigBee Router |

Chapter 1

Introduction

1.1 Overview

People with intellectual disabilities are known to have limited learning abilities and defects in adaptive behavior or daily-life skills (self-care, communication, community participation, etc.). An intellectual disability may result because of brain damage or birth defects; however social, environmental, medical and cultural deprivation are important contributors of which they can be prevented. According to World Health Organization (WHO), 2-3% of the total world population have intellectual disabilities - with different levels (ranging from mild to profound). Given this fact, what can be done in providing care for these individuals? [1]

The bad news is that mental retardation is generally a life-long condition and it cannot be 'cured' with medical treatment. However, this does not mean excluding retarded people from the society under the pretext of saving them from hurting themselves. Studies have shown that with the right kind of support and assistance, most of those people will be able to live relatively independent and fulfill their community roles. With this point in mind, our idea came to design an indoor monitoring system that will constantly track the retarded patients without annoying them or violating their privacy.

The main target for our monitoring system is to determine patients' locations within their local environment. The idea behind that is if we know where the patient is, we can expect what he's doing. Thus, we can estimate his situation and take proper actions if the patient is in improper place, or if the situation becomes suspicious (e.g. being in the same place for long time). The system will track each patient within the area of interest. When the carer ask to know where the patients are, their locations will be displayed on a pre-installed map for the building with some relative information about each patient (such as name, age, personal picture, type of his disability, ...).

In its general form, the system will be based on ZigBee technology to carry out all needed communications. Each patient will carry a small transceiver wherever he goes. Other transceiver

will distributed in the area at fixed and known locations to serve as reference nodes. The care-provider will be able to interact with the system from his computer. When the patient's location is needed, the target transceiver will broadcast a message to its neighbors fixed node. The received signal strength (RSS) values at these nodes will be measured and then delivered to the computer, where the processing will take place, and the estimated location will be displayed on the map. Additional features will be discussed later.

1.2 Objectives and Requirements

1.2.1 Design Objectives

The proposed project is aim to:

1. Design and build an indoor monitoring system specially for treatment centers that take care of people with intellectual disabilities.
2. Introduce an innovation monitoring way which is based on location determination to overcome limitations of traditional monitoring systems that use cameras.
3. Try a new method for indoor localization based on smart and adaptive neural networks algorithms

1.2.2 System Requirements

The proposed project should be able to:

1. Constantly monitor the patients within the treatment center. This will be accomplished by identifying and locating each target within the covered area at any time.
2. Have a GUI that is easy for user to interact with. We will use preinstalled maps for the site to show estimate locations of the targets in a neat and clear way. These maps will be updated and presented upon user's request.
3. Efficiently store the collected data (Patient name, ID, picture, illness type; location, ...), so that the data can be used during the treatment process to study patient's statistics and psychological progress.
4. Guarantee that all patients are within the allowed area. This can be done by notifying the staff about any attempt to "escape" in order to take proper actions.
5. Discover and report any failure, in order to ensure that the provided location data are accurate.

In order to achieve the aforementioned objectives and requirements we need the following hardware and software components:

- ZigBee Transceivers: for wireless communication.
- PIC Microcontrollers: to control used devices.
- GPS Device: to enable the outdoor monitoring. (Optional)
- Sensors: for patient's health monitoring. (Optional)
- Server: to store data and run the software.
- C# Programming Language: for graphical user interface and the database.
- MATLAB: for the Artificial Neural Networks implementation.
- C Programming Language: for PIC Microcontrollers.
- L^AT_EX for writing the documentation.

1.3 Motivations

The idea of this project has its importance due to our social responsibilities toward people with intellectual disabilities. Those individuals have limited capability to think and control their actions. However, they still can move easily and don't stick to their places where they are assumed to be. Thus, there is a substantial need to be aware of their locations to guarantee their safety. Our system supports care-providers with a powerful tool to constantly monitor large number of patients without the need to be with them all the time. This will save the staff's efforts, and will provide the patients with higher level of comfort and privacy.

The Need for our project has arisen because of the inefficiency of the global positioning system (GPS) inside buildings [2]. Specialists know that indoor localization is a big challenge due to complex nature of the indoor environment. Wireless signals suffer from fading, scattering, distortion and other effects that severely degrade the quality of the received signal. Our system will overcome these limitations with high accuracy and low implementation costs.

Wireless Sensor Network (WSN) has been chosen (in particular ZigBee chips) because of its small size, low cost and very low power consumption. On the other hand, there is a growing trend towards exploiting WSN in *smart building's* applications. This means that the needed infrastructure for our system will be available and this will decrease the implementation cost dramatically. However, in order to achieve further cost reduction, we will rely on data processing instead of using complex hardware. Neural Networks (NN) will be used to optimize the performance in terms of accuracy and response time.

It is worth to mention that our system has many advantages over traditional monitoring systems that are based on cameras. These advantages include: having lower implementation costs, full coverage over the whole area (including hidden places where cameras can't see, or even private rooms where cameras are not suitable choice), compact data size which allows for long-term data storage, the ability to track each individual patient and others features that are not available in the traditional monitoring systems.

In addition to the aforementioned advantages, our system can be easily expanded to provide additional features and cover very wide range of applications. It can be used for health-care monitoring (by using special sensors on the person's body) to report his status to the care-provider. On the other hand, the system is not limited to the chosen category (retarded patients). It can be used to monitor elderly people who are living alone or in the care homes, children in kindergartens, pilgrims in Hajj seasons and many other scenarios where the location of the users is needed. And if we include the GPS in the game, our system will become a complete solution for localization problem for both indoor and outdoor environments.

1.4 Related Works

Location-based services are one popular branch of context aware computing. They make use of location information about an entity to offer services based on this information [3]. This project focus on exploiting localization and tracking algorithms to build a robust monitoring system dedicated for treatment centers that take care of individuals who suffer from some intellectual disabilities. Many efforts have been proposed in the literature to develop and test different ideas and algorithms to enable localization in wireless sensor networks; and there are many literature surveys on this topic [4], [5], [6].

The first survey classify localization techniques into centralized and decentralized techniques depends on where the localization process is performed (on the mobile nodes or on a central unit). The centralized method is found to be more accurate due to the computational powerfulness of the centralized node which allow to use extensive processing to calculate nodes location [7]. It also save have a better power efficiency since it reduce the amount of power needed to be consumed on the mobile nodes (which are usually powered constrained) [8]. On the other hand, a decentralized algorithms are more reliable and scalable since the location determination is distributed among mobile nodes.

S. Nikitaki propose paper that combines sparse approximation and distributed consensus theory to efficiently perform decentralized localization in wireless networks. In particular, it considers the spatial correlations among the received signal strength measurements at distributed Wi-Fi base stations (BSs) to provide global accurate position estimation, while reducing significantly the amount of measurements exchanged among the BSs. Experimental evaluation with real data achieve positioning accuracy with mean error of 2.5 m [9].

LAURA system [10] is a project dedicated for health care application that performs localization, tracking and monitoring of patients hosted at nursing institutes by exploiting a wireless sensor network based on the IEEE 801.15.4 (ZigBee) standard. They focus on the indoor personal localization module, which leverages a method based on received signal strength measurements, together with a particle filter to perform tracking of both static and moving patients, achieving an average localization error around 0.5 m. However, the particle filter was reported to be very heavy on the mobile nodes as we mention before. For that, LAURA system replace particle filter with a light low pass filter to reduce RSSI variation and achieve mean error lower than 2 m in 80% of the cases.

In his master thesis, Anja Bekkelien performs an evaluation and deployment of existing localization techniques. In particular, he evaluate several different machine learning algorithms for fingerprinting, as well as evaluate the system performance for different scenarios (in Hospitals, Museums and Airports). His indoor positioning system relies on Bluetooth technology, and is integrated into the Global Positioning Module (GPM) developed at the Institute of Services Science at the University of Geneva. The system compares the signal strengths of surrounding Bluetooth devices to a database of measurements taken across the indoor area, in order to estimate the user's position. The deterministic *k-Nearest Neighbor* algorithm is combined with the probabilistic *Nave Bayes classifier* to obtain the target's location with average error of 1.5 meters [11].

1.5 Project Plan

| TASK | WEEKS | | | | | | | | | | | | | | | |
|---------------------------------|-------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Preparing the proposal | | | | | | | | | | | | | | | | |
| A complete study for options | | | | | | | | | | | | | | | | |
| Studying the needed information | | | | | | | | | | | | | | | | |
| Documentation and writing | | | | | | | | | | | | | | | | |

Figure 1.1: Project time table for this semester.

Chapter 2

Background

2.1 Indoor Localization

localization is the process of providing the position information for specific nodes with respect to a set of reference positions within a predefined space [12]. In the last decades, localization and navigation systems have become popular, thanks to the availability of effective and accurate technologies for outdoor positioning such as the GPS. However, the established GPS technology does not work well indoor because of the absence of direct line of sight (LOS) with the satellites, in addition to the large attenuation introduced by buildings' walls and ceilings. Therefore alternative solutions are needed for indoor environments. This section will talk about indoor wireless channel characteristics. Then, some basic concepts related to localization will be presented.

2.1.1 Indoor Environment

A *communication channel* refers to the transmission medium that conveys the information signal. In wireless communication, all the transmission signals share the same medium -air-. Radio wave propagation has been one of the most difficult problems to analyze and design for, since unlike wired communication systems which have a predictable and stationary transmission channel (i.e. a wired path), radio channels are extremely random and don't provide easy analysis, particularly when one of the terminals is in motion. The situation become more complicated when the *indoor environment* is considered. Unlike outdoor areas, the indoor environment imposes different challenges on location discovery due to the dense multipath effects and building material dependent propagation effects. Thus, an indepth understanding of indoor radio propagation for positioning is crucial for efficient design and deployment.

The mechanisms behind the electromagnetic (EM) wave propagation are diverse, but can generally be attributed to *reflection*, *diffraction*, and *scattering*. Reflection occurs when a propagating wave impinges upon an object which has very large dimensions when compared to the wavelength of the propagating wave. While Scattering, which can be defined as the deviation of reflected radiation from the angle predicted by the law of reflection, is resulted from objects

with dimensions that is small compared to the wavelength and when the number of obstacles per unit volume is large. However, diffraction occurs when the propagating wave is obstructed by a surface that has sharp edges, giving rise to a bending of waves around the obstacles at the expense of losing the signal power [13].

Propagation models that are designed to model the effect of the aforementioned phenomena on the RSS over a large transmitter-receiver separation distance (several hundred or thousands of meters) are called *large-scale models*. On the other hand, propagation models that characterize the rapid fluctuations of the RSS over very short travel distances (a few wavelengths) or short time durations (on the order of seconds) are called *small-scale models* or simply *fading models*. Fading models are of more importance in our project because it used to model indoor environments.

Fading, which comprise the biggest problem and the hardest challenge inside buildings, is used to describe the rapid fluctuation of the amplitudes, phases or multipath delays of a radio signal over a short period of time or travel distance, so that large-scale path loss effects can be ignored [14]. Fading is caused by interference between two or more versions of the transmitted signal which arrive at the receiver at slightly different time. These waves, called *multipath waves*, are combine 'destructively' at the receiver antenna leading to a sever loss in the resultant signal power.

Depending on how rapidly the transmitted baseband signal changes as compared to the rate of change of the channel, a channel may be classified as *fast fading* or *slow fading* channel (figure 2.1). In a fast fading channel, the channel impulse response changes rapidly within the symbol duration of the transmitted signal. This causes frequency dispersion due to Doppler spreading which leads to signal distortion. However, in slow fading channels, the channel impulse response changes at a rate much slower than the transmitted baseband signal (i.e. the Doppler spread of the channel is much less than the bandwidth of the baseband signal). In this case, the amplitude and the phase change imposed by the channel can be considered roughly constant over the period of use. The velocity of the mobile (or the velocity of objects in the channel) and the baseband signaling determine which type of fading the signal will undergo.

Analogous types can be defined in the frequency domain according to the *coherence bandwidth* of the channel. The coherence bandwidth measures the separation in frequency after which two signals will experience uncorrelated fading. If the coherence bandwidth of the channel is larger than the bandwidth of the signal, then we have *flat fading* channel. Therefore, all frequency components of the signal will experience the same magnitude of fading. Typical flat fading channels cause deep fade, and thus may require 20 to 30 dB more transmitter power to achieve low bit error rate during times if deep fades as compared to systems operate over non-fading channels [15]. On the other hand, in *frequency-selective fading*, the coherence bandwidth of the channel is much smaller than the bandwidth of the signal. Different frequency components of the signal therefore experience decorrelated fading. Frequency selective fading is due to

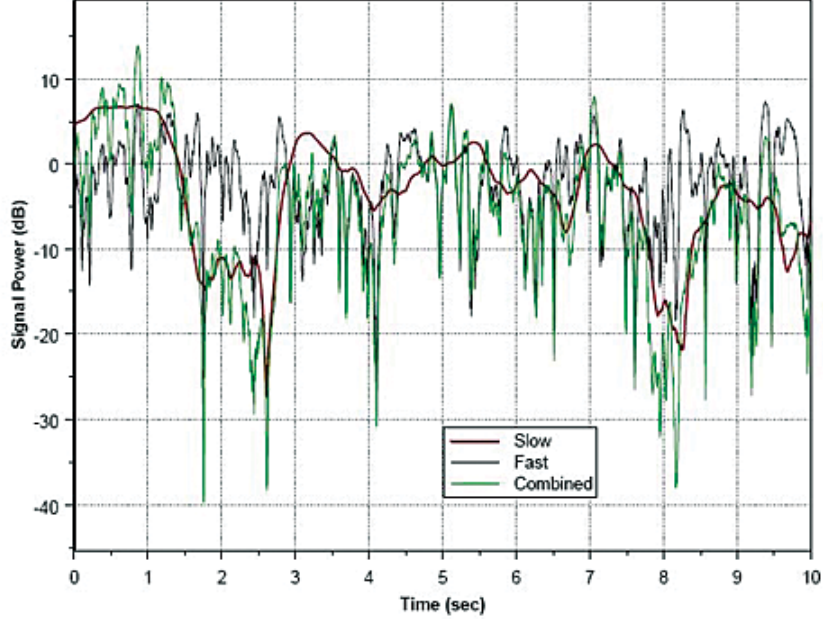


Figure 2.1: Fast fading, slow fading, and their effect on the received signal power.

time dispersion of the transmitted symbol within the channel which leads to the Intersymbol Interference (ISI) problem.

2.1.2 Localization

Localization is a cornerstone for many applications that require a prior knowledge of the user's location, so there is a real need to have a robust indoor localization system which provides high accuracy with low implementation cost. The need for such systems has conducted large body of researches and many algorithms have been developed over recent years.

localization techniques can be categorized into different families depending on several factors that include: technology (ZigBee, RFID, Wi-Fi or Bluetooth), algorithm (trilateration, triangulation, fingerprinting), or even the physical quantity that varies with the position (radio frequency waves, photonic energy, sonic waves, mechanical energy (inertial or contact) [16]. Figure 2.2 shows general taxonomy of localization techniques. We will concentrate on the radio frequency techniques, in particular those which are used in personal and local area networks.

- **Time of arrival (TOA):** It is non-fingerprinting technique based on the measurement of the arrival time of a signal transmitted from a mobile device to several receiving sensors. Signals travel with a known velocity (approximately the speed of light), the distance can be determined from the propagation time of the received signal. This is a simple and cheap technique, but it needs clear LOS between the and precise synchronization between the

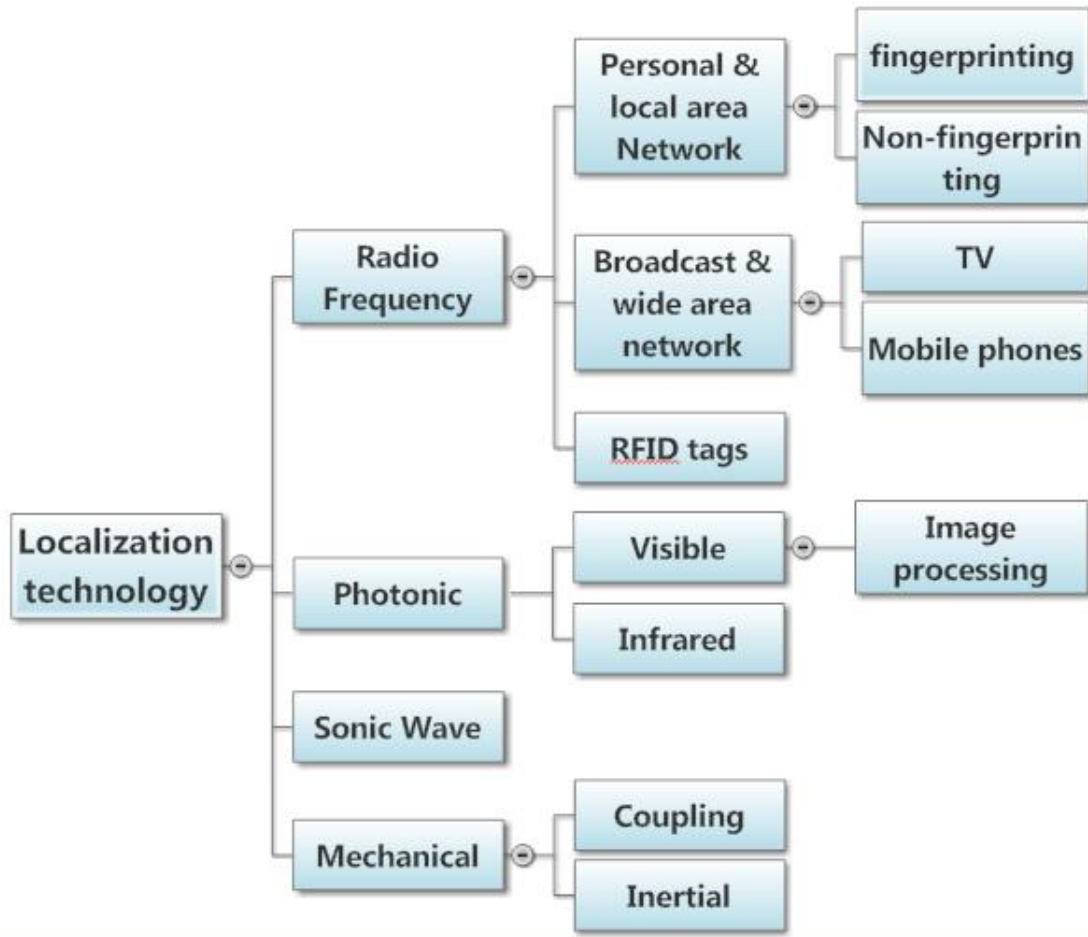


Figure 2.2: General taxonomy of localization techniques.

transmitter and the receiver. Synchronization problem can be solved using the relative time measurements of two signals or more at each receiving sensor. This technique is known as **Time Difference of Arrival (TDoA)**.

- **Angle of Arrival (AoA)**: it locates the position of the target by determining the angle of incidence at which signals arrive at the receiving sensor. This technique gives good results, but has a high cost due to the complexity of the system. **Direction of arrival (DOA)** technique use similar principle, the intersection of several measured directions pointers yields to the position value of the target. These techniques are also non-fingerprinting techniques.
- **Fingerprinting**: Fingerprinting requires a detailed signal strength database for each reference point that can be compared with received signal strength in the field. It can be generally divided into two phases: an offline phase and an online phase. The offline phase involves spreading certain number of fixed nodes with known locations called *reference*

nodes in the experimental area, then the received signal strength from these reference nodes is measured and stored in a database. During the online phase, the RSSI value of the target node is measured and compared to the database to determine the target location.

2.2 Artificial Neural Networks

Artificial Neural Networks (ANN) are parallel computational models, comprised of densely interconnected adaptive processing units. These networks are "neural" in the sense that they have been inspired by neuroscience. A very important feature of these networks is their adaptive nature to "learn". This feature makes such computational models very appealing in application domains where one has little or incomplete understanding of the problem to be solved, but where training data is available.

2.2.1 Historical Background

ANN begins in the early 1940's and thus nearly in parallel to the programmable electronic computers. Since then, many efforts were proposed such as the formulation of the classical Hebbian rule in 1949 [18]. The first adaptive learning system followed and became widely commercially used in 1960 in nearly every analog telephone for real-time adaptive echo filtering. Progress continued until 1969, where Marvin Minsky published a precise mathematical analysis of the perceptron [19] to show that the model - which was considered as cornerstone of ANN - was not capable of representing many important problems, and so put an end to overestimation, popularity and research funds for that field for more than 15 years.

However, a learning procedure called "backpropagation of error" was introduced in 1974, and after a decade it gained researchers' attention. Moreover, non-linearly separable problems could be solved by multilayer perceptron (MLP), and Marvin Minskys negative evaluations were disproven at a single blow. The field has survived again, the IEEE First Annual International Conference on Neural Networks was held in 1988 in San Diego, California. And now ANN find a wide range of applications in different science and engineering domains.

2.2.2 Biological Neural Networks

The nervous system is a complex network of nerves and cells that are capable of carrying messages between the central and the peripheral nervous systems. The brain is the place where the learning process occurs. It consists of billions of simple processing units (called *neurons*) that communicate and cooperate with each other to perform very complex tasks we are familiar with from our brain.

As shown in figure 2.3, a typical neuron has a cell body and long arms that conduct impulses from one body part to another. The cell body has several thick and highly branched extensions

called *dendrites*. Their function is to carry a nerve impulse into the cell body through special connections called *synapses*. The output nerve impulse is carried by an *axon* from the cell body to its terminals to be sent to another neuron or tissue. The axon is isolated by *myelin sheath* in order to achieve better conduction of the electrical signal.

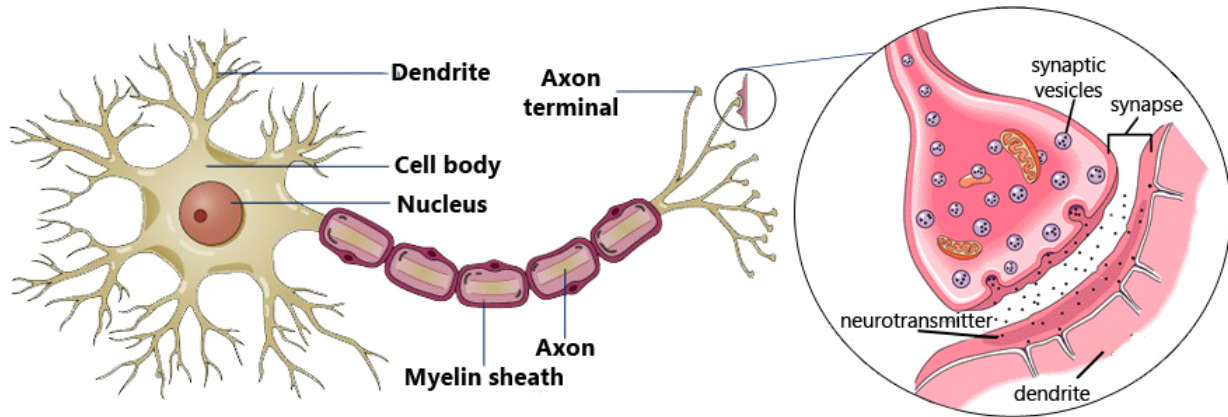


Figure 2.3: Illustration of a biological neuron with the components discussed in this text.

In order for the communication to begin, the accumulated received stimulus should exceed a certain *threshold*. If so, an action potential will be conducting near the cell body portion of the axon - based on the differences in concentration of sodium and potassium ions inside and outside the cell membrane. The axon carries the generated electrical signal to the synapses. The synapse is the area of contact between two neurons (neurons do not physically touch because they are separated by a *cleft*). *Neurotransmitters* are the chemical medium through which signals flow from one neuron to the next, where the transmitted signal strength is controlled (*weighted*) by the amount of neurotransmitters being used.

It is interesting to have a brief overview of different organisms and the number of neurons they have. Starting from 10^4 neurons, we will have something as simple as an *ant*. With 10^5 neurons the nervous system of a *fly* can be constructed. A fly can evade from any object in real-time in three dimensional space, we all know that a fly is not easy to catch! 10^7 neurons are sufficient for a *rat*, an animal which is denounced as being extremely intelligent. However, *cats* known to have around 10^8 neurons, a good advantage for them against their historical enemy. *Chimpanzees* have $6 * 10^9$ neurons, while 10^{11} neurons make a *human*. Surprisingly, there are nervous systems having more neurons than the human nervous system; *Elephants* have around $2 * 10^{11}$ neurons! [19]

It is worth to mention that our state-of-the-art computers are not able to keep up with the aforementioned processing power of a fly! Allah has told us this in the Quran in Surat Al-Haj [verse 73], he says: "O people, an example is presented, so listen to it. Indeed, those you invoke

besides Allah will never create [as much as] a fly, even if they gathered together for that purpose. And if the fly should steal away from them a [tiny] thing, they could not recover it from him. Weak are the pursuer and pursued”.

2.2.3 ANN Terminology

The field of ANN has evolved independently in and among many communities such as mathematics, computer science (artificial intelligence), physics and psychology. As a result, a variegated terminology and synonymic nomenclature has been developed [20]. So in this section, we will try to provide the reader with brief background about ANN and specify all notations that will be used through this project.

Artificial Neuron and Artificial Network

The terminology of ANN has developed from the biological model of the brain. ANN are nonlinear, data driven, self adaptive, parallel computational models comprised of interconnected processing units called *neurons*. These networks have the ability to identify and learn correlated patterns between input data set (denoted by the vector \mathbf{x}) and the corresponding target outputs (denoted by the vector \mathbf{y}) through a process called *training*. After training, ANN can be used to predict the outcome of new independent input data.

An artificial neuron is the basic element of a neural network. Neurons are grouped in the network in layered form. As shown in figure 2.4, neuron consists of three basic components that include *weights* (denoted by the vector \mathbf{w}), *threshold* (w_0), and a single *activation function* ($\varphi(\cdot)$).

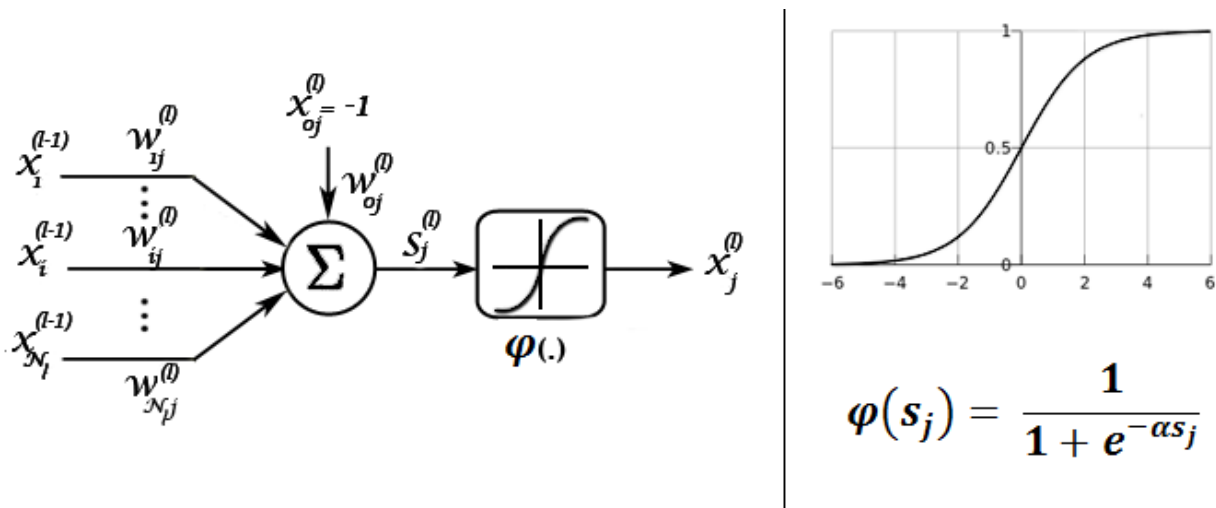


Figure 2.4: A single artificial neuron with 'S-Shape' activation function.

We'll define $w_{ij}^{(l)}$ as the weight factor associated with the neuron j in the layer l , to determine the strength of the input that comes from the neuron i . The Weights and the threshold can be fixed or adapted using a number of different algorithms (learning rules) during the training phase in order to minimize the error function. Different neurons are characterized by different activation functions ($\varphi(.)$). The activation function reflects the nonlinearity of the neuron that allows it to learn relations between nonlinear patterns. *tanh(.)* and *S-shape* function are commonly used because they are differentiable, unlike *sign(.)*. This is important requirement for many optimization algorithms such as the well-known *delta rule*.

An artificial neuron mimics the processing of a biological neuron. Information is received from many input sources and summed. If the summed total of the input is at or beyond the active threshold; i.e. for N_l inputs if

$$s_j^{(l)} = \sum_{i=0}^{N_l} (w_{ij}^{(l)} . x_i^{(l-1)}) > 0, \quad x_0 = -1$$

then the neuron will be activated and the output $x_j^{(l)} = \varphi(s_j^{(l)})$ will be transmitted to feed other neurons in the next layer in the network.

The way in which neurons are interconnected determines the type of neural network's topology. There are several types of ANN, however, the two most widely used are *feed forward networks* and *recurrent networks*. The neuron layers of a feed-forward network (figure 2.5) are clearly separated: one input layer, one output layer and one or more processing layers which are invisible from the outside (also called hidden layers). The input layer is not really neural at all: these units simply serve to introduce the values of the input variables (which are RSSI values in our project). The hidden and output layer neurons are each connected to all of the units in the preceding layer. In fact, it is possible to define networks that are *partially-connected* to only some units in the preceding layer; however, for most applications *fully-connected* networks are better.

Unlike recurrent neural networks, feed-forward networks only permit connections to neurons of the following layer. There is no feedback (loops) i.e., the output of any layer does not affect that same or preceding layer. However, in recurrent networks a neuron can affect itself directly by self-feedback (*direct recurrent networks*), or indirectly by affecting other neurons in the preceding layer, which in turn will affect it (*indirect recurrent networks*). The presence of feedback in recurrent networks can make it unstable and has very complex dynamics. In the feed-forward networks signals flow from inputs, forwards through any hidden units, eventually reaching the output units. Such a structure has the advantage of being stable. In fact, recurrent networks are very interesting to researchers in neural networks, but so far it is the feed-forward structures that have proved most useful in solving real problems.

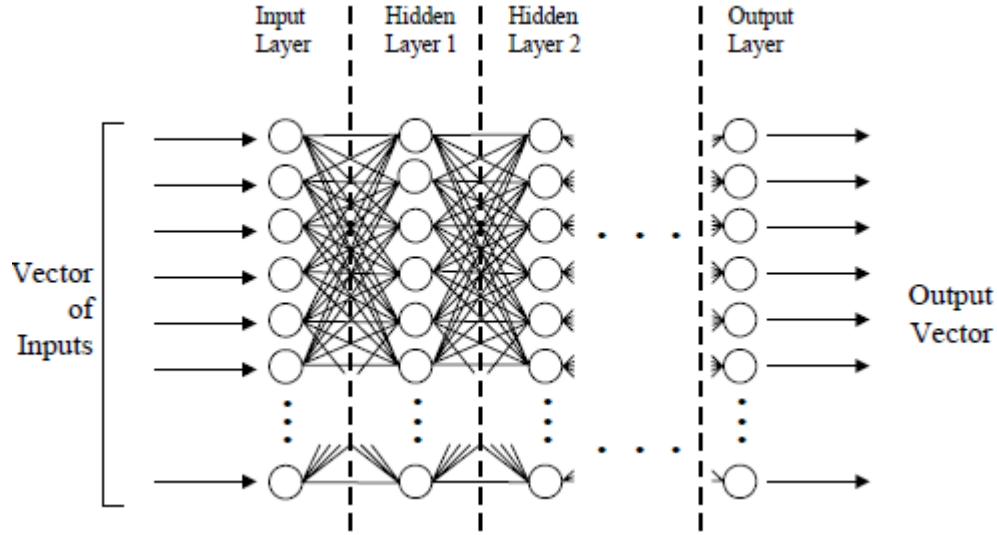


Figure 2.5: A typical Feed-Forward neural network with 'L' layers.

The Learning Principle

As mentioned before, the most interesting characteristic of neural networks is their capability to learn. In fact, 'learning' is a comprehensive term and always there will be a question on how to implement it. In principle, a neural network changes when its components are changing. Thus, we let our neural network learn by modifying the connecting weights according to rules that can be formulated as algorithms, which can easily be implemented by means of a programming language. The training algorithm of the network determines neurons' types, network topology and which error criterion to be used.

First of all, let us define the *training set* to be a subset of the data which the ANN uses to learn the relationships between input and output response. We previously defined the input of the neural network to be the input of the first layer ($\mathbf{x}^{(0)}$ or simply \mathbf{x}) and the output of the neural network to be the output of the last layer ($\mathbf{x}^{(L)}$ or simply \mathbf{y}). We will use different notation to describe the training set; namely the training inputs will be denoted by \mathbf{p} and target output will be denoted by \mathbf{t} . The goal is to learn something from the training set that can be used to predict the output of unknown inputs of the same class, this approach is referred to as *generalization*.

There are essentially two types of ANN learning models *supervised learning* and *unsupervised learning*. With supervised learning, for each input pattern in the training set the value of the desired target output is specified. The goal of is to find the mapping between training inputs and target outputs and modify the weights to reflect that relation. This is done by applying the training input vector to the network, then the output of the network is calculated and compared to the corresponding target vector with the difference (error) being fed back through the network

to change the weights so that the error function is minimized. The *perceptron learning rule*, the *delta rule* and *backpropagation* are examples on this type of learning (backpropagation is considered to be most powerful among them).

In unsupervised learning, however, the target output is not known and the training set contains only the input data. Instead of trying to map the data input-output relationship, the goal here is to find an underlying structure of the data, called *clusters*. Mainly two categories of learning rules fall under this heading: *Hebbian learning* and *competitive learning* (such as *Kohonen networks*). While unsupervised learning has some advantages and theoretically shows potential for solving difficult problems, it has remained largely unsuccessful in solving many engineering problems. As a result, it does not yet have widespread engineering applicability.

The important point we should always remember is that the target output is always unknown, all what we know is its value at some points (training set). Neural network will try to *fit* these samples to approximate the target function for any point in general. The only indicator about how well we are doing is the *in-sample error*, i.e. the error function between the calculated output and the target output for the training input inside the training set ($\mathbf{E}_{in} = \mathbf{t} - \mathbf{y}$). However, we have no idea about how well the error is outside this training set (\mathbf{E}_{out}). i.e. the prediction error. In order to solve this problem, we don't use all our data for training. Instead, the data is divided into three independent groups: *training set*, *validation set* and *testing set*. We train the network using the training set according to \mathbf{E}_{in} . Meanwhile, we provide the network with the input vector of the validation set while keeping the target output hidden, and ask the network to guess the output then we observe \mathbf{E}_{out} . Figure(2.6) show typical situation.

As we see in the left-hand side diagram, the network start learning and the error start falling, but the approximation is still not so good (*underfitting* state). With each iteration in the training algorithm (epoch), the network continue adjusting the weights to fit the data and reduce \mathbf{E}_{in} with no idea about \mathbf{E}_{out} . After a certain point, the training set performance continues to improve, while the validation set performance no longer does. The data are well described, but the predictions do not generalize to new data outside the study sample, this is because the network start to fit the *noise* in the data rather than the data itself (this problem is referred to as *overfitting* - right-hand side graph). One direct solution is to stop the learning process early.

Note that the validation set is used in the training procedure to determine when to stop the training, so there is need to another set for testing the network after the training is complete. This set is what remains from our data which is the *testing set*. It is used in the same way as validation set, the inputs are applied to the network, and the output is calculated and compared with the target output, and the error is calculated. If it is acceptable, then the networks is ready to be used.

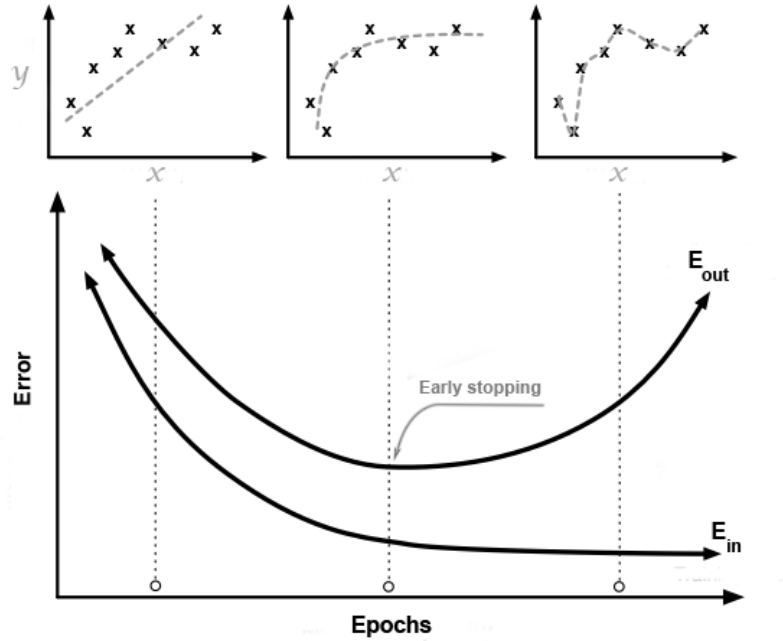


Figure 2.6: Overfitting in supervised learning

2.3 ZigBee Technology

ZigBee is a new personal-area network (PAN) technology developed by the ZigBee Alliance based on the IEEE802.15.4 standard. Its mission is to define a reliable, cost-effective, secured wireless communication, using low power consumption with the ability to operate for months or even years [22].

2.3.1 ZigBee Stack

ZigBee protocol stack is shown in figure 2.3. It extends the IEEE802.15.4 standard with two layers (Network layer and Application layer) to support additional features such as routing reliability and network security and many other features.

IEEE 802.15.4 standard defines two layers which are the physical layer (PHY) and the medium access control (MAC) layer [23]. The PHY layer is concerned with the interface to the physical transmission medium (wireless channel) as well as exchanging data bits with the upper layer (MAC layer). The PHY layer specifies 27 channels distributed along three license-free frequency bands as follows: 16 channels at 2.4 GHz with data rates of 250 kbps, 10 channels at 902 to 928 MHz with data rates of 40 kbps and one channel at 868 to 870 MHz with a data rate of 20 kbps. Only the 2.4-GHz band operates worldwide, the others are regional bands.

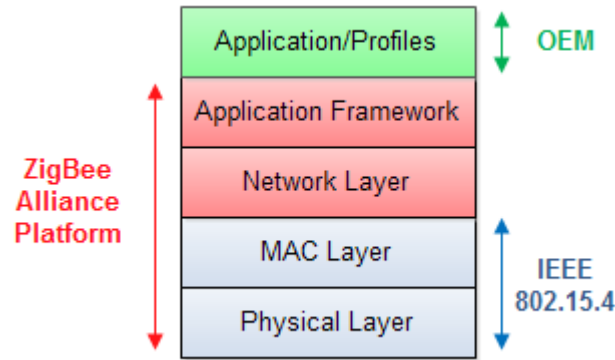


Figure 2.7: Stack of ZigBee protocol.

The MAC layer is responsible for providing reliable communications between a node and its immediate neighbors, helping to avoid collisions and improve efficiency. The MAC Layer is also responsible for assembling, and decomposing data packets and frames. MAC layer defines four frame structures: a *beacon frame* which is used by a coordinator to transmit beacons, *data frame* which is used for all data transfer, *acknowledgment frame* which is used for confirming successful frame reception and the *MAC command frame* which is used for handling all MAC peer entity control transfers. Figure 2.8 shown the data frame.

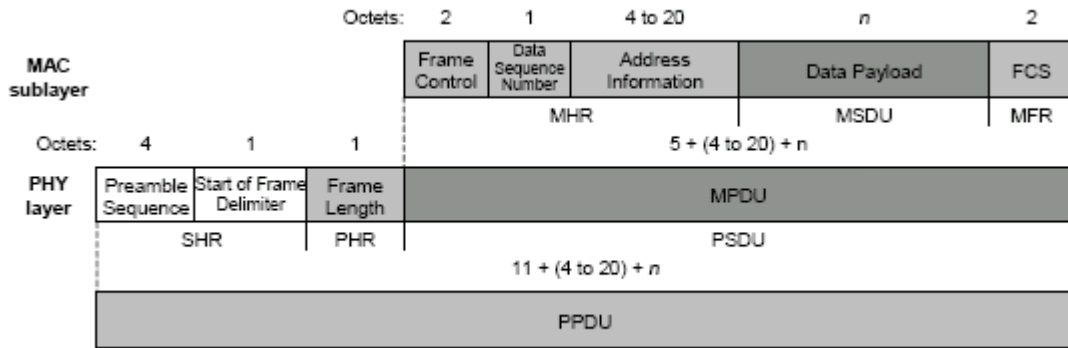


Figure 2.8: IEEE802.15.4 Frame format for PHY and MAC layers.

IEEE 802.15.4 has adopted the direct-sequence spread-spectrum (DSSS) technique in order to ensure coexistence and robustness against interference. Another technique is frequency division multiple access (FDMA), allowing devices operating in adjacent channels to coexist without problems. In addition to the techniques described previously, Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is also necessary in most networks without beacons because several devices may be working in the same channel.

ZigBee networks include three types of devices defined according to their role in the network: *ZigBee Coordinator* (ZC) which is responsible for creating and maintaining the network, *ZigBee Router* (ZR) that interconnects the nodes in the network and responsible of routing mechanism, and *Zigbee End Device* (ZED) which has limited functionality and is required to have a parent (ZC or ZR) to connect it to the network. his kind of nodes are usually powered-constrained so it sleep most of the time to save its power.

The way at which nodes are interconnected defines the network topology. ZigBee allows three different topologies which are *star*, *tree* and *mesh* topologies. In the star topology the network has a central node (coordinator), which is linked to all other nodes in the network and serve as their parent. All messages that are exchanged are traveling through this central node. Tree topology has a root that serve as a parent node for nodes in the lower level. Nodes in the second level can have children if they are full functional nodes (ZC or ZRs). To reach its destination, a message travels up the tree from its source until it reach the first common parent with its destination and then goes down to the destination. Star and Tree networks define single path between a source and its destination. This means that if the parent is off for some resone, its children will lost their connection to the network.

On the other hand, nodes that form mesh topology are interconnected with each other so that multiple pathways are available. Connections between nodes are dynamically updated and optimized through sophisticated, built-in mesh routing tables. Mesh networks are decentralized in nature; each node is capable of self-discovery on the network. If one node leaves the network, the mesh topology allows the others to reconfigure their routing paths based on the new network structure. The characteristics of mesh topology and ad-hoc routing provide greater stability in changing conditions or failure at single nodes. Figure 2.9 shows a typical hybrid network.

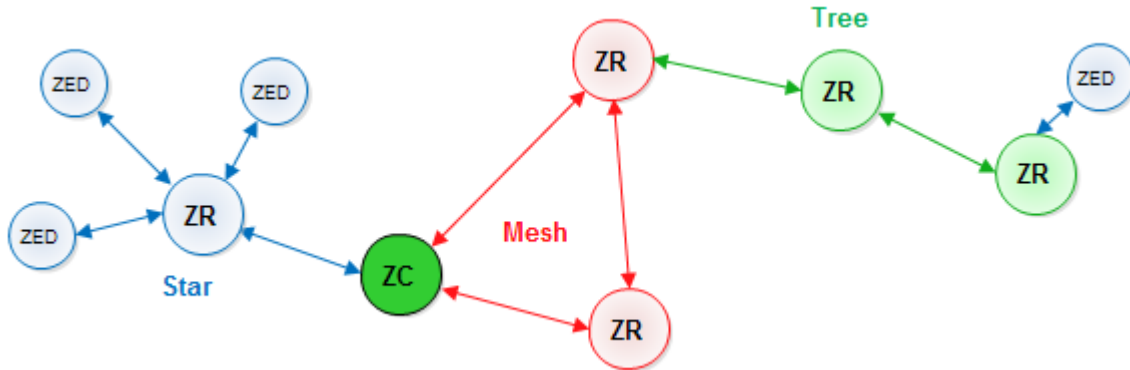


Figure 2.9: Typical ZigBee network.

In order to ensure reliable communication, ZigBee employs a range of techniques that helps a message reaches its destination without being corrupted. A Channel Selection is performed

when a new ZigBee network is created to avoid the interference from other networks. Medium access control algorithms are used to avoid conflicting transmissions. Data Coding is used to enable error detection together with active acknowledgement mechanism that corrects any discovered error by asking the source to resend the corrupted message again. The network has also a built-in intelligent routing discovery algorithm to ensure that messages reach their destinations using a default route (short path), or alternative routes if the default route becomes down; due to a failed intermediate node or link.

Routing algorithms in the ZigBee networks can be divided into two categories: The first is the *Hierarchical Routing Algorithms* (HERA), that can be implemented in a parent-child networks, where the relationship between coordinator or routers and end nodes as defined in the tree topology. The second is the *Ad Hoc On-Demand Distance Vector* (AODV) algorithm [24] that is suitable for mesh networks in order to allow direct communications among network nodes through multi-hop paths. In this project we will focus on the second type.

AODV is a routing protocol which bases route discovery on a *route request* (RREQ) and *route reply* (RREP) query cycle. The primary objectives of the algorithm are to broadcast discovery packets only when necessary, to distinguish between local connectivity management and general topology maintenance and to disseminate information about changes in local connectivity to those neighboring mobile nodes that are likely to need the information. Its metric is based on the number of hops from the source to the destination. When a source node aims to send a message to a destination node, the source broadcast a RREQ message to discover a route to the destination. Intermediate nodes will forward the RREQ and any node knows the route to the destination (or it is the destination itself) will reply with a RREP message to the original source. When the source receive RREP message, it becomes ready to send data to the destination. If any error occurs during the route valid time, a route error message (RRER) will be sent back to the original source. AODV algorithm is explained in figure 2.10.

Whenever an AODV router receives RREQ message, it checks its routing table to see if it knows the route to the required destination. Each entry in the routing table consists of the destination address, next hop address, destination sequence number and hop counts. If the route exists, the router will simply forward the message to the next hop. Otherwise, it saves the message in a message queue and initiates a route request to discover the route. In order to prevent cycles, each node remembers recently forwarded route requests in a route request buffer. As these requests spread through the network, intermediate nodes store reverse routes back to the original node. Since an intermediate node could have many reverse routes, it always picks the route with the smallest hop count to send its RREP message (when the route to destination become known). As the RREP message passes through intermediate nodes, they update their routing tables.

Because it is possible for the RREQ originator to receive a RREP message from more than one node, it will update its routing table with the most recent routing information. Each node

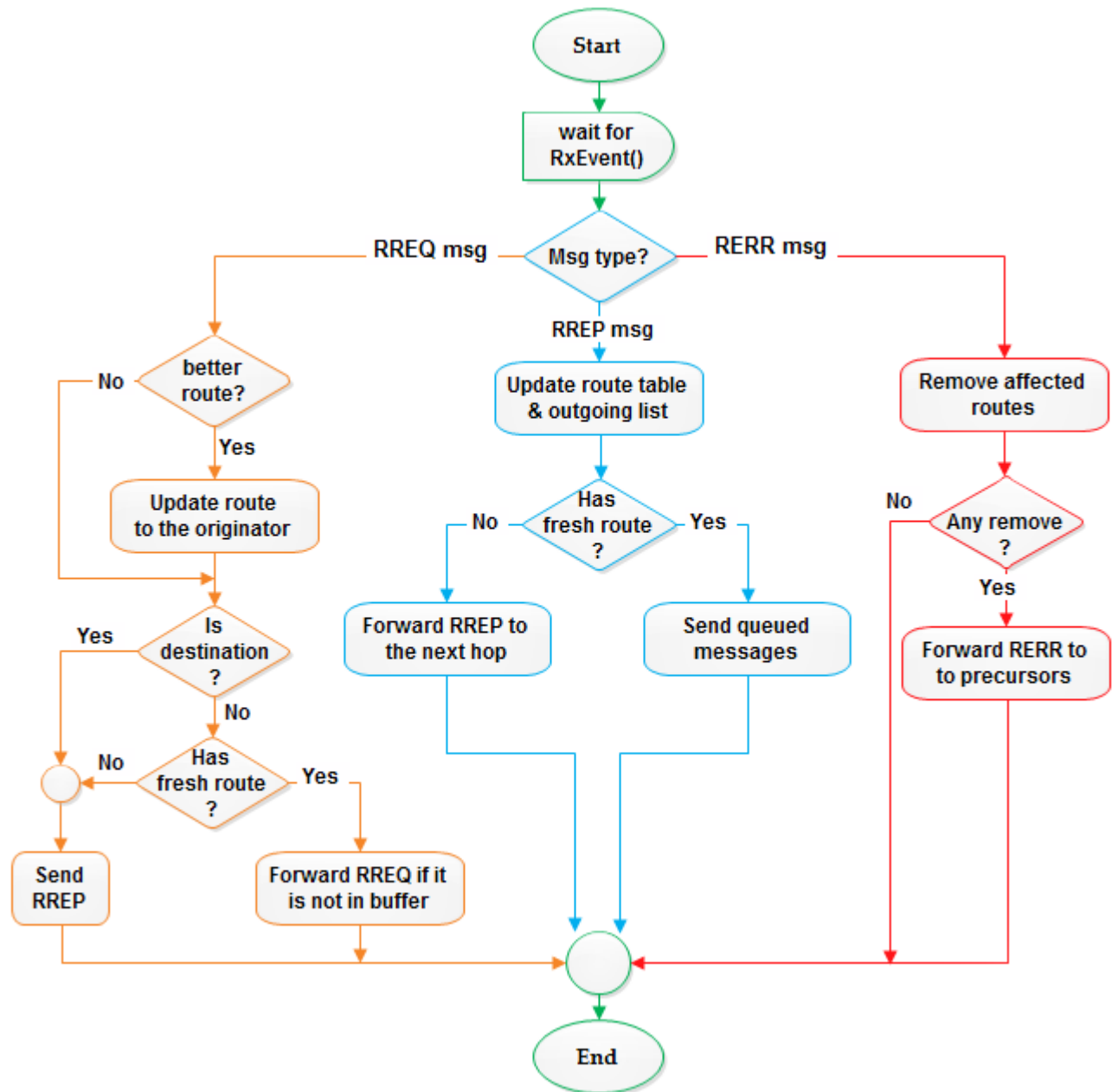


Figure 2.10: AODV routing algorithm.

keeps track of a *precursor list*, and an *outgoing list*. A precursor list is a set of nodes that route through the given node. The outgoing list is the set of next hops that this node routes through. Each node periodically sends HELLO messages to its precursors. A node decides to send a HELLO message to a given precursor only if no message has been sent to that precursor recently.

Whenever a node determines one of its next hops to be unreachable, it removes all affected route entries and generates a RRER message. This RRER message contains a list of all destinations that have become unreachable as a result of the broken link. The node sends the RRER to each of its precursors. These precursors update their routing tables, and in turn forward the RRER to their precursors, and so on. To prevent RRER message loops, a node only forwards a RRER message if at least one route has been removed.

2.3.2 ZigBee and other competitors

Wireless sensor networks are one of the most rapidly growing technologies and has a wide variety of applications. As a wireless technology, ZigBee becomes popular in the recent years due to its ultra-low power consumption. However, there are many wireless technologies that are competing with the ZigBee such as Wi-Fi and Bluetooth. Wi-Fi is based on the IEEE 802.11 standards and considered as the most prevalent technology in the world. Most researchers prefer to exploit Wi-Fi technology in their projects due its availability in almost everywhere, and in every wireless device (such as laptops, smart phones, tablets and so on). A typical wireless router uses 802.11b or 802.11g with a stock antenna might have a communication range of 32 m indoors and 95 m outdoors. Wi-Fi transceivers are very suitable for applications that requires high data rate and/or an internet connection.

On the other hand, Bluetooth is primarily designed for low power consumption and short ranges communication (1m and 10m). Bluetooth is not targeted for a specific application, but supports a multitude of applications. Therefore the technology has been adopted by wide variety of devices, including computers, cell phones, headsets, PDAs and cars. Table 2.1 summaries the main differences between these technologies [25].

2.4 Security Algorithms

The security represents one of the main priorities in this project, and it will be implemented at different levels. Starting from the database where all important information about the system and the patients will be stored. Moving to the wireless communication of the ZigBee network. Ending up with protecting the whole software with a log-in module that requires a password to make sure no one other doctors can use the map and other features of the software. So in this section we will discuss the main popular encryption algorithms that will be implemented in this project.

Table 2.1: Comparison between Wi-Fi, Bluetooth and ZigBee technologies [25].

| | Bluetooth | ZigBee | Wi-Fi |
|-------------------|------------------|-----------------------|--------------------|
| Frequency band | 2.4 GHz | 868/915 MHz, 2.4 GHz | 2.4 GHz, 5 GHz |
| Data rate | 1 Mbps | 250 Kpbs | 54 Mbps |
| Nominal range | 10 m | 10-100 m | 50-100 m |
| # RF channels | 79 | 16(for 2.4 GHz) | 14(for 2.4 GHz) |
| Channel bandwidth | 1 MHz | 2 MHz (for 2.4 GHz) | 22 MHz |
| Modulation type | GFSK | BPSK(+ASK), O-QPSK | QPSK, M-QAM |
| Spreading | FHSS | DSSS | DSSS, OFDM |
| Basic cell (BC) | Piconet | Star | BSS |
| Extension on BC | Scatternet | Cluster-tree, Mesh | ESS |
| Max # of nodes | 8 | > 65000 | 2007 |
| Encryption | E0 stream cipher | AES block cipher | WEP |
| Authentication | Shared secret | CBC-MAC (ext. of CCM) | WPA2 (for 802.11i) |
| Data protection | 16-bit CRC | 16-bit CRC | 32-bit CRC |
| Power Consumption | Medium | Low | High |

2.4.1 Advanced Encryption Standard

The Advanced Encryption Standard (AES) was issued as a federal information processing standard (FIPS 197). AES uses a block length of 128 bits and a key length that can be 128, 192, or 256 bits. In this section, we will focus on keys that have a length of 128 bits since they are the most commonly used. This the algorithm is implemented in the ZigBee standard.

Figure 2.11 shows the overall structure of AES. The input to the encryption and decryption algorithms is a single 128-bit block. In FIPS PUB 197, this block is depicted as a square matrix of bytes. This block is copied into the *state array*, which is modified at each stage of encryption or decryption. After the final stage, *state* is copied to an output matrix. Similarly, the 128-bit key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words; each word is four bytes and the total key schedule is 44 words for the 128-bit key. The ordering of bytes within a matrix is by column orders.

The *AES key expansion algorithm* takes a 4-word (16-byte) key as an input and produces a linear array of 44 words (156 bytes). The key is copied into the first four words of the expanded key. The remainder of the expanded key is filled in four words at a time. The key that is provided as input is expanded into an array of forty-four 32-bit words. Four distinct words (128 bits) serve as a round key for each round. Four different stages are used, one of permutation and three of substitution to scramble the bits and then mix in key information: the first stage is a *substitute bytes*, that uses a specific table called *S-box*, to perform byte substitutions. The second is *shift rows* where a simple permutation is performed row by row. Then *mix columns*

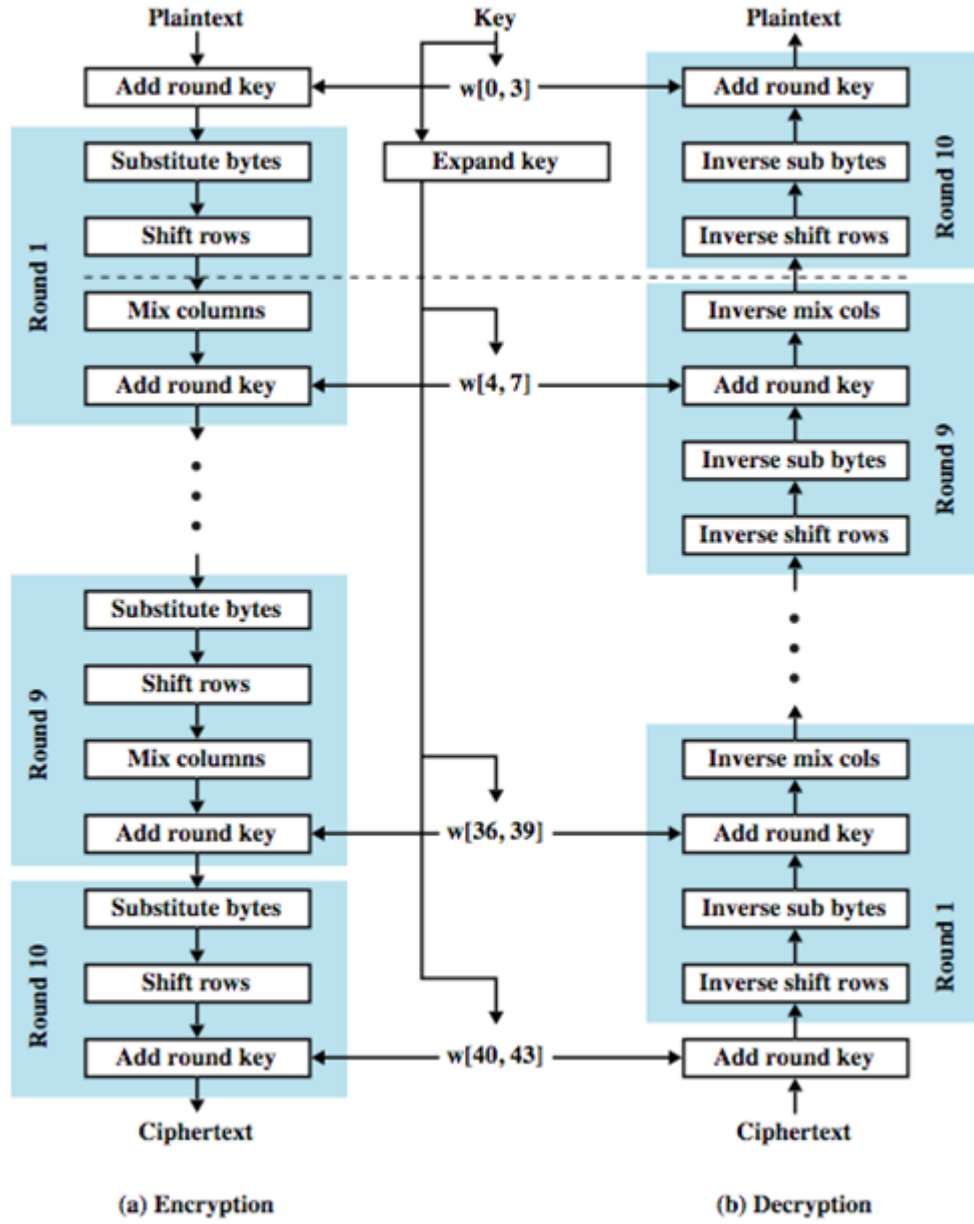


Figure 2.11: The overall structure of AES algorithm [26].

take place where a substitution rule alters each byte in a column as a function of all of the bytes in the column. Finally the *round key* is added where simple bitwise XOR of current block with expanded key.

The structure is quite simple. For both encryption and decryption, the cipher begins with an *Add Round Key* stage, followed by nine rounds each includes all four stages followed by a tenth round of three stages. Figure 2.12 depicts the structure of a full encryption round. Each stage is easily reversible, and the decryption algorithm uses the expanded key in reverse order with some modifications.

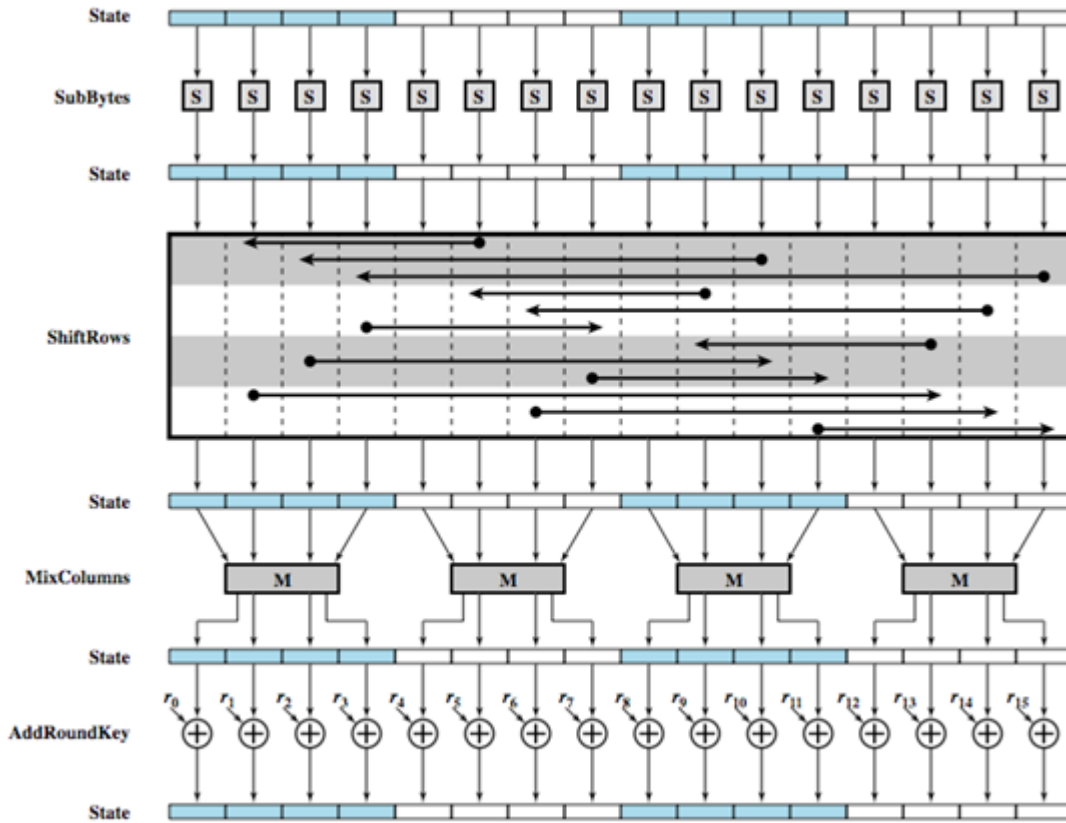


Figure 2.12: AES encryption round [26].

2.4.2 The Secure Hash Algorithm

The Secure Hash Algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993. The known versions of SHA are with hash value lengths of 256, 384, and 512 bits, known

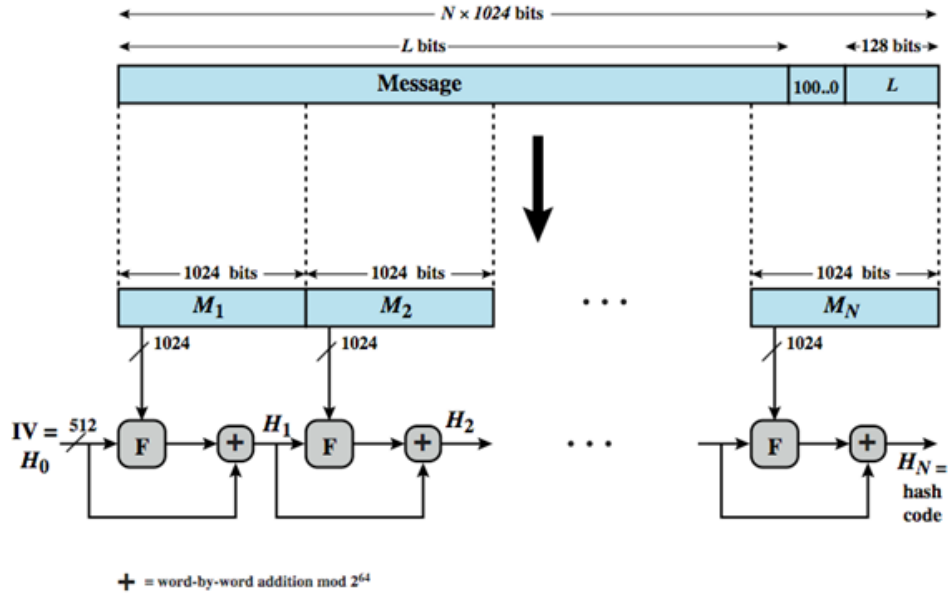


Figure 2.13: The secure hash algorithm [26].

as SHA-256, SHA-384, and SHA-512. These versions have the same underlying structure and use the same types of modular arithmetic and logical binary operations. Followed it is provide a description of SHA-512. The other versions quite similar, the algorithm takes as input a message with a maximum length of less than 2128 bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks. Figure 2.13 depicts the overall processing of a message to produce a digest.

The processing consists of the following steps: Step one appends padding bits so that message length is congruent to 896 modulo 1024 [length = 896 (mod 1024)]. The padding consists of a single 1-bit followed by the necessary number of 0-bits. Step two appends length as a block of 128 bits being an unsigned 128-bit integer length of the original message (before padding). In step three, the algorithm initializes hash buffer to the specified 64-bit integer values. While in step four it process the message in 1024-bit blocks, which forms the heart of the algorithm, being a module, labeled F in this figure that consists of 80 rounds. Finally, the Output of the final hash buffer value as the resulting hash. After all process the result length is 512 bit.

Hash algorithms, such as SHA, are one way functions. They turn any amount of data into a fixed-length "fingerprint" that cannot be reversed. They also have the property that if the input changes by even a tiny bit, the resulting hash is completely different. This is great for protecting passwords, because passwords needed to be stored in an encrypted form that is impossible to be decrypted. At the same time, passwords is also needed to be used to verify that a user's

password is correct without the need of decrypting it. The following example illustrate the powerfulness of such algorithms.

```
hash("hello") = 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
hash("hbllo") = 58756879c05c68dfac9866712fad6a93f8146f337a69afe7dd238f3364946366
hash("waltz") = c0e81794384491161f1777c232bc6bd9ec38f616560b120fda8e90f383853542
```

Figure 2.14: Illustration shows how powerful hashing algorithms are [27].

However, applying only normal hashing can be easily guessed using some advanced attacking techniques such as *dictionary attacks* and *brute-force attacks*. This is because passwords are always hashed in the same way such that if two users have the same password they will have the same hashed results. This problem can be solved by randomize each hash, so if the password hashed twice the results will not be the same. Randomize hashes can be done for the password before being hashed by preparing random string for it, called *salt*. Figure 2.15 example shows how this will make completely different hashes for the same password. In order to check the correctness of the password we need to keep its salt usually it is stored in the database along with the hash, or as a part of the hash string itself. There are many other common mistakes that weakening the system's security such as re-using a salt many times, or using short salts (since they are easy to be guessed).

```
hash("hello") = 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
hash("hello" + "QxLUF1bgIAdeQX") = 9e209040c863f84a31e719795b2577523954739fe5ed3b58a75cff2127075ed1
hash("hello" + "bv5PehSMfV11Cd") = d1d3ec2e6f20fd420d50e2642992841d8338a314b8ea157c9e18477aaef226ab
hash("hello" + "YYLmfY6IehjZMQ") = a49670c3c18b9e079b9cfaf51634f563dc8ae3070db2c4a8544305df1b60f007
```

Figure 2.15: Illustration shows randomized hashing algorithms [27].

2.5 Software Development Tools

2.5.1 C# Programming Language

C# (pronounced as 'C Sharp') is described as a simple, modern, object-oriented and type-safe programming language which derived from C, C++ and Java, to adapt the best features of each and adding new features of its own [28].

Developers discovered the need of Web-based applications that can be accessed and used via the internet, also the need for accessible software to anyone and available via almost any type of device. To achieve these needs, in 2000, Microsoft announced the C# programming language. C# developed at Microsoft by a team led by Andres Hejlsberg and Scott Wiltamuth, was designed specifically for the .NET platform as a language that would enable programmers to migrate easily to .NET [29].

C# and Java C# definition has been derived from C++ primarily with same strengths. Because of the industry rivalry between Microsoft and Sun, the impact perception that java hadnt on the development on C#. for anyone who know both languages, the similarities are evident. **The following list shows some features shared between C# and Java**, which are improvement on C++:

- Exceptions for error handling are used in both.
- Arrays are bound checked.
- Using inline code comments to produce API documentation.
- Packages/namespaces are used to avoid type collision.
- All classes are descend from Object and are allocated in the heap when created.
- There are no global functions or constants because everything must belong to a class.
- Multiple inheritances are abandoned although multiple interfaces implantations are allowed.
- Automatic garbage collection is supported by each runtime environment.
- Compilation of the source code to intermediate format that is run in managed environment.

Besides sharing number of features with Java, also most of C# keywords have its counterpart in Java, most part of these keywords are coming from C++ like *class*, *new*, *this*, *throw* and *break*. But it is interesting to note that Java keywords with no C++ equivalent have other names in C# for example keywords *super*, *import* and *final* in Java are called in C# *base*, *using* and *sealed*.

As said before there are a lot of similarities between the two languages, which brings a great advantage of ease the transition in learning the language. But C# also brings some significant changes and features that are not found in Java. **The following list will show in brief some of the main differences** and features between C# and Java:

- C# provides easier way for handling iterations by adding iteration statement *foreach*.

- C# has more concise syntax for encapsulating data using property which is essentially a pair of *get* and *set* accessory methods for accessing a field, that access actually implementing through a class method.
- C# provides a powerful alternative for constants which is enumerations with *enum* keyword; *enum* has distinct values with a set of named constants.
- C# allows operator overloading to redefine the semantics of operators for classes, which is not allowed in Java.
- C# provides XML style documentation for inline documentation which is more flexible than Javadocs. This makes the creation of online and print references documentation more simplified for applications.
- C# natively supports the Component Object Model (COM) and Windows-based APIs.
- C# allows the use of pointers for direct memory access, which provide programmers with external level of control that is not available in Java.
- C# provides a type-safe mechanism for implementing callback functions and events.

As a conclusion, C# brings the raw power of C++ and the simplicity of Java language.

C# in our project C# will be the programming language used in this project to deal with the optimized location data, display it on the map, deal with information stored database and making interface for user usage.

2.5.2 Database

A database is a structured collection of data stored for the use by one or more applications. In addition to data, database contains the relationships between data items and groups of data items. Database is managed by a *database management system* (DBMS), which is suite of programs for constructing and maintaining the database and for offering query facilities to multiple users and applications. A query language provides a uniform interface to the database for users and applications [26].

Figure 2.16 provides a simplified block diagram of DBMS architecture. Developers make use of *data definitions language* (DDL) to define the database logical structure and procedural properties, which are represented by set of database description tables. A *data manipulation language* (DML) provides a powerful set of tools for application developers. *Query language* is a declarative language designed to support the users. The database management makes use of the database description tables to manage the physical database.

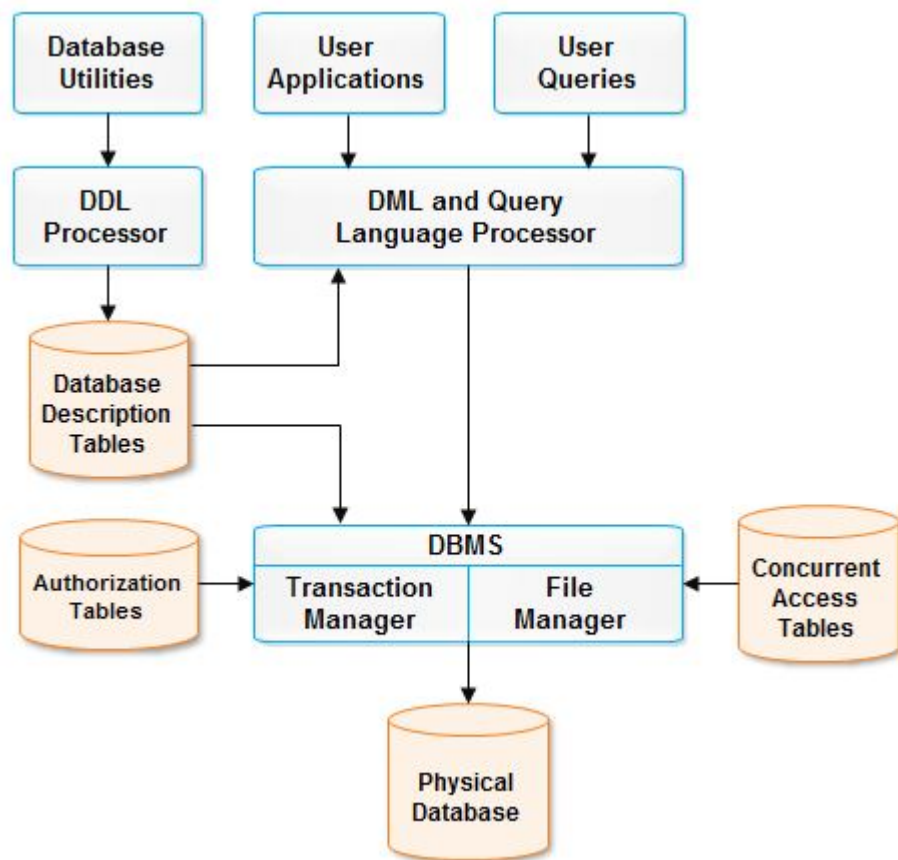


Figure 2.16: Simplified block diagram of DBMS architecture.

Chapter 3

System Design

It is desired to build a monitoring system for treatment centers, that are dedicated for people with intellectual disabilities, to take care of those individuals while they are living inside. The best thing a monitoring system can do is to support care-providers with "some information" about their patients from which they can provide each patient with the needed support. The first step in designing an effective monitoring system is to determine what kind of information the system should provide, how these information will be used, and how meaningful is it. On the other hand, these information should easily be managed and stored, especially when we talk about long-term monitoring. In addition, it should be clearly displayed when large number of patients are needed to be monitored over wide area.

One direct solution will be deploying a grid of cameras to provide the system administrator with real time videos about what do patients do around the center. These cameras will save a lot of effort and will reduce the number of staff members needed to constantly monitor the patients during their daily life. A patient can spend his time in the way he wants without being under direct supervision, this will give him a certain level of self-confidence, independence and privacy. In the same time, the administrator will sit on his computer and keep switching between different cameras to make sure that everything in the center is in going well. If a patient needs help, the administrator will call a staff member to go and provide that patient with the needed support.

In fact, there are many problems with this type of monitoring systems. We will try to analyze them to move to our model. Firstly, it is not easy to cover all treatment center with cameras. Cameras are limited in their coverage and can't see hidden corners, can't see in dark, and even can't be placed in private rooms. Secondly, the admin is not able to monitor all patients at the same time. He is limited by the computer screen dimensions so that he can view only few cameras simultaneously. He have to periodically switch between different cameras to verify that everything is ok, which is not an easy task when there is a large number of cameras. In this case, a patient in need will have to wait for a long time until he gets helped. The third problem is that storing video records is not an easy task due to their large size, especially for long-term monitoring. Moreover, it is not possible for the admin to focus on a certain patient while he is

moving around. The admin has to remember the name and the location of all cameras, which cover the path a patient is travelling along, and keep switching between them in order to track that patient.

In our system we use different approach to overcome the aforementioned problems. The first step is to use a dedicated monitoring device for each patient instead of dedicated monitoring device for each specific location in the center. Here, we replace the cameras with a wireless sensor network consists of small wireless sensors, each sensor will be attached to an individual patient. The admin will not be able to *see* the patients anymore, but he can use the wireless sensor attached to each patient to *locate* and *track* him wherever he goes. In doing so, the monitoring problem becomes localization problem, and this is the innovation in the project.

In section 3.1, we will continue discussing the the project design and its calculations. While section 3.2 is dedicated for the Software part of the system which includes the security and the database.

3.1 Problem Formulation and Modeling

The system can be divided into three parts: The first is the *mobile node* which is carried by a patient for the monitoring purposes. Then the *anchors network* which is a wireless infrastructure consists of several anchor nodes connected together, and used as reference points to estimate the patient location with respect to them. They are also responsible for delivering the data from the mobile nodes to the server. Finally, the *server* is the system core. It is composed of four functional blocks each perform different tasks: it controls and manages the wireless network through the *network coordinator*. It stores different types of data about each individual patient in a *database*, the *location and tracking engine* (L&TE) use the received signal strength data collected by different anchors to predict the location of all patients within the treatment center. Then it will deliver the final result to the main *software* to be displayed. The software is the user's interface which allow him to interact with whole system. A general block diagram is shown in figure 3.1

3.1.1 Mobile node

A *mobile node* is simply a ZigBee transceiver powered by a portable power source (i.e. battery). Each patient have to have one in order to be connected to the monitoring system. Once the connection has established, the transceiver will become under the coordinator control. When the admin initiates a command to locate a patient or track him, mobile nodes start broadcasting a message with specific format (beacon). This message will be received by the neighboring anchors and the received signal strength will be recorded and delivered to the coordinator for further processing. An optional sensor may be included to extend the mobile node capability, such sensors can be used to monitor the patient's health status. In this case, a

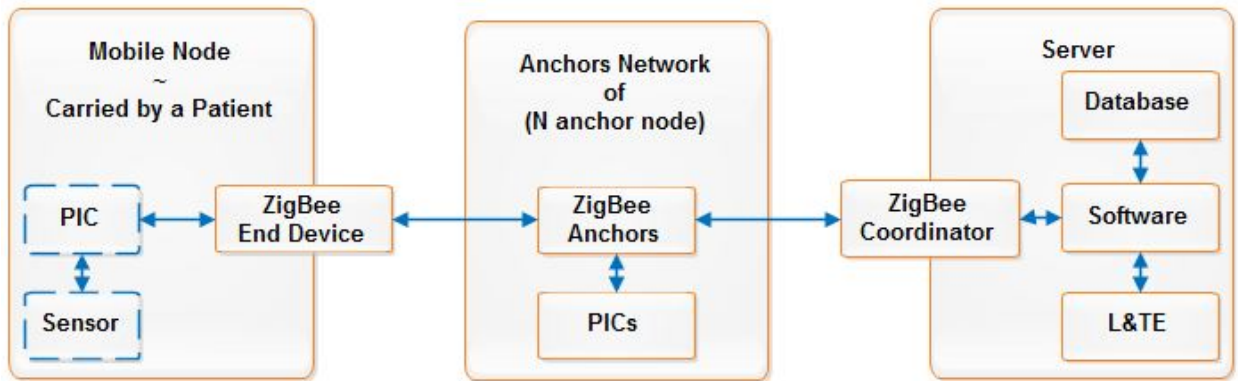


Figure 3.1: General block diagram of the system.

PIC microcontroller will be required. Further discussion on this point will be explained later in the project.

There are wide range of wireless transceivers available in the market with different specifications. Wi-Fi, Bluetooth, and ZigBee are mostly used for localization. As was mentioned in section 2.3, Bluetooth transceivers suffer from the limited range which is typically less than 10 m, Wi-Fi transceivers achieve higher communication range but consume a lot of power. However, ZigBee transceivers are known to have ultra low power consumption and provide long communication range. These advantages in conjunction with their low cost and small size make them perfect solution for this project.

We will use XBee-PRO DigiMesh 2.4 transceiver for both mobile and anchor nodes (as seen figure 3.2).



Figure 3.2: XBee-PRO DigiMesh 2.4 module.

Table 3.1: Main specifications of XBee-PRO DigiMesh 2.4 transceivers

| Performance | Typical Value |
|------------------------------|---------------------|
| RF Data Rate | 250 Kbps |
| Indoor/Urban Range | 300 ft (90 m) |
| Transmit Power | 63 mW (+18 dBm) |
| Receiver Sensitivity (1%BER) | -100 dBm |
| Transmit Current | 250 mA |
| Receive Current | 55 mA |
| Sleep Current | 50 μ A |
| Physical Dimensions | 24.38 mm x 27.61 mm |

This module provides wireless connectivity using a globally deployable 2.4 GHz transceiver using DigiMesh networking protocol. This innovative, peer-to-peer network offers users added network stability through self-healing, dense network operation and support for sleeping routers to extend the operational life of the mobile node battery [30]. Three types of nodes are defined in the ZigBee protocol: *coordinator*, *router* and *end device*. However, DigiMesh implementation define a peer-to-peer network and support a single type of nodes which is the *router*. Table 3.1 summarizes the main specifications of the XBee-PRO DigiMesh 2.4 transceiver.

Since the mobile nodes are energy-constrained, energy efficiency is an important requirement in our system. Various mechanisms have been proposed in the literature to minimize the energy consumption of the nodes. From table 3.1 we can observe that the energy consumption of a radio transceiver in the receive or transmit modes is much larger than the energy consumption of the sleep mode. Thus, significant energy saving can be accomplished if the radio transceivers stay in sleep mode as long as possible. In fact, ZigBee standard does not allow routers to sleep, because they must always be available for routing. Only end devices can sleep. Messages destined for a sleep-enabled End Device are buffered by its parent for collection by the End Device once it is awake.

However, DigiMesh supports *sleeping routers* feature with different modes. These sleep modes are enabled with the **SM command** and characterized as either *asynchronous* or *synchronous*. Asynchronous sleep modes can be used to control the sleep state on a module by module basis. However, we will use synchronous sleep mode which makes it possible for all mobile nodes in the network to synchronize their sleep and wake times and form a *cyclic sleeping network*. Nodes that are involved in this process can be configured in one of two modes: *synchronous sleep support mode* (with **SM** = 7) and *synchronous cyclic sleep mode* (with **SM** = 8). A node with sleep support mode will synchronize itself with a sleeping network but will *not* itself sleep in order to allow new nodes to join the sleeping network. In our system, *anchor nodes* will be our sleep supporters while *mobile nodes* will sleep to save their power.

The sleep and wake times for the entire network can be controlled by the coordinator depends on the desired function (i.e. location or tracking the patient, or just regular update). The coordinator will broadcast a special RF packet called *synchronization message* (SyncMsg) the beginning of each wake period. This message contains synchronization information for the current cycle. All cyclic sleep nodes receiving a SyncMsg will remain awake for the wake time and then sleep for the sleep period specified. Generally, sleep and wake times are specified by the **SP** and **ST** commands respectively. A newly-powered unsynchronized sleeping node (and even those have lost the synchronization) will *poll* for a synchronized message and then sleep for the period specified by **SP**, repeating this cycle until it becomes synchronized by receiving a SyncMsg. Once it becomes synchronized, its sleep and wake times can be queried with the **OS** and **OW** commands respectively.

In order to guarantee that a destination radio will be awake when a transmission is sent, sleep guard times (SGT) are allocated at the beginning and end of the wake time. The size of the SGT varies based on the sleep and wake times selected and the number of cycles that have elapsed since the last SyncMsg was received. The delivery of the SyncMsg to the whole network is guaranteed using negative acknowledgement mechanism which is used for any broadcast transmission. If the sleep coordinator does not hear a re-broadcast of the SyncMsg by one of its immediate neighbors (sleep supporter), then the message is assumed to be lost and the coordinator will re-send the message once again. In fact, the sleeping network is robust enough that an individual node can go several cycles without receiving a SyncMsg. As a node misses SyncMsgs, the sleep guard time increases in duration and the time available for transmitting messages in the wake time is reduced to maintain synchronization accuracy. By default, a module will also reduce its active sleep time progressively as SyncMsgs are missed. Figure 3.3 summarizes the behavior of the sleeping network.

The following table summarizes some important AT Commands that will be used to configure Mobile nodes.

3.1.2 Anchor Network

The *anchor network* is the hardware part of the L&TE. It consists of a number of fixed nodes (anchors) distributed in the treatment center to cover the whole area. These nodes are anchors in the sense they are in fixed and known locations, and used to send periodic messages (beacons) for different purposes. Anchors network is responsible for three main functions: collecting the RSSI data from mobile nodes to enable the location and tracking process (L&TP), packets routing through the network and synchronizing mobile nodes cyclic sleeping.

An *anchor node* consists of XBee-PRO DigiMesh 2.4 transceiver controlled by a PIC micro-controller. Unlike mobile nodes, anchor nodes are not constrained by the power consumption

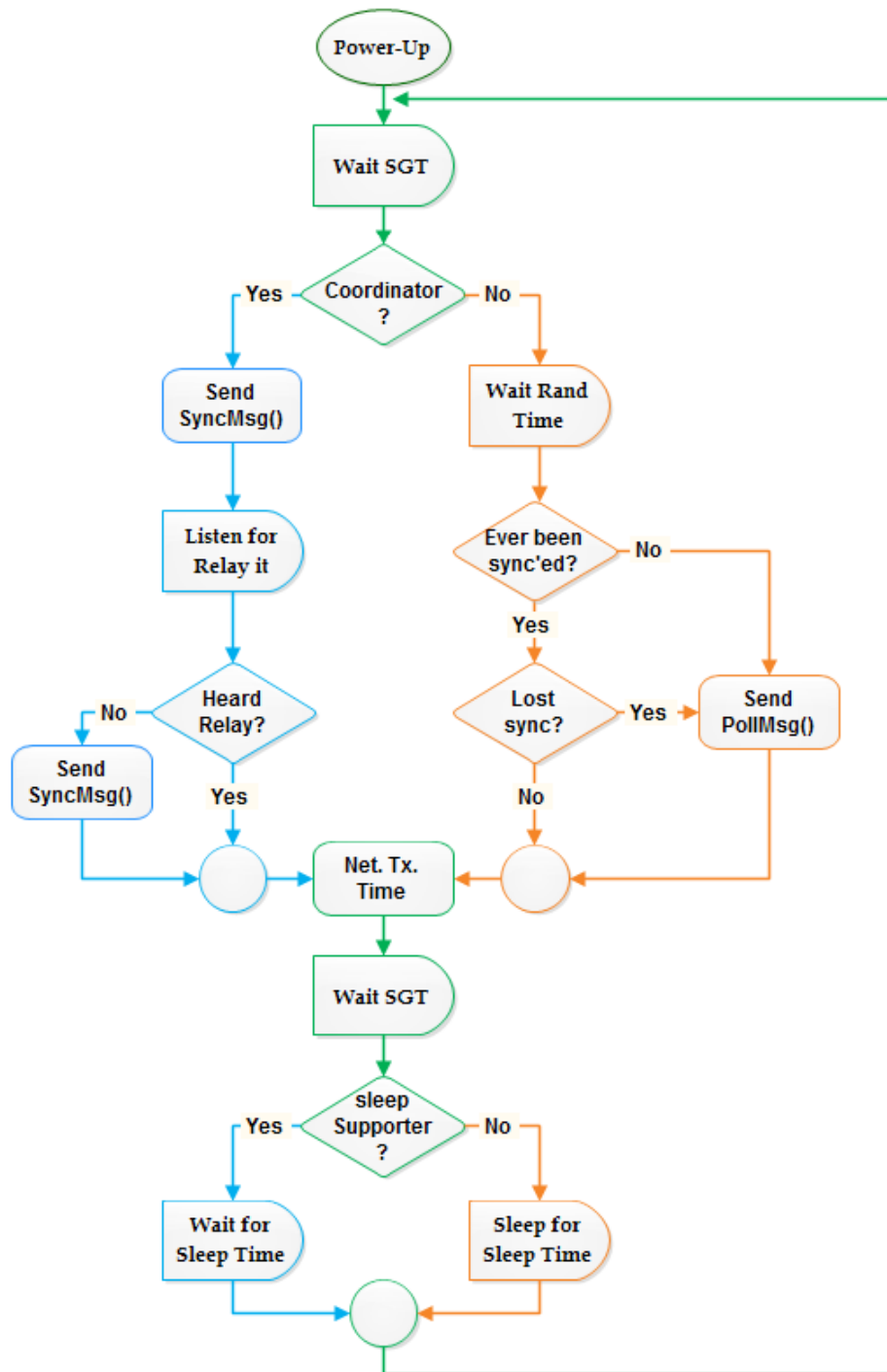


Figure 3.3: The synchronization behavior of sleep compatible nodes.

Table 3.2: Summary of AT Commands for mobile nodes.

| AT Command | Function Description | Suggested Parameter(s) |
|------------|------------------------------|------------------------------------|
| CE | Define node type | 0 = Router |
| SM | Define sleep mode | 8 = Synchronous cyclic sleep mode. |
| SO | Mobile Node sleeping options | 0x02 = Non-sleep coordinator |
| ST | Wake Time per cycle | 0x1388 = 5 s |
| SP | Sleep Period per cycle | 0x61A8 = 25 s |
| WH | Wake Early to be prepared | 0x01 = 1 ms |
| PL | Transmit power level | 0 = 10 dBm |

since they are powered by a constant electric source. They are configured as routers and serve under the control of the network's coordinator associated with the server. When the admin initiates a command to locate a patient or track him, the system will pass through the L&TP which is mainly three steps: firstly, mobile nodes start broadcasting messages called *update message* (UpdateMsg) with specific format, and the received signal strength will be recorded by the anchor nodes. Then the measured RSSI data will be arranged in a packet called *data message* (DataMsg) and delivered to the coordinator. Finally the output result (patient location) will be calculated and displayed on the map.

Both localization and tracking follows the same procedure with two minor differences: the patient's location update rate is higher in the case of tracking in order to have smooth transition when displaying patient's movement on the map. Tracking is activated only upon the admin request for specific patients for specific time period. However, localization function is performed periodically every 30 sec to update the location information of the whole mobile nodes. The maximum number of simultaneous patients to be tracked is determined by the maximum time delay can be allowed between two updates (time between two successive UpdateMsg), so assume that patients move in an average speed of 1 m/sec then the update rate should be at most every 2 sec. In other words, tracking is nothing more than multiple location updates performed at higher update rate for specific patients. Other than that, both commands use the same packets' format and the same procedure shown in figure 3.4.

As we see in figure 3.4, L&TP is three stages procedure each needs different time period to complete depends on the amount of data will be exchanged through the network. In the first stage, mobile nodes will compete on the channel to send their UpdateMsg while anchor nodes will read and store the power level of these message. This stage is the heaviest in terms of the amount of traffic being generated, so efficient multiple access algorithm is needed and reliable routing protocol should be used. DigiMesh uses the AODV routing algorithm (which is described in section 2.3). In order to accomplish reliable delivery for these broadcast messages,

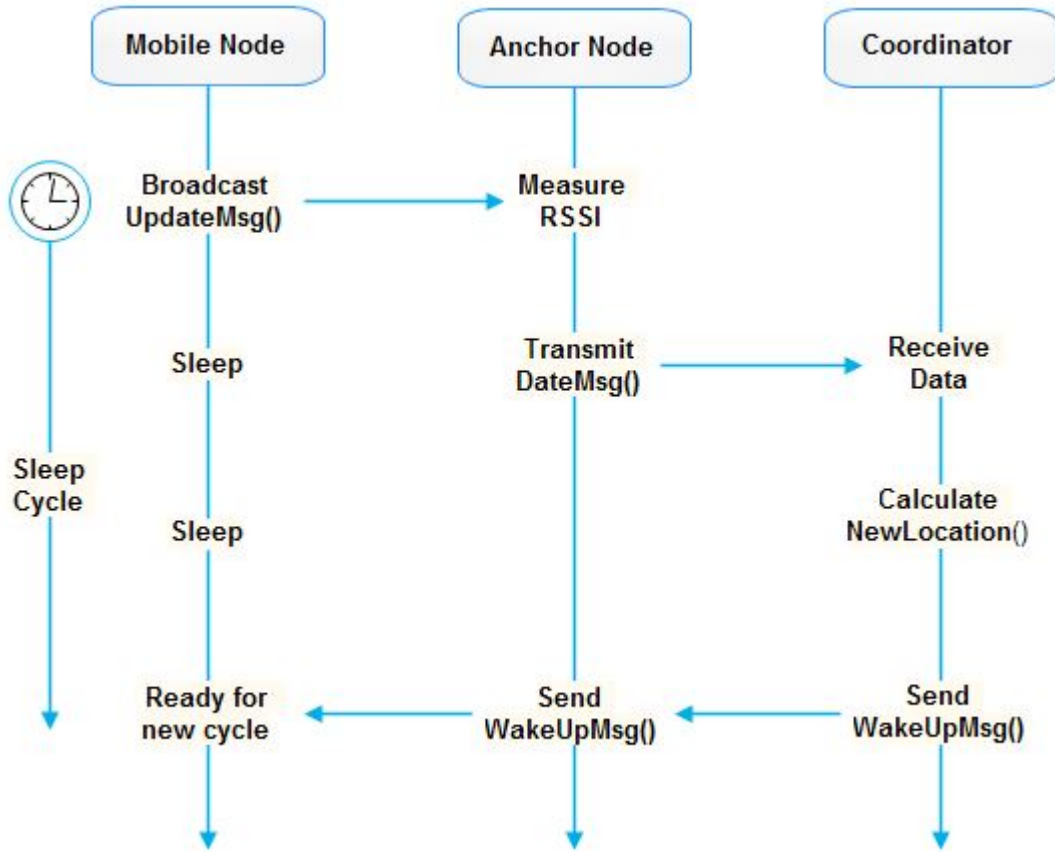


Figure 3.4: Localization and tracking procedure.

ZigBee protocol use to transmit multiple copies of the same packet to assure its arrival to the desired destination instead using the acknowledgement mechanism. By default a broadcast transmission is sent 4 times, and the broadcast radius is set to be maximum. To reduce overhead in the network, only 2 copies of the UpdateMsg will be sent. This will also reduce the power consumption on the mobile node.

We use broadcast message type for the UpdateMsg because its destination address is not known to the sender. The mobile node does not know which anchor node will be able to hear the UpdateMsg and which will not. For that we use the broadcast address (0x000000000000FFFF) as the destination address. Broadcast messages, by their nature, propagate through the network to reach all nodes which is undesired behavior in our case. In principle, we are interested in a subset of anchor nodes (AN_1 in figure 3.5) which are within the "one hop" range. The RSSI values measured at those anchors are meaningful because they reflect the real power-distance relationship and will be used to calculate patient's location. RSSI values can be read on pin 6 using DB Command. These values only indicates the received signal strength of the last hop. If a

transmission spans multiple hops, the measured RSS value provides no indication of the overall transmission path. For that, we set the broadcast radius to be 1 hop (using **BH Command**); to make sure that an anchor node will not retransmit the message after recording its RSSI. When a mobile node completes its transmission, it went in the sleep mode to save its power.

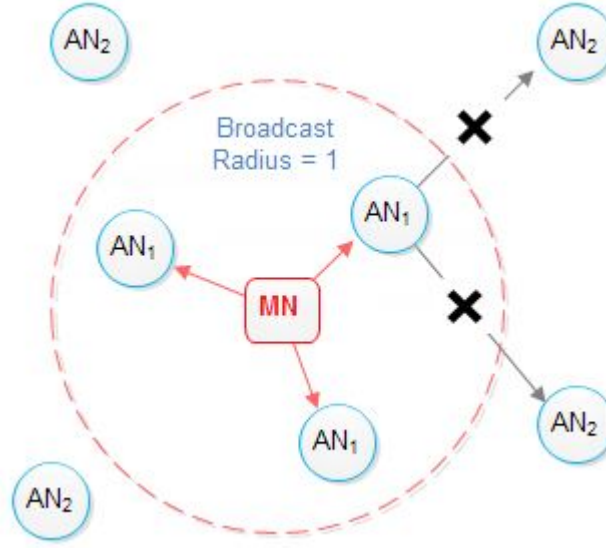


Figure 3.5: Only anchor nodes within one-hop range will receive UpdateMsg sent by the mobile node.

On the anchor node side, multiple UpdateMsg will be received from different mobile nodes. The address of each mobile node will be stored together with the power level of its UpdateMsg. When the first stage is completed, mobile nodes will sleep while each anchor node will generate a packet called DataMsg that contains all stored data in order to deliver it to the coordinator for further processing. Reliable delivery of data is accomplished using retries and acknowledgements (simple unicast transmission). *DataMsgAck* is not cumulative and is sent as soon as the DataMsg is received at the coordinator side. The packet format of both UpdateMsg and DataMsg is shown in figure 3.6.

MobID is a string identifier assigned to each mobile node once it joins the network through the **NI Command**. The string accepts only printable ASCII data and can't start with a space. This ID is not part of the network addressing and do not used for routing. The idea behind this ID is to replace the 64-bit MAC address used in the DataMsg packet in order to significantly reduce the packet size. Note that at most two UpdateMsg will be received from each mobile node if no packet loss occurs. This duplication will be resolved at the server by averaging those RSSI values. The last step of L&TP will be done on the server through the L&TE which is covered in section 3.1.3.

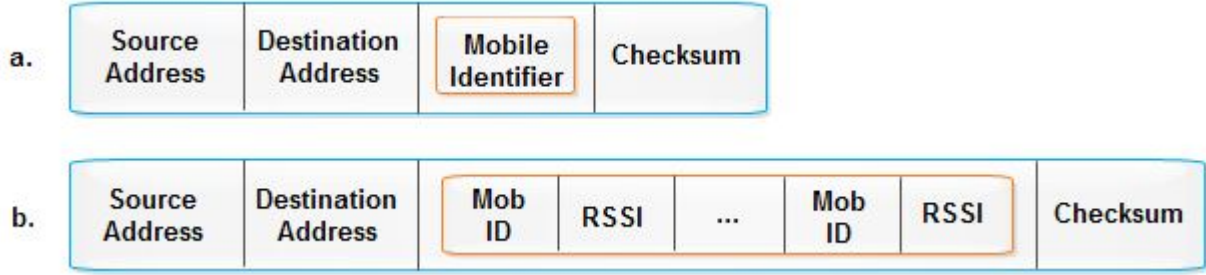


Figure 3.6: Messages exchanged in the L&TP: (a) UpdateMsg contains mobile node ID, (b) DataMsg used to deliver the information on the RSSI samples collected at the anchor nodes out of UpdateMsg.

In DigiMesh, the AODV is used with some modifications. By sending a message to the next hop address, either the message will reach its destination or be forwarded to an intermediate router which will route the message on to its destination. A message with a broadcast address is broadcast to all neighbors. All routers receiving the message will rebroadcast the message $MT+1$ times (MT is Broadcast Multi-Transmit). Packet tracking prevents a node from resending a broadcast message more than $MT+1$ times.

3.1.3 The Server

The *server* is the heart of the system where the data storage, processing and presentation will take place. From the software main window, the admin can initiate commands to track some patients or to monitor the others. He can add new patients to the system and/or edit the available information about existing patients. In this subsection, we will discuss the role of the server in the L&TP. The Software and the Database will be described in details in section 3.2.

The L&TP starts from the server, takes place during the available transmission time when the network is awake and returns to be completed at the server. The RSSI values are collected and sent to the server in the first and second stages of the L&TP as seen in figure 3.4. Further processing will be done in the L&TE, where the first step here is to extract and separate the RSSI measurements of each mobile node from the received DataMsgs. This is important because the location of each mobile node will be calculated from its RSSI values, which are not contained in one DataMsg but distributed on different messages. This depends on which anchor node was able to hear the UpdateMsg sent by that specific mobile node and which was not. The output of this stage will be an *RSSI matrix* with number of rows equal to the number of mobile nodes needed to be located or tracked (say m), and the number of columns are equal to the number of available RSSI samples for each mobile node (say k). The RSSI matrix will be applied to the neural network to calculate the associated location. The location will be defined in terms of four coordinates: x , y , z (floor number), θ (orientation: North, East, South and West).

However, before applying the RSSI data of each mobile node to the neural network, we need to make sure that the number of RSSI samples available for the mobile nodes are fixed and equal to the number of neural network's inputs. In general, this will not be the case since the number of available RSSI samples will differ from one mobile node to another depends on the location of each mobile node with respect to the surrounding anchor nodes. Figure 3.7 shows a typical situation where four anchor nodes (namely AN4, AN5, AN6 and AN7) can hear the UpdateMsg from the second mobile node (MN2), while only three anchor nodes (AN2, AN3 and AN4) are able to hear the UpdateMsg sent by the first mobile node (MN1). In order to solve this problem, we have two choices: either we simply assume that the number of inputs will be fed to the neural network is equal to the number of fixed node (i.e. $k = N$), and consider the RSSI values from the far fixed nodes to be zeros. Or we restrict ourselves with only the first k -dominant RSSI samples, where k will be determined depends on the average number of available RSSI samples on each mobile node. This will reduce the number of zeros in the RSSI measurements that will be applied to the neural network. For this semester, our calculations will be based on the former choice.

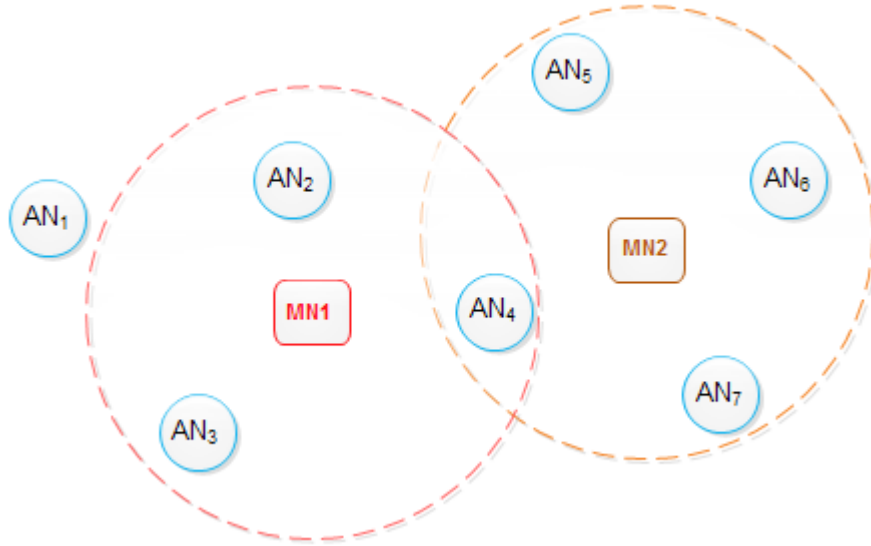


Figure 3.7: Illustration shows how k may change from one mobile node to another depends on the mobile node position with respect to the anchors network: $k(MN1) = 3$, $k(MN2) = 4$.

The number of available RSSI samples from each mobile node (i.e. k) is affected by the criteria that is used to distribute the anchor nodes in the area. One of the factors that will be taken into account while distributing the anchors in the area, is to make sure that every point in the treatment center will be covered by at least three anchor nodes. Since there is no available software that can do this job for us, we are required to do that manually. The location of anchor nodes is also constrained by the 'RSSI fingerprint map' they generate. An RSSI map

will be generated (in an offline phase) by dividing the treatment center into a grid of equally spaced points (with 90 cm separation), and use the anchor nodes to measure a vector of RSSI values associated with each one. The map should provide a unique relationship between the RSSI vector of one location and the coordinates of that location, so that later (in the online phase) the L&TE can use the RSSI data of a mobile node with unknown location to find its coordinates. In order to achieve this uniqueness, we need to have a rich multipath environment by avoiding any direct LOS connection between anchor nodes.

After preparing RSSI matrix, every row in the matrix will be applied to the neural network to find the location (x, y, z, θ) of the associate mobile node. Since the location is a function of the measured power, we will exploit the neural network's fitting capability to approximate that function. We will use MLP network, which is a feed-forward network, that utilizes back-propagation algorithm to execute the training across the multiple layers. The gradient learning rule will be employed to minimize the error function, which is defined as the Euclidean distance between the correct location and the network output.

The number of layers and the number of neurons in each layer are important parameters that affect the network performance. Unlike the hidden layers, input layer and output layer are transparent (with unity activation function) and have known number of neurons. The number of input neurons is equal to the number of anchor nodes (i.e. N), while the number of output neurons will be four (x, y, z and θ). However, there is no general rule to determine the number of hidden layers and hidden neurons but to try different values and chose the one which provides us with the least error. In general, it is proven that a 3-layer MLP network represents an universal function approximate (Theorem of Cybenko [31]), so that a single hidden layer maybe good enough. On the other hand, the more the hidden neuron we have, the more the powerful the network is and the better the approximation will be. This is in general true if the number of training data is much larger (i.e. 10 times) than the total number of neurons in the network. Otherwise, the lack of enough training samples will lead to a poor results due to the over-fitting problem described in section 2.2.3.

Figure 3.8 illustrates how the L&TE works. The network's sleep cycle period is determined according to the required update rate of patients' locations. The update rate depends on the localization/ tracking command (L&TC) initiated by the admin. When the DataMsgs arrive, RSSI matrix will be extracted and applied to the neural networks. The hidden layers are responsible for the location calculation, and the result will be displayed on the map.

Note that even the relationship between the neural network and the human knowledge of the system may not be comprehensible or transparent, the hidden layers of the network are capable of producing a model with high accuracy. The lack of transparency makes it difficult task to discover the error source if the network produces unexpected results. This is the resone why artificial neural network models are often referred to as *black box* models. However, to partially overcome this problem, we will divide our 4-outputs neural network into 3-neural networks each

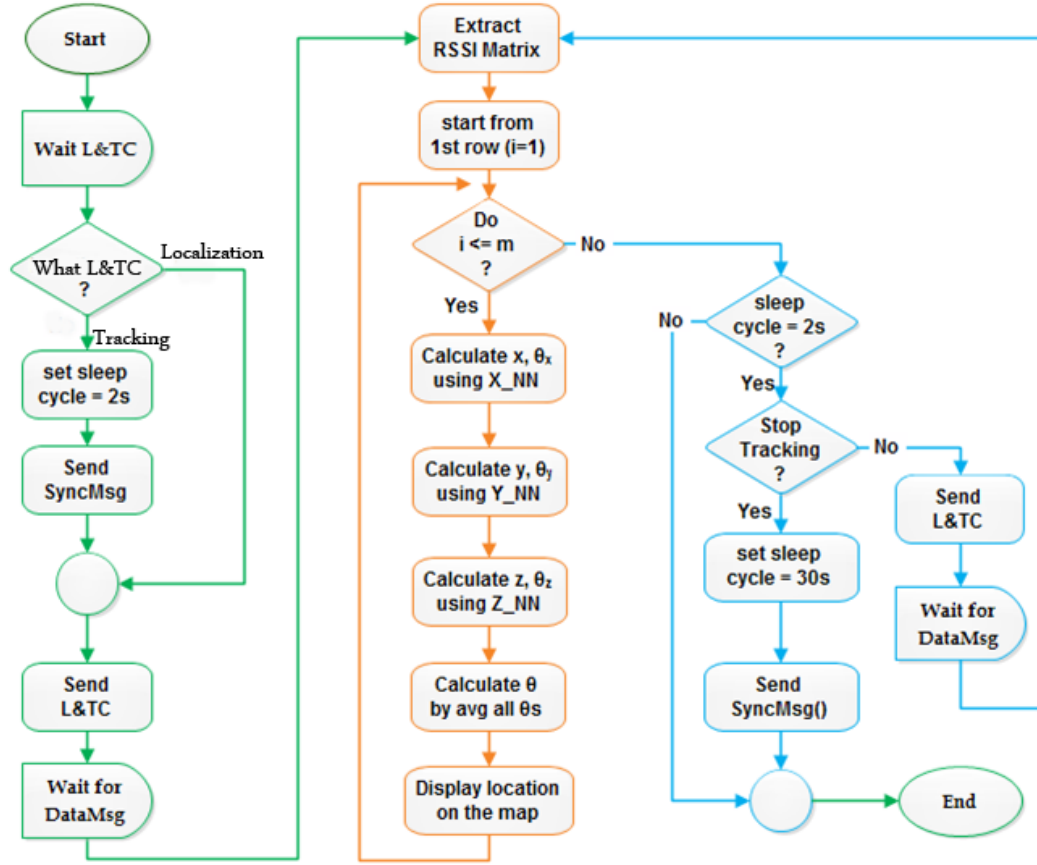


Figure 3.8: The Localization and tracking algorithm.

with two outputs(one coordinate and expected θ). Since we do not expect the neural network to be able to predict θ accurately, we find θ three times (one with x , and one with y , and one with z) then we average them all to get the final prediction of the patient direction of movement.

All neural networks will be trained independently using the same training set collected in the offline phase. Doing so, we will end up with 3 different neural networks each one describes one coordinate's component: X_NN is the one used for determining x , Y_NN is used for y , and Z_NN is used for z . This separation may help us to find a physical meaning for each parameter in the resultant equations to explain the output results we will have.

3.2 Software

To make the system available to interact with users, it needed to present it in understandable way. Figure 3.9 shows how the main monitoring window will look like. The main parts are the *map*, the *patients' list* and the *monitoring options*. The map will show the patients, care-

providers and fixed nodes locations each in different color. The idea behind determining the location of care-providers is to help the admin choosing the nearest member to a patient that needs help. The map supports *zooming* feature which allows the admin to focus on a specific region of the treatment center. The patients' list is connected to the database where patients' information are stored. The admin can select one or more patients at a time to view their profile, edit their data, or even start tracking them.

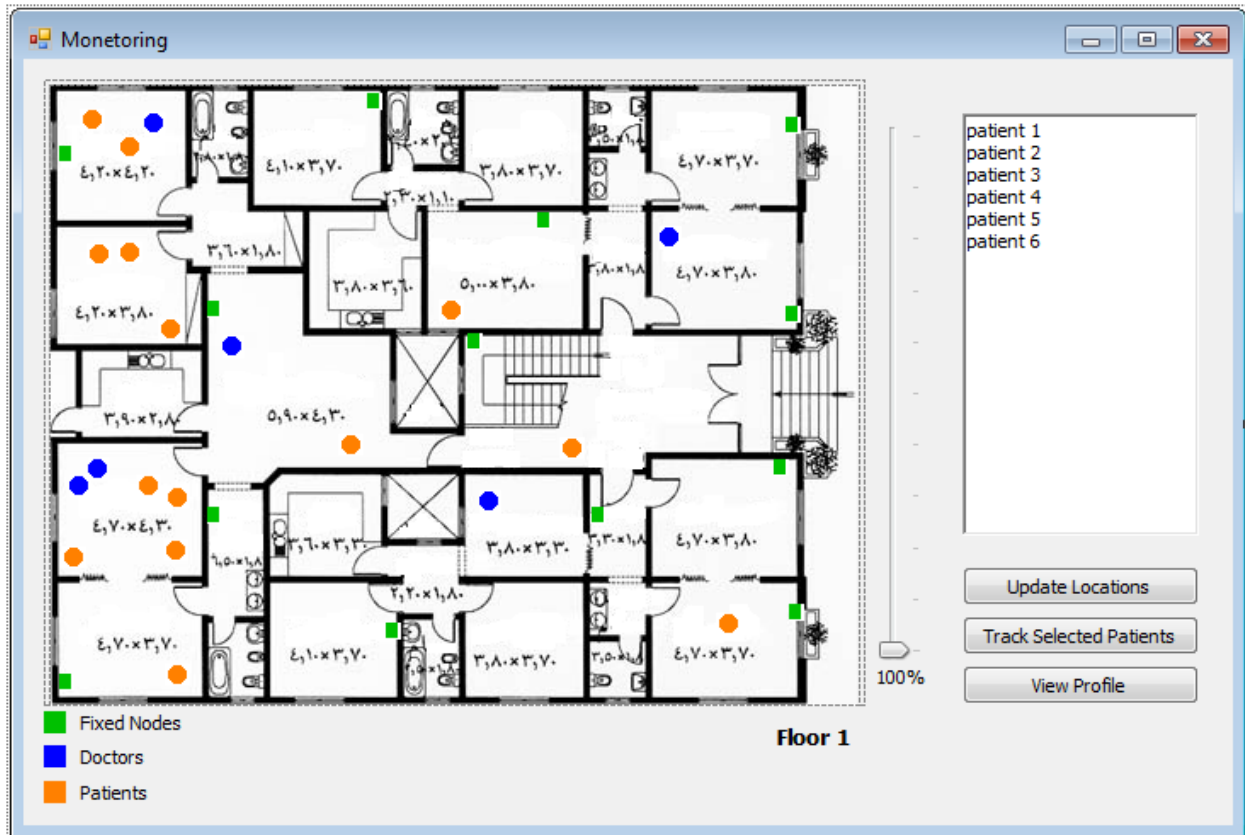


Figure 3.9: The graphical user interface of the software.

3.2.1 System users

The system has three users categories:

1. Patient: patients are the main category of which the system is made for. The database contains patients data in order to monitor their status and position. However, patients does not have any access to the system, their role is to hold the mobile node only.
2. Doctor/Nurse: those are care-providers. They can monitor patients locations, add new patients and add notes to patients profile in order to keep up with patients' status.

3. System admin: maybe the maintenance employee in the hospital, or the technical support employee. This user have full access to the system. He can configure the anchor nodes when they need to be replaced or maintained. Also this user can access the main log file for the system usage.

Before accessing the software, a username and password are required, after entering a correct username and password the system can be accessed. After accessing the system, it will wait to get an order from the admin to do one or more of its functions. The admin can deal with fixed nodes, view logger file and add new doctor. Figure 3.10 explains this process.

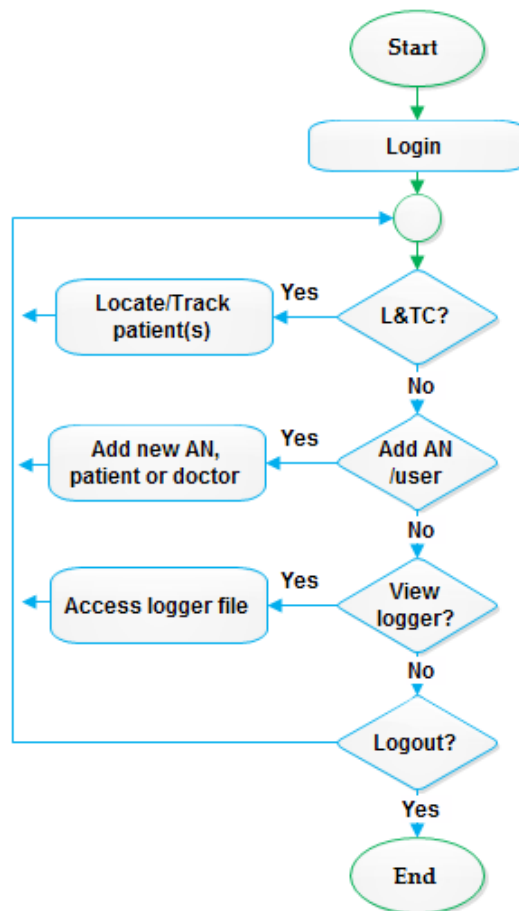


Figure 3.10: Software - admin interaction flowchart.

Before accessing the software, a username and password are required, after entering a correct username and password the system can be accessed. After accessing the system, it will wait to get an order from the doctor to do one or more of its functions. The doctor can request to

update patients locations, track one or more patients, add new patient, or view patient profile and logout the system. It is important to know that automating updating for location happened every 30 sec. Figure 3.11 explains this process.

3.2.2 System security

Its known how security become of most interest to all in recent years. It gives more reliability while using the system. Also, it is important to keep secret information away of unauthorized individuals. This monitoring system application will be opened on one computer and no information will be sent to other computers in the center. The security will be implemented at different levels:

Username/Password Before the system being accessed, it will ask the user to enter his username and password. The passwords will be encrypted before being saved in the database according to nominated encryption algorithm. The login process is shown in figure 3.12, where the user enters his username (`_name`) and password (`_pwd`) to login the system. The username is checked at first; if it does not exist, an error message will appear. If it exists, then the corresponding saved password (`pwd`) will be retrieved from the database. The retrieved password (`pwd`) is a *hashed* password that contains the unique user *salt*. To make sure that the password retrieved correctly its length will be checked, then we take the user *salt* from the password and add it to the entered password (`_pwd`) and send it to the *hashing function*. Finally the entered hashed password will be compared with the retrieved one (`pwd`). If they are equal then the user is authorized and the system can be accessed. Otherwise an error message will be appeared.

In order to provide high level of security, *salt* should be generated with high level of randomness in order to be unpredictable. The *salt* needs to be unique per-user per-password. Every time the users create new account or change password, the password should be hashed using a new random salt. It is preferred for the salt to be as long as the hash function output. We will implement a three step procedure to safely store system passwords and validate them. In order to store a password a random salt will be generated, then the password will be appended to the generated salt. The result is then hashed using a standard hashing function. Finally both, the salt and the hash, will be stored in the corresponding user record. The validation procedure was described before.

Mobile nodes block list Nodes from unknown source will be blocked and prevented from joining the network. A mobile node may send a request to join the network when a new patient is added to the system (i.e. new mobile node), or when an existing mobile node needs to rejoin the network after replacing its battery. The second case represent a good chance to unauthorized individuals to try jonahing the network since large number of requests are sent at this stage. The DigiMesh protocol does not provide a mechanism to control this activity. So in figure 3.13 we present an algorithm to check the source of the received *JoinRequest*.

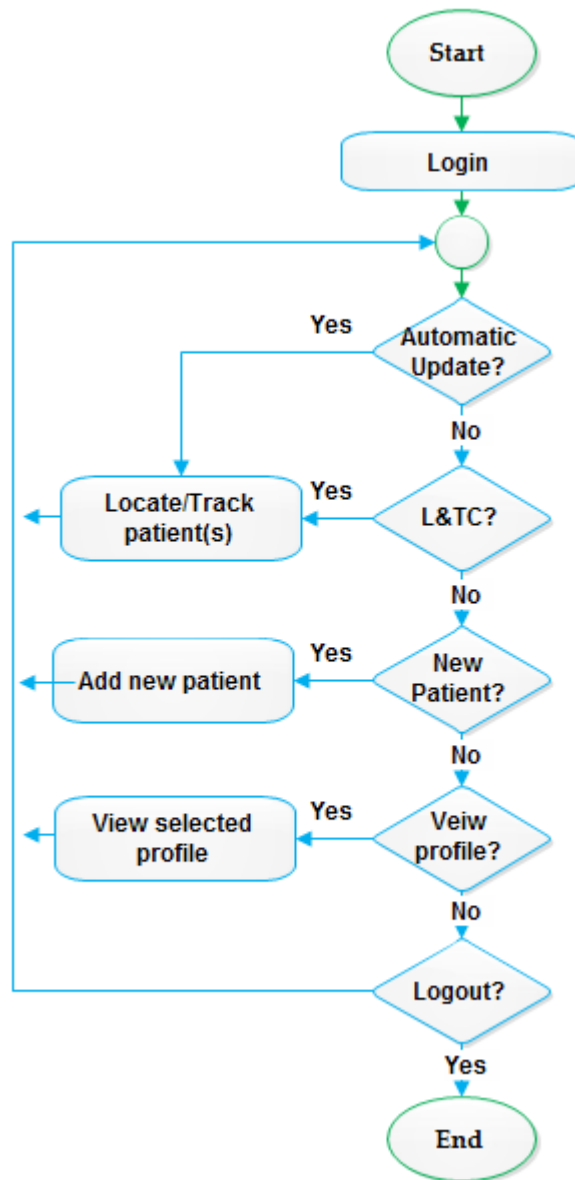


Figure 3.11: Software - doctor interaction flowchart.

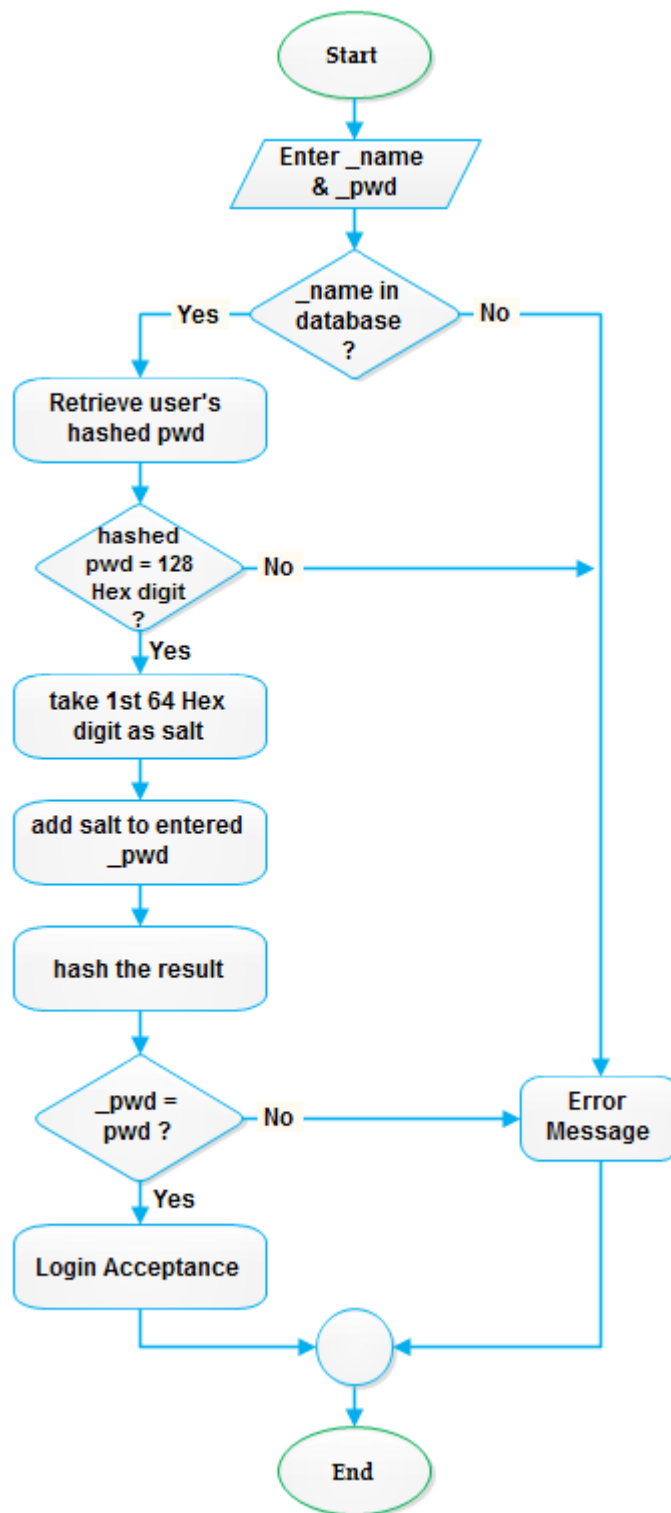


Figure 3.12: Login flowchart.

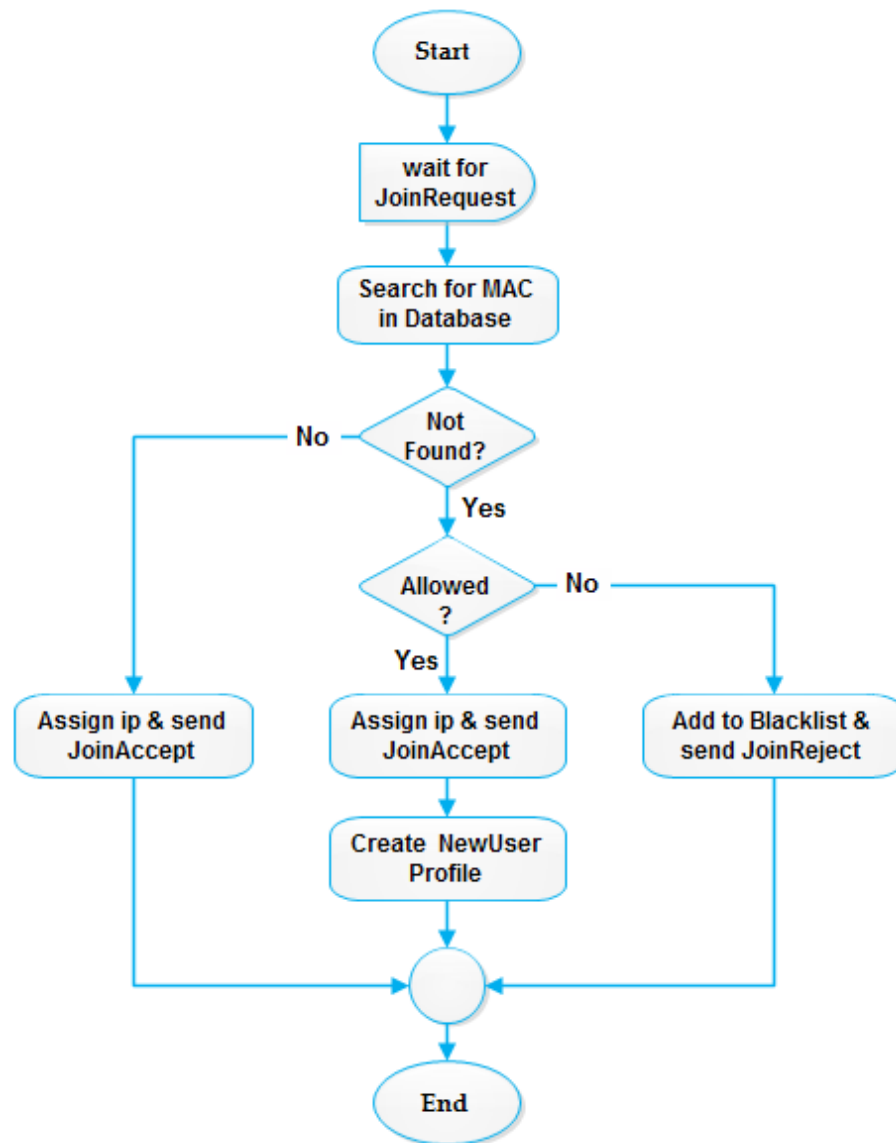


Figure 3.13: The procedure of joining the network.

When the doctor wants to add new patient he will give him a mobile node to hold. The node will discover the network and ask to join through a *JoinRequest* packet. The system will receive this request and search for the MAC of that mobile node in the database. If its exist it can join the network, otherwise the system will ask the admin to confirm this process. If its allowed, the join request will be accepted and the new patient will be added. However, if it is not allowed (i.e. The doctor did not ask to add new node) then its MAC address will be added to a block list and its request will be rejected.

Logger file In order to know some important actions happened to the system and to have the ability to return to it later, a *logger file* will be used. The logger file will save the login and logout information, the addition and removal operations, and the actions that are done in the system in general.

Separation of duties To have more reliable system, the user can access the information which are relative to him, so that not all system users can access every piece of information in the system. In this since, the doctor can use the monitoring options, updating locations, tracking patients, add notes on patient profile and view it. The doctor cannot control fixed nodes information or viewing the logger file.

ZigBee secure network ZigBee Networks can be highly secured since they have a built in 128-bit AES encryption algorithm (see section 2.3) which can be enabled using the **EE command**. When this command is enabled, all transmitted data will be encrypted by a sixteen byte encryption key so that only devices that have the key can communicate with the network.

3.2.3 Database

In order to store and organize the collected data in the system, a database is needed. The database is divided into two parts: the first for application, which is used by doctors in the hospital, and the other is for data used in the training process for neural network. Neural network data consist the power values taken from reference nodes with mobiles corresponding locations. Figure 3.14 shows the *ER model* for this part of the data base.

The ID is the key value for the data values; its a sequential number for this data which will be taken to train the network. We will use N anchor nodes to find the corresponding location in terms of the four coordinates (x , y , z and θ). The database contents for monitoring application are the patients, doctors and nodes data, as the following ER model shows (Figure 3.15).

Patients information consists the patient ID and name which are unique for each patient in the database. Also for each patient we have a picture, illness type, date of birth, data come in, date went out and the mobile node MAC address which he/she holds. For each patient we keep the name of the doctor who registers this patient and notes about this patients which helps in monitoring his progress. The *doctor information* consist of doctors name and ID, his

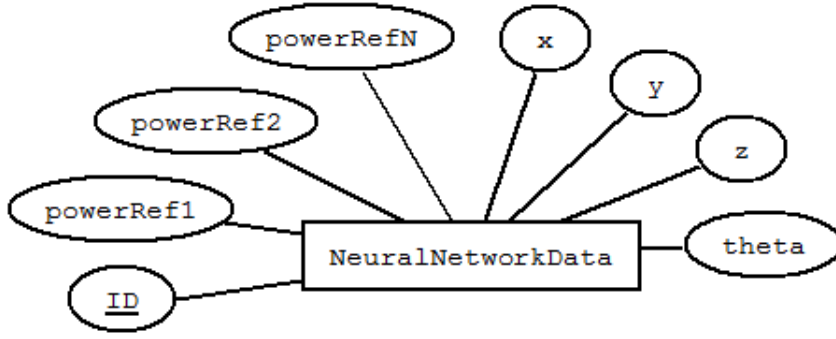


Figure 3.14: ER model for database used to train the neural networks.

picture and the MAC address for the mobile node which he holds and some information for the application security like the username and the password. Moreover, the last login and last logout times which will be saved in the logger file. The *nodes information* contains the MAC address for the node which is unique for each node, node ID, node type (to show if it is mobile, reference or master node), the floor where its placed especially for fixed nodes and the location information (x , y , z and θ) which comes after power calculation to be placed on the map. After normalization of ER-module the following tables are resulted as shown in figure 3.16.

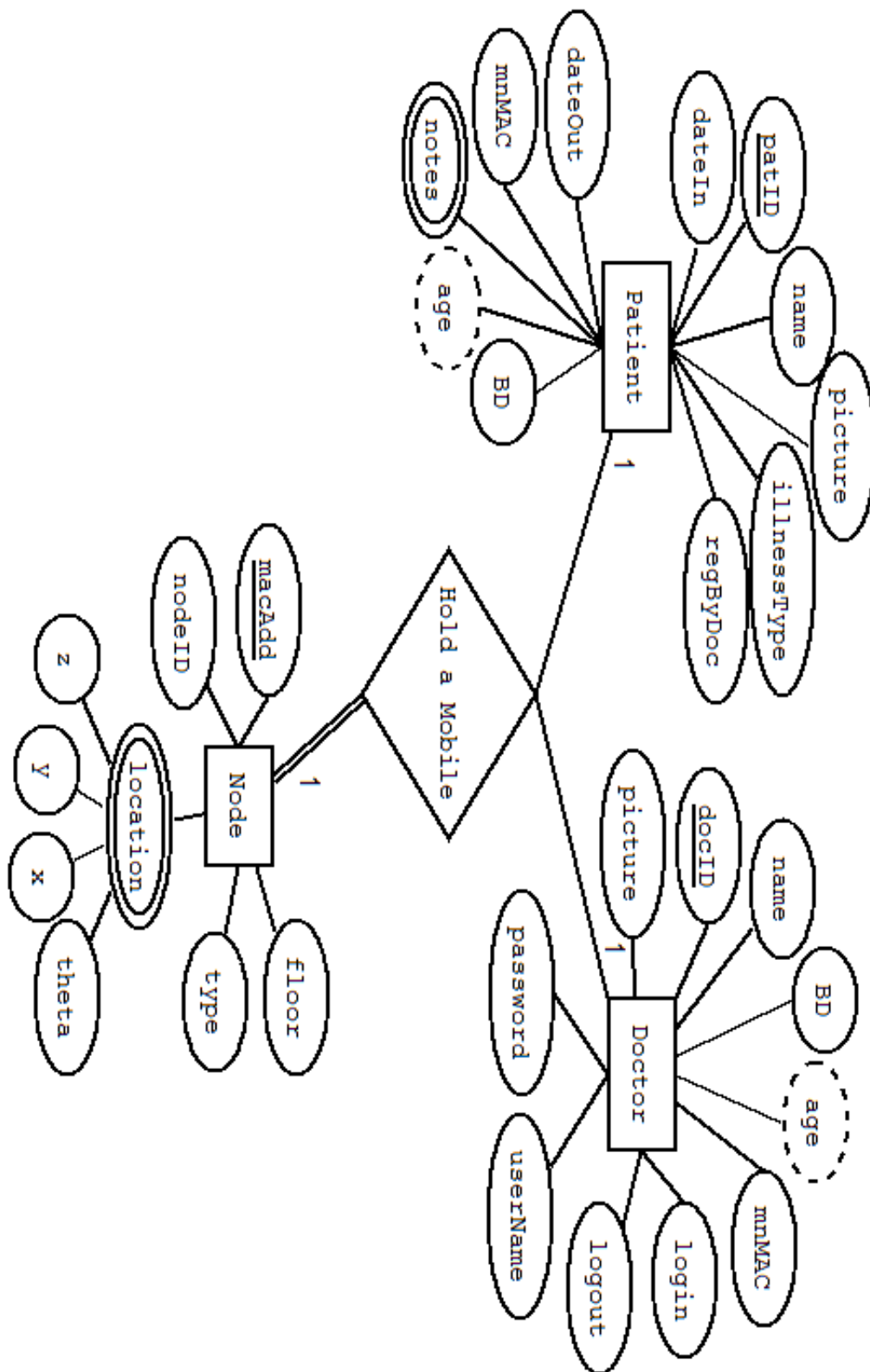


Figure 3.15: ER model for system database.

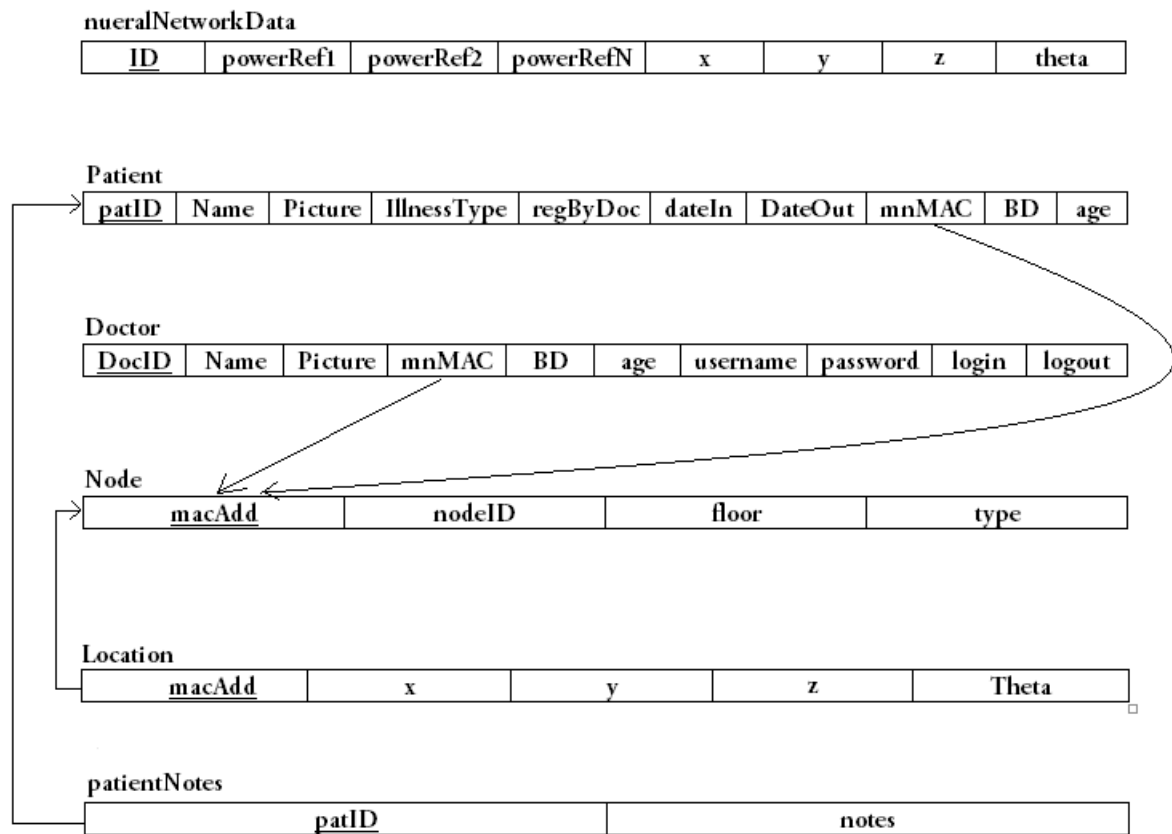


Figure 3.16: Normalized logical models: for neural network (above) and patient (below).

Chapter 4

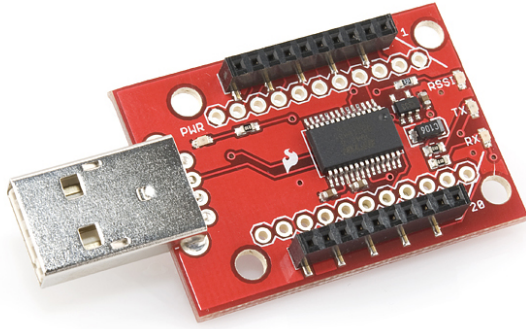
Hardware Implementation

In chapter 3, we discussed the conceptual design of the system in details including the idea, the main system parts, flow charts that describes their interaction with each other, and the algorithms that will be applied to determine patients' locations in the treatment center.

In this chapter we will dig more deeply in the implementation side of the project. Section 4.1 will focus on the electrical considerations in the design including the main circuit diagram, electrical calculations, and will discuss other possible variation of the design beside the advantages and disadvantages of each one. The electrical design should support all specified requirements for the localization algorithm to work well. However, in section 4.2 we will address the main hardware limitations that require some modifications on the proposed algorithm. The new algorithm itself will be explained here while its software implementation will be leaved to the next chapter. The last section of this chapter was dedicated to describe the method of collecting RSSI data. This is the first step in the localization algorithm and will be done outside the MATLAB, so it was included here as a transition from the hardware to the software chapter. Note that the datasheets of all used components are given in the appendix A while the complete software code is available in CD provided with the documentation.

4.1 Detailed Circuit Design

As mentioned in section 3.1, the system consists of three types of nodes: mobile node, reference node and the coordinator. All nodes, in general, share the same structure: a *power source*, an *XBee transceiver* and a *microcontroller* unit (MCU). The circuit of the coordinator is the simplest one, we used an XBee explorer dongle (shown in figure 4.1a) to connect the XBee to the administrators laptop which serves as its MCU. The dongle has two main functions: firstly it is used to interface the XBee to the laptop by converting serial data coming from XBee to the USB format (with proper voltage level translation) to be readable by the computer. Secondly, the dongle contains an inherited voltage regulator at 3.3V to power the XBee module from the laptop. Figure 4.1b shows the coordinator nodes components connected together.



(a) XBee explorer dongle.



(b) XBee, PC and the dongle all together

Figure 4.1: Coordinator node.

On the other hand, the power is fed to the fixed node through an AC-DC power adaptor with 7.5V and a maximum supplied current of 250mA. This voltage is passed through voltage regulation stage to power the XBee and PIC MCU. According to their datasheets (refer to appendix A, the XBee module operates at 2.8-3.4V while PIC is known to work at 5V. From this point we have had two choices: either to use two separate regulators to power XBee and PIC on their suitable voltage levels, this choice means we will need a voltage level-shifter circuit (usually a transistor with a resistor) between serial port of XBee and serial port of the PIC.

The second choice, which we followed, is to choose a CMOS-type PIC MCU. The advantage of using a CMOS MCU is that they operate at low voltage (from 2.2-5.5V) which is very suitable in our case. Note that this solution does not only reduce the cost of the node by using a minimum components but also keep the board size more compact. Figure 4.2 shows the circuit diagram of the fixed node, the TS317 variable regulator is derived by two resistors of 110 and 180 ohm to provide the desired 3.3V output voltage according to the following equation:

$$V_{out} = 1.25V * (1 + \frac{R_2}{R_1}) + I_{adj} * R_2 \quad (4.1)$$

The presence of two capacitors is optional to keep voltage stable at the input and filter out the ripples at the output. As a general rule, when external capacitors are used with any I.C. regulator it is sometimes necessary to add protection diodes to prevent the capacitors from discharging through low current points into the regulator. The protection diode is recommended for output voltages in excess of 25 V or high capacitance values. However, since we are dealing with low rating powers we ignore this diode.

The second component in the FN is the XBee module, it is 20 pin transceiver with only four pins are of our interest which are Vcc, GND (power lines) and Dout, Din (serial lines). An additional optional pin will be used to indicate a packet reception on from the wireless

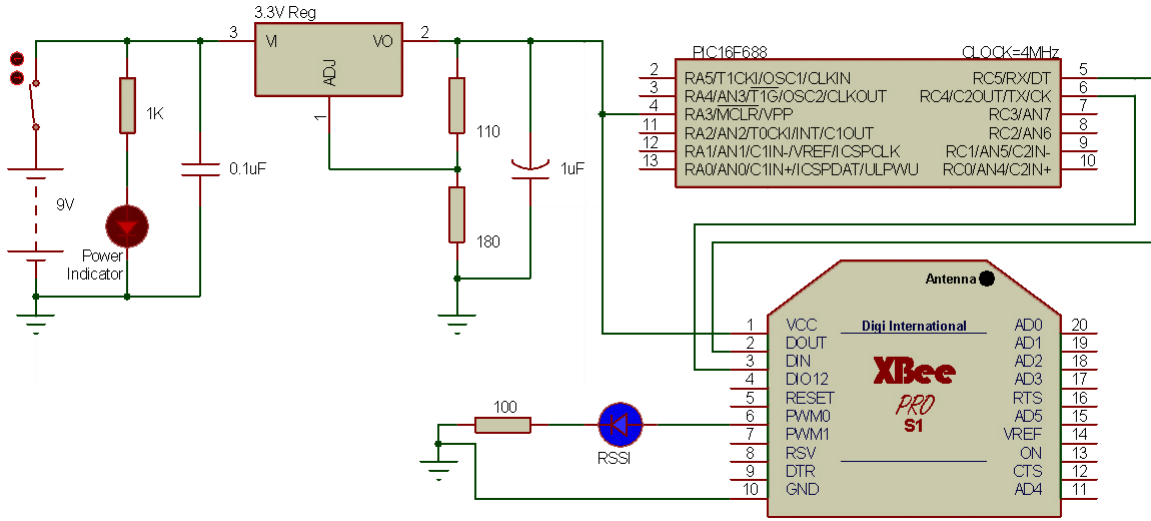


Figure 4.2: Circuit diagram of the Fixed Node.

link which is pin 6 (PWM0). In fact, through this pin XBee module generates a pulse width modulated (PWM) signal with the pulse frequency is varying according to the RSSI value of the received packet from the last hop as explained in chapter 3.1.2. However, we will not attempt to extract the needed RSSI readings from this signal since it requires additional hardware which means additional costs and board size. The RSSI extraction will be done through software and will be explained in section 5.5.

The last part in the fixed node is the PIC MUC. From our previous discussion we set the requirements for the needed MCU: it should operate at 3.3V, and have a serial port with minimum number of pins to ensure smaller size. As in XBee, all what we need is five pins, two for power lines (pin1 is Vcc and pin 14 is GND), two for serial communication (pin 5 is Rx and pin 6 is Tx) and the last one is master clear (pin 4) which we maintain disabled (i.e. at level high). Now since PIC12F683, which has 8 pins, do not has UART port we fined PIC16F688, which has a 14 pins, to be good choice. Even if it is possible to program any two pins to serve as serial Tx and Rx, we do not prefer this choice since the needed program will occupy some valuable space in the limited memory of MCU. Furthermore, Software Serials are known to be slower than the Hardware UARTs.

One beautiful feature of this PIC16F688 is being able to operate at 4 MHz through its internal oscillator without much timing problems, this means that we were able to free two pins and get rid of the external crystal and its two 10pF capacitors to reduce the cost even further and gaining a smaller board size. However, the baud rate at which PIC communicates with XBee through the serial port is directly affected by the chosen clock. For asynchronous mode, the

desired baud rate is given by the equation:

$$BaudRate = \frac{F_{osc}}{64(BRG + 1)} \quad (4.2)$$

Where BRG register stores 8-bit number that controls the baud rate of the PIC. And the oscillator frequency is four time the operating clock frequency. The default baud rate for serial communication is 9600. Let say we want to find the actual baud rate, then solve for RGB value

$$RGB = \left(\frac{F_{osc}/Baud\ Rate}{64} \right) - 1 \quad (4.3)$$

$$RGB = \left(\frac{16000000/9600}{64} \right) - 1 \approx 25$$

Now calculate the actual baud rate using equation 4.2, the result is 9615 with error given by

$$Error = \frac{Calc.BaudRate - DesiredBaudRate}{Desired\ Baud\ Rate} = 0.16\% \quad (4.4)$$

This small drift in timing did not cause much problems if we make sure that our transmitted packets through serial ports are short. A 0.16% error means that every 640 bits (i.e. 9600/15) there is one additional bit (i.e. one error occur every 80 byte). However, our packets do not exceed 50 bytes at any circumstances so we are safe. One additional thing we should mention here regarding the baud rate is that, in general, higher baud rates means higher error rates; since the timing becomes more critical and required a very stable crystal oscillator. In this situation, external oscillators are recommended instead of the internal oscillator because they are more stable and precise. However, we have noticed that operating at a baud rate of 19200 (i.e. at double speed) do not cost us any additional errors. The actual baud rate at 4MHz clock is 19231 with error of $19231-19200 / 19231 = 0.16\%$. For that we use this baud rate instead of 9600 to reduce the required time between two successive location updates.

After exploring the advantages of using PIC16F688, it is worth to mention one of the most limitations that this PIC suffers from. As we state before, the communication between XBee and PIC MCU will be carried on the UART port. Since the communication is asynchronous, neither the PIC nor the XBee knows when the data will arrive at its port. So there is a substantial need to have a buffer at each side to store the incoming data until the device become ready to process it. In fact, the XBee module has a receive buffer of 250 byte at its serial port as well as at its RF receiver. This mean XBee module can store 250 bytes from PIC before transmitting them wirelessly to the specified destination, or it can receive an RF packet from a remote XBee and store it until the PIC become ready to receive it through the UART port. However, the buffer size of the PIC is only one byte! This means that if a byte was received from XBee and not fetched from the buffer before the next byte arrival, the first byte will be lost and then the whole packet will be processed in wrong way. This problem will be discussed further when we talk about the software part of the project in chapter 5.

For mobile nodes, the whole previous discussion is valid with one difference. Mobile nodes are powered by a portable power source instead of the AD-DC power adapter to support patient mobility. This requirement add some constrains and limitations regarding power consumption as same as the source voltage regulation.

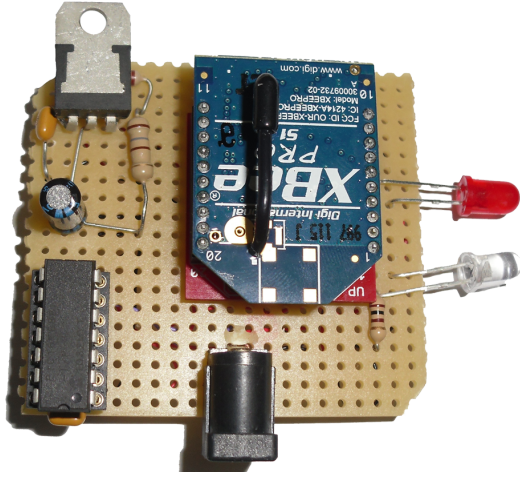
There are many types of batteries available in the market with a wide range of DC voltage values (from 1.5V-9V and more). The cost of a battery is determine by the rating voltage and current it can support, and is affected by other features such as being rechargeable or not. Since our mobile node is running at about 3V, the first attractive choice is to power mobile node with 2 batteries of 1.5V in series. This choice eliminate the need of using voltage regulator which is a big advantage. However, using this type of batteries has some disadvantages regarding the stability of the output voltage level with time.

Batteries are chemical power supply sources that produce fixed DC voltage through a chemical reaction. The output voltage remains stable as long as the reaction is active and strong enough. The voltage then starts to drop with time until the battery is down. Since we need voltage to be maintained at fixed level all the time, we switch to 9V batteries. A 9 V battery with a voltage regulator will provide us with a fixed regulated voltage at 3.3V even if the battery voltage drops from 9V to 6V which is good enough for our project. Note that a minimum acceptable voltage threshold is set to 6V since the voltage regulator requires about 1.5V margin between input and output voltages to ensure proper voltage regulation.

The amount of current being consumed by the mobile node components is another important factor that affects the life time of the used battery. XBee transceivers and PIC microcontrollers are the main parts that consume current during their operation. Some current is also wasted across other components whether it is in the regulation stage or even through connection points on the soldering board.

As a mobile node, the PIC consumes about 1mA when it operate at 3.3V and 4MHz internal oscillator. The XBee typically consumes 55 mA when it is in the receive or idle mode, and up to 250mA during transmission if max power level is used ($63\text{mW} = 18\text{dBm}$). However, we restrict mobile node to Very Low mode which equal to 10dbm (i.e. 60mA). Since the red LED indicator (which is used to indicate the on state of the module) is directly connected to the 9V power supply, it needs a 1Kohm resistor for protection which will drain the battery very quickly. For that we chose to exclude this led and use the blue LED, which is used to indicate packet reception, to indicate that the module is on. This is valid since a successful packet reception also indicates that the battery is connected and can provide enough voltage to power the mobile node.

Methods of connecting electrical components is one final issue needed to be discussed. After assembling and testing the circuit design of the mobile and fixed nodes on a breadboard we become ready to implement the circuits permanently through *soldering* or *wire wrapping*(WW).



(a) Circuit - Upper view.



(b) As a final product - Ready for deployment.

Figure 4.3: Fixed node implementation.

Both techniques ensure fine connection with two advantages of the WW over the soldering. However, we use both techniques in our implementation.

WW is a technology used to assemble electronics. It is a method to construct circuit boards without having to make a printed circuit board. Wires can be wrapped by hand or by machine, and can be hand-modified afterwards. WW construction can produce assemblies which are more reliable than printed circuits: connections are less prone to fail due to vibration or physical stresses on the base board, and the lack of solder precludes soldering faults. The connections themselves are firmer and have lower electrical resistance due to cold welding of the wire to the terminal post at the corners. This means a lot of current losses can be avoided using this technique. Figure 4.3a shows the complete design of a fixed node while figure 4.3b shows the fixed node as a final product.

4.2 Some Modifications on the Localization Algorithm due to Hardware Issues

Since the wireless channel is very complicated in the indoor environments, it is known that the signal strength of a wireless signal does not related to the travelled distance through a clear relationship. Even though in literature we assume to have weaker signals at farther distances, this rule is not valid in general since the variation in RSS due to fading is the dominant effect over the normal attenuation effects. These observations will be understood in a better way after we plot the RSSI values for each location from each FN. These plots will be the subject of the chapter 6.

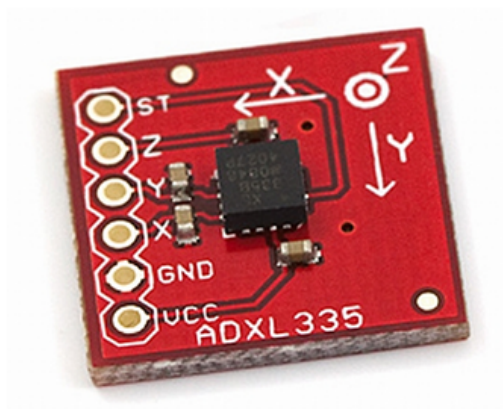
In our localization algorithms we based on two simple assumptions: the first assumption state that in order to predict the coordinates of a MN it does not matter how strong or weak the received signal strength at his specific location. The most important thing is to have a unique RSSI pattern at that location. The trained NN should be able to map this pattern to that location when localization is performed. Note that this is true as long as the same power level is measured every time if it is read from the same location. Any variation in the power level should be handled before mapping process is performed. We add a step to the localization algorithm to try to reduce the variation of the power due to external moving objects which is explained in the next section.

The second assumption states that the collected RSSI values from a MN at multiple FNs will be highly affected by the direction of which the patient is looking to. This is valid since the patients body will act as a fixed attenuation source. The problem with this attenuation is being random because we could not predict which FN is behind the patient and which FNs have a clear LOS with his attached module during his movement. So in order to have better mapping between RSSI and patients coordinates we should firstly determine his movement direction and used one of four separate NNs to perform the mapping. Those NNs are assumed to be trained separately on RSSI values that collected from four direction (i.e. measure RSSI while looking at the North, east, south and west) at all location.

In order to ease predicting the direction of the MN movement, a directional accelerometer sensor (ADXL335 figure 4.4a) was attached to the mobile node circuit. According to the datasheet the sensor can measure the static acceleration of the gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration. We assume we can measure the acceleration in the x and y directions: if the x has a positive value we can transform this to one of the fourth direction (say north). If the value was negative this will indicate the motion is in the opposite direction (i.e. south in this case). The same thing was assumed to be valid for the y-direction.

However, after we programmed the PIC microcontroller and tested the sensor, we found that the sensor readings wasn't useful because the sensor wasn't able to distinguish between the four directions of movement. It measures tilting acceleration only which have no meaning in our project. Figure 4.5 shows two sensor readings while it was aligned to the north and east. The reading was taking by the PIC in the response of receiving an *update* command. The data then was transmitted wirelessly from the MN to the coordinator and were presented using X-CTU. Note that there is no real difference between the readings with the direction which is bad!

When we looked for other alternative sensors we found a more complex sensor called "Compass Module with Tilt Compensation - HMC6343" (shown in figure 4.4b). It consists of 2 directional sensors with embedded PIC that performs the required complex directional calculations. In fact, this sensor is more expensive (five times the price) and is sensitive to the tilting angle. it



(a) ADXL335 sensor.



(b) HMC6343

Figure 4.4: Two Directional Sensors.

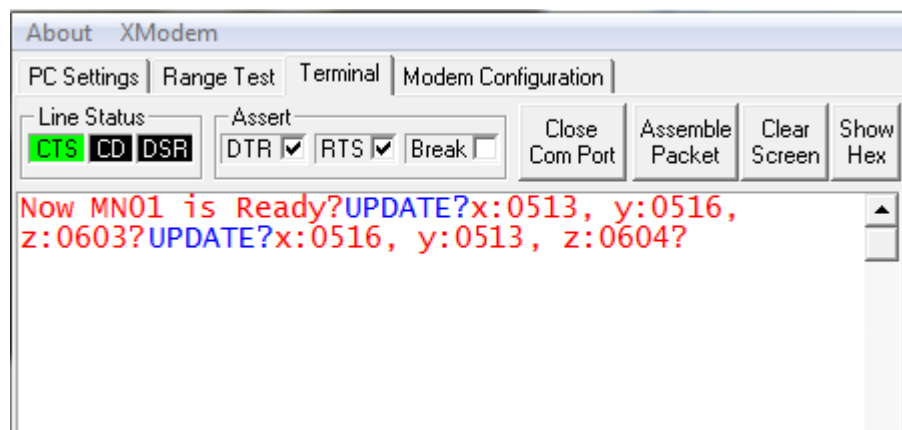


Figure 4.5: Two ADXL335 readings while the sensor was aligned to the north and east respectively.

requires to be parallel with the earth surface to provide accurate results - and this, of course, is not guaranteed during patients' movements.

In order to overcome the weaknesses that appeared due to being unable to determine the direction of MN movement, we suggested two modifications to the localization algorithm. The first one is related to the NN training while the second one is related to the XBee sleeping feature.

Before we trained the NN, we had collected four RSSI readings from each location during the offline phase. Each reading was measured while the MN was looking at one direction (from north and rotating 90 degrees counterclockwise). Now in the online phase if the mobile is known to be directed towards the north (for example), we'll use the NN which was trained on all RSSI that were collected while the MN was moving towards north. However since the sensor was not able to determine the direction of the MN movement, we chose to average these RSSI data to produce a non-directional RSSI dataset. One NN was trained and used to determine MN coordinates.

Losing the information about MN direction will lead to a low localization accuracy. For that we suggest another modification to the algorithm to enhance the prediction error. The idea was to correlate the previous calculated location of a MN to the new calculated coordinates based on the current RSSI measurements. This is valid if we keep measuring RSSI of the MN in relatively short time intervals where the location does not change by a large distance. In other words, we need to keep *tracing* the MN instead of only *updating* its location from time to time using the power values only. However applying this option will contradict with the idea of keeping the MN in the sleep mode except when its location is required. Now we need to compromise between saving MN power and having a better localization accuracy. We finally decided to disable the sleeping feature.

In the next section we will describe the fingerprinting steps in details.

4.3 RSSI Collection (Fingerprinting stage)

As mentioned before, the first and hardest step is collecting the RSSI fingerprint from the location of interest. We study the design of treatment centers dedicated for mentally retarded patients. A typical treatment center consists of one floor divided into rooms by small partitions rather than thick walls. We chose two different places in the university with different structure to simulate the hospital: the first is the sixth floor of B-Building of the Engineering Department, while the second is Abu-Aishah Mosque in the university square. We are interested in studying the effects of walls on the RSSI pattern being inside.

This mosque represent the case where no walls or physical separators exists. It is 17-by-18 m and has an open area with only some columns inside. We use a combination of 1 MN, 3 FNs and 1 coordinator to collect RSSI values in the offline phase. FNs was distributed in triangular shape with the master node at the center of the triangle. A LOS link between MN and FNs can easily be found in this open area. The values of RSSI was expected to be distributed according to the power-distance relationship described by general pathloss models. However, the existing furniture with sharp edges and extensive amount of decorations will cause heavy multipath and fading effects. The localization is expected to be more difficult under this environment since the RSSI values assumed to be close to each other regardless the location they were took from. This analysis will be more clear when we plot the heatmap of the RSSI readings. The main step we have followed to collect data are:

- We've Divided the mosque into grid of a cell area equal to one meter square. Then weve assigned a relative x,y-coordinates to each cell. The total number of segments was 306.
- Then we've distributed the fixed nodes in the mosque in triangular shape (2 FNs at north-west and north-east corners and the last FN in the middle of the other two corners at the south - near the Mehrab). The used transmit powers are high enough to assure a good coverage to the whole location from all three FNs. We try to avoid LOS between FNs but the area was widely open and this condition was not satisfied.
- Finally we've started from north-west corner near FN1 and took 4 RSSI values from the 4 directions at each segment. Each RSSI value was measured at west, north, east and south.
- At the coordinator, 6 RSSI values will be received as a result of one update command. Three of them are the RSSI values that are measured at FNs after receiving MN packet. The other three represent the RSSI between FNs themselves.
- The total number of RSSI values at all locations was $18(\text{along } x) * 18(\text{along } y) * 4(\text{direction}) * 6(\text{readings}) = 7344$ RSSI value.

The idea behind measuring the RSSI between fixed nodes is that in the offline phase no mobile objects or people was allowed to move. The offline phase was aimed to copy the nature of the RSSI data from the mosque to see how the RSSI is distributed as a result of the mosque structure itself. Any variation in the measured RSSI will definitely occur from the unique MN movement (i.e. the variation of the received power should be function of location and nothing else). However, when we move to the online phase, the environment will change due to people who may move around and obstacles may exist. When the MN move and his RSSI is measured, the RSSI will change as a function of his movement as same as the movement of his surrounding environment. Now we need eliminate the variation component of the environment before we predict MN location by observing the variation in the RSSI - and this is the job of the second 3 RSSI values.

Table 4.1: Packets flow to collect RSSI data - Senario 1.

| Source | Destination | Type | Notes |
|-------------|-------------|-----------------|---|
| Coordinator | MN | Broadcast | Update packet. |
| MN | FNs | Broadcast(1hop) | Sensor readings. |
| FNs | themselves | local | read RSSI of MN. |
| FNs | Coordinator | Unicast | send MN RSSI one FN after another. |
| FNx | FNy | Unicast | packet to read RSSI between FNs, where $(x,y) = (1,2), (1,3), (2,3)$ |
| FNy | itself | local | read RSSI of FNx |
| FNs | Coordinator | Unicast | send FNx RSSI. |

In both offline and online phases, FNs location is fixed and well-known to us. In the offline phase we measure RSSI between FNs and keep it for the online phase. Now when a MN location is required while it is moving in a dynamic environment with many moving obstacles, we measure the RSSI between FNs again and compare it the RSSI taken in the offline phase. Any change in the power will be caused by the moving obstacles since FNs themselves did not move. The difference between 3 RSSI values taken between FNs in the online and offline phases will be added to the 3 RSSI values of the MN measured by FNs in the online phase as a correction factor to eliminate the effect of the environment of the MN. These new RSSI values will then be used in NN to predict MN location.

One final, but important, design consideration is the amount of required time to complete one location update. This is important since the maximum required delay determine the location update rate (which must be at least one update every 2 seconds). In our implementation we study three possible variations of packets flow to collect the required 6 RSSI values. Table 4.1 shows the first possible scenario. All FNs are using unicast transmission to deliver their RSSI data to the coordinator. This produces a large number of packets which short length. Loosing FN packet means losing only one RSSI value, but the cost is relatively high delay. The delay was found to be 1600 ms max and is expected to be increased if the number of FNs was more than 3 FNs. If we want to use this strategy, we expect that the system will not be able to track more than two MNs at the same time.

One simple modification to the aforementioned stratigy can save some valuable time. This can be done by using broadcast messages to deliver RSSI data from FNs to the coordinator, and in the same time use this packet by other FNs to measure power of that FN. However, since broadcast is not a reliable communication technique and do not involve acknowledgements, we may loose some RSSI readings if a collision occurs somewhere in the network. For that we do not consider this solution. The third scenario was chosen since it combine the advantages of both scenarios; it is reliable and quick! Table 4.2 shows packets flow. In this scenario we use unicast packets to assure its delivery. However, we collect all RSSI data from all FNs in FN2

Table 4.2: Packets flow to collect RSSI data - Senario 3 (is implemented).

| Source | Destination | Type | Notes |
|-------------|-------------|-----------------|--|
| Coordinator | MN | Broadcast | Update packet. |
| MN | FNs | Broadcast(1hop) | Sensor readings. |
| FNs | themselves | local | read RSSI of MN. |
| FN1 | FN2 | Unicast | send MN RSSI measured by FN1. |
| FN1 | FN3 | Unicast | packet to FN3, meanwhile FN2 read RSSI of FN1. |
| FN3 | itself | local | read RSSI of FN1. |
| FN3 | FN2 | Unicast | send RSSI measured from MN and FN1. |
| FN2 | coordinator | Unicast | send all collected RSSI measured from MN, FN1 and FN3. |

then send them all to the coordinator in one and relatively longer packet (note that the packet is still short!). The anatomy of these packets will be described in details in section 5.5.

After we discuss fingerprinting process in the mosque, we should make some notes on the fingerprinting implementation of building B of Engineering department. It follows the same rules we followed in the mosque. We took 2592 RSSI reading from 108 locations as shown in the figure 4.6. The green circles represents the FNs locations. The height of the FNs was about 1m to make sure that any moving person will affect the RSSI values between FNs. So that we can reduce the effect of surrounding peoples around the MN in the online phase. Note that the location of coordinator is not important since we don't carry any RSSI measurements on it. This gives the doctor the ability to move freely in the hospital while being able to run the monitoring system without affecting the accuracy.

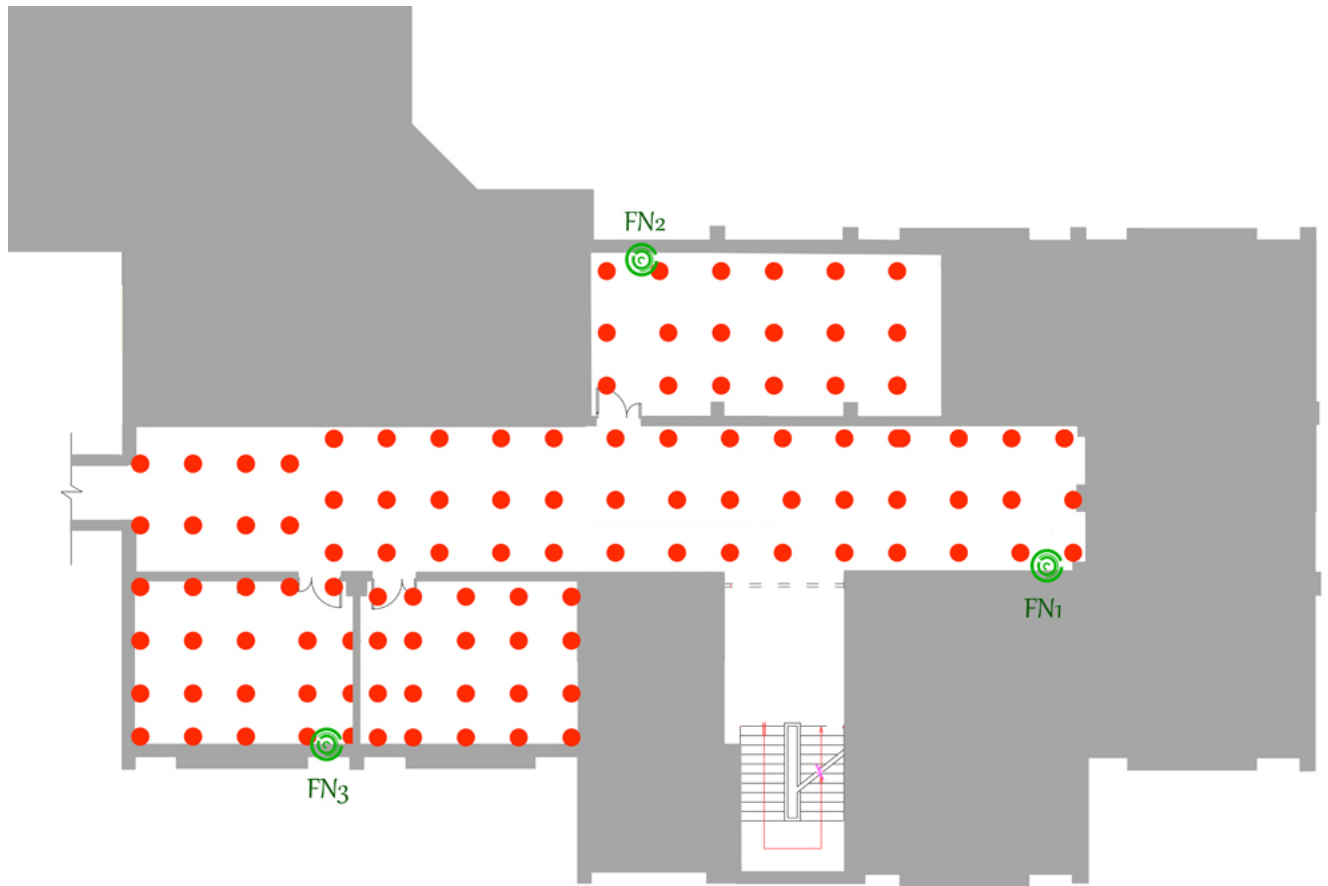


Figure 4.6: Fingerprinting process on the 6th floor of B-Building of Engineering Department.

Chapter 5

Software Implementation

In this chapter we will focus on the software part of the project; whether the software code was used to develop the graphical user interfaces (GUIs) or being implemented at lower levels for processing purposes. The power of three programming languages was combined, and in sometimes was compared, in this project to get all possible advantages of the underlying hardware. The result is simply a powerful and reliable system dedicated to develop, compare and test different localization algorithms for both indoors and outdoors based on ZigBee wireless technology.

The main software was implemented on the admin laptop. It contains GUI and connects all the monitoring system parts together including database, L&TE and ZigBee coordinator node. On the other part of the network, each fixed and mobile node has a customized version of the C code for its PIC MCU that defines its role in the localization process according to the applied algorithm.

5.1 The main software - GUI

The GUI constitutes the primary interface between the user and the monitoring system. Stemming from the advantages of C# over the other programming languages mentioned in chapter 2.5 we chose to build the system in this language. However, this decision was made while MATLAB being out of the game. In fact, we had discovered that MATLAB has a powerful GUI toolbox together with other additional new features in the new release such as *Profiler*, which is used to optimize the written MATLAB code to achieve the best performance. So we decided to implement the software in both C# and MATLAB for the sake of comparison. Two parameters are of our interest: code length and execution time.

The main monitoring system window contains two tabs to show the maps for two floors of the building. Although this project will be implemented in one floor, we add this option to support multiple floors in the future releases. The main window also contains some options for monitoring like *Update Locations*, *Track Patients* and *Open Selected Profile* buttons. The first

two buttons interact with the L&TE and ZigBee coordinator. The Update Locations process will be executed once when the doctor choose it manually. The whole system will be disabled when the Track Patients option is pressed to execute the update location automatically every some time interval. Note that its more convenient to use *toggle* button for tracking option rather than a pushbutton. However, the toggle button is not implemented in C# so create one by ourselves. Finally the *Open Selected Profile* button interacts with the database and when pressed, it will open the profile of the selected patient in the Patients' List. The list shows all patients who are holding a ZigBee chip (i.e. the available mobile nodes). Figure 5.1 shows the main window in its final form.

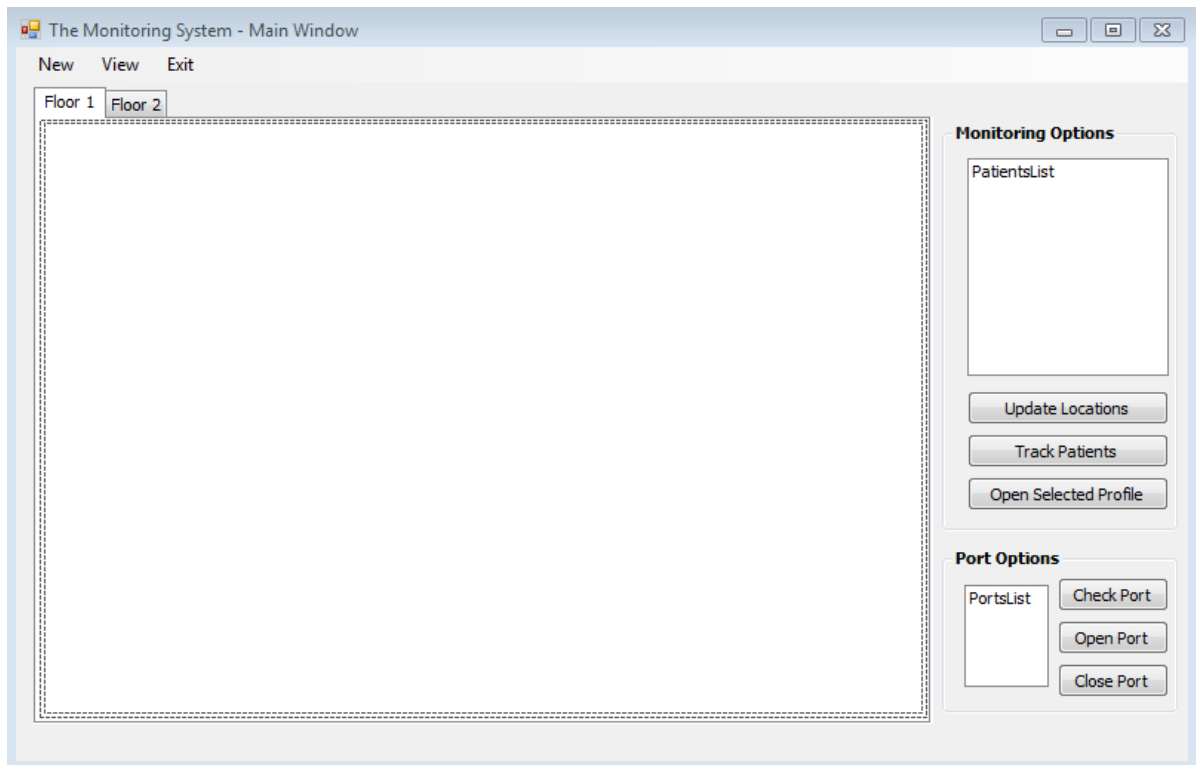


Figure 5.1: The main window of the monitoring system.

Before being connected to the whole ZigBee network, the main software should open a serial port to communicate with the ZigBee coordinator associated with the computer. Once the master node is connected to the main computer and *Check Port* button is pressed, the port at which the master node is connected will appear in the port list. Other ports may appear if additional instruments are connected to the laptop. The doctor then select the desired port and open it through *Open Port* button. A status message will appear to indicate whether the process succeed or not. *Close Port* button should be used to close the serial port before closing the program or unplugging the XBee dongle. The following piece of code is used to prepare serial port.

```

1 private void ChkPort_Click(object sender, EventArgs e)
2 {
3     List<String> allPorts = new List<String>();
4     foreach (String portName
5         in System.IO.Ports.SerialPort.GetPortNames())
6     {
7         allPorts.Add(portName);
8     }
9     PortsList.DataSource = allPorts;
10 }
11
12 private void OpnPort_Click(object sender, EventArgs e)
13 {
14     // configure the port
15     MyPort = new SerialPort(PortsList.SelectedItem.ToString(), 19200, ...
16         Parity.None, 8, StopBits.One);
17     // open port code is here ...
18 }

```

The system also allows the user to zoom in and out using the mouse wheel focused on the mouse place on the map. Also the mouse right button will be used to navigate through the map. Finally, the *Menu* strip (above the map) has three options: *New*, *View* and *Exit*. The first option is used to add patients, nodes and doctors to the system. The second one is used to view all parties' names that are stored in the database. The third menu is used to logout from the monitoring system. All these components and menus are used to control the rest parts of the monitoring system. In fact, the system has been established using many other classes beside this main monitoring class. These classes interact with each other in order to make the system working properly. These classes will be described throughout this section.

1 - Login class

This class is used when the user starts the program. In this form the user is asked to enter his username and password in order to enter the main window of the monitoring system. According to his/her account type, different controlling options will be available to the doctor. The user *System*, which represent the system admin, have the permission to access all the system parts, while the other users have a less accessibility to the system. Figure 5.2 shows that login window.

When the user presses the *Login* button (or press *Enter* key from the password field) the *login_Click* function will be called. The function starts by searching for this user name in the database. If it does not exist, an error message will appear to indicate that either the username or the password is incorrect. If the user name is correct, the password correctness will be checked by retrieving this user password from the database; the retrieved password is hashed using SHA256 algorithm with its salt stored before it as described in section 2.4. The salt and the hashed password will be separated from each other; the hashed password will be kept while

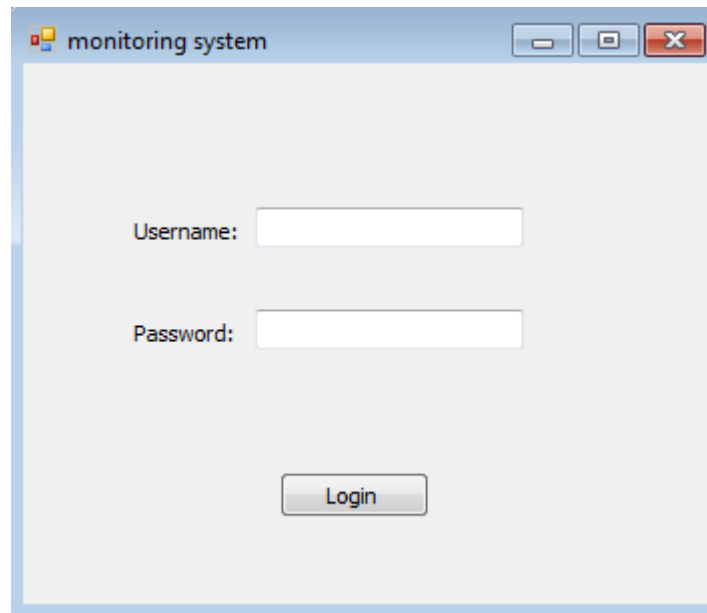


Figure 5.2: The login window.

the salt will be sent to the hashing code with the entered password. This will be done using the following piece of code:

```
1 private void login_Click(object sender, EventArgs e)
2 {
3     // check if username is exist in the database ...
4
5     // Read dr password from the DB
6     pass = dr.password;
7     string salt = pass.Substring(0, 64);           // get the salt
8     string hashedStr = pass.Substring(64, 64);    // and hashed password
9     string pa = enteredPassword.Text;            // and userentered ...
10    password
11    string str1 = hashingClass.Sha256Hex(salt + pa); // send to hashing...
12    function
13    // compare enteredPassord with hashedPassword ...
14 }
```

After getting the enteredPassword hashed, it will be compared with the retrieved password from the database, if they are the same the monitoring window will be opened and this action will be stored in the logger file for this username with corresponding system date and time. if the password didnt match, an error message will show that the username or password is incorrect.

2 - Hashing class:

This class is used when login process is in progress to check the password correctness, or when adding a new doctor to store his password. The process of adding a new doctor starts by getting the random salt, followed by sending the password and its salt to be hashed. However, while checking the password correctness, we sent the password to be hashed using the function Sha256Hex with its stored salt. The following functions describe the main process for hashing class:

```
1 public static string HashingProcess(string password)
2 {
3     string salt = GetRandomSalt(); // Returns a rand 64 char hex
4                                     // string (256bit)
5     string hash = Sha256Hex(salt + password);
6     return salt + hash;
7 }
8
9 //returns the SHA256 hash of a string, formatted in hex
10 public static string Sha256Hex(string toHash)
11 {
12     SHA256Managed hash = new SHA256Managed();
13     byte[] utf8 = UTF8Encoding.UTF8.GetBytes(toHash);
14     return BytesToHex(hash.ComputeHash(utf8));
15 }
```

3 - Add patient class:

When the user presses on *Add Patient* option in the menu strip in the monitoring form, the following form will appear (see figure 5.3). The user should fill in the patient information to be added. The *Date Entered* and the *Age* are 'read only' fields since they will be filled automatically by the system. The date at which the patient is entered to the system will be taken from the system date once this form is opened. Similarly, the age will be calculated after the birth date field is filled. When the user enters an invalid card ID, an error sign will appeared using these code lines:

```
1 if((IDcard.Text.Length != 9 ||
2     !(Regex.IsMatch(IDcard.Text, @"[0-9]{9}"))) && IDcard.Text != "")
3 {
4     errorProvider1.SetError(IDcard, "ID card must be 9 digits");
5 }
6 else errorProvider1.Clear();
```

Figure 5.3: Add patient form

When the user press on the *Browse Picture* button, an open file dialog will be opened to let the user choose the picture. If he chose one, it will be shown in the picture box and the image path (not the image itself) will be stored in the database (to save space). The *Device MAC* field will allow the user to choose one of the chips which is pre-entered to the database and still not in use (as shown in this line of code):

```
1 PmnMAC.DataSource = dataFromDB.Node_tbls.Where(s => s.used == "No")
2   .Select(x => x.macAdd );
```

When the user click on the *Add Patient* button, all patient's information will be stored in the database. Then a confirmation message will appear to indicate that the patient has been added successfully. Finally, the from will be closed. The following function summarize this process.

```
1 private void AddPat_Click(object sender, EventArgs e)
2 {
3     Patient_tbl newPat = new Patient_tbl();
4     newPat.patID = IDcard.Text;
5     newPat.name = PatName.Text;
6     newPat.illnessType = IllnessTyp.Text;
7     newPat.regByDoc = RegisteredBy.Text;
```

```

8      newPat.dateIN = temp;
9      newPat.mnMAC = PmnMAC.Text;
10     newPat.BD = DateTime.Parse(PatBD.Text);
11     newPat.age = int.Parse(PatAge.Text) ;
12     newPat.picture = path;
13
14     dataFromDB.Patient_tbls.InsertOnSubmit(newPat);
15     dataFromDB.SubmitChanges();
16
17     MessageBox.Show("The Patient has been added.", "Information",
18                     MessageBoxButtons.OK, MessageBoxIcon.Information);
19     this.Close();
20 }

```

4 - Add doctor class:

The *Add doctor* option in the menu strip in the monitoring form will invoke this form to add a new doctor. When it is opened (as in the *Add Patient* form) the device MAC address box will be filled from the non used pre-entered MAC addresses. Also an error sign will appear if the ID card number is not valid. This form can be accessed by the user (System) only. A user name and password should be assigned to the new doctor to allow him to use the monitoring system if needed; when the add doctor button is pressed the doctor information will be taken to be saved in the database while the password field will be sent to the hashing code to be hashed before being stored as this line shows.

```

1      newDoc.password = hashingClass.HashingProcess(password.Text);

```

5 - Nodes class:

The nodes form will be invoked when the user (System) presse on the *Add Node* option in the menu strip. The user must enter the MAC address and select the type for this node which is either Mobile Node or Anchor Node. The node ID will be assigned automatically when the node type is chosen as the following code shows. The default value for the device is *Not Used*; because the user 'System' inters it to be used later. The field *Floor* is important for the anchor nodes to know its place.

```

1  private void nodeTyp_SelectedIndexChanged(object sender, EventArgs e)
2  {
3      if (nodeTyp.SelectedItem.ToString() == "Mobile Node")
4      {
5          var nodes = dataFromDB.Node_tbls.Where(s => s.type == "Mobile ...
           Node")

```

The screenshot shows a window titled 'addDoctor' with a standard Windows-style title bar (minimize, maximize, close buttons). The main content area is titled 'Add Doctor Form'. It contains several input fields and buttons:

- ID card no. :** A text input field.
- Doctor Name:** A text input field.
- Device MAC:** A text input field with a dropdown arrow on the right.
- User Name:** A text input field.
- Password:** A text input field.
- Birth Date:** A date picker showing 'Wednesday, May 15, 2013' with a calendar icon.
- Age:** A text input field.
- Image Placeholder:** A large empty rectangular box on the right side.
- Buttons:** 'browse picture' (below the image placeholder), 'Add Doctor' (below the birth date), and 'Close' (below the 'Add Doctor' button).

Figure 5.4: Add doctor form

The screenshot shows a window titled 'Nodes' with a standard Windows-style title bar (minimize, maximize, close buttons). The main content area is titled 'Nodes Form'. It contains several input fields and buttons:

- MAC Address:** A text input field with a hint text 'example: 0013A200 40919AB5' below it.
- Type:** A dropdown menu currently showing 'Select Node Type'.
- Node ID:** A text input field.
- Floor:** A text input field.
- in use:** A text input field currently containing the text 'No'.
- Buttons:** 'Add Node' and 'Close' at the bottom of the form.

Figure 5.5: Nodes form

```

6         .Select(x => x.macAdd);
7         int MNs = nodes.ToArray().Length;
8         NodeID.Text = "MN" + (MNs + 1);
9     }
10    else // text selected is Anchor Node
11    {
12        var nodes = dataFromDB.Node_tbls.Where(s => s.type == "Anchor ...
            Node")
13        .Select(x => x.macAdd);
14        int FNs = nodes.ToArray().Length;
15        NodeID.Text = "FN" + (FNs + 1);
16    }
17    AddNode.Enabled = true;
18 }

```

6 - Patient profile class:

The form in the figure (5.6) shows how patient profile looks. When a patient profile is opened, all his information will be shown in the corresponding fields. All fields in the patient profile are not changeable (read-only fields), However, if the doctor would like to change some data, the button *Update Data* will allow him to do that by opening the *Update Patient Profile* form. After updating is complete, the changes will be shown on this form directly as the following code describes:

```

1    updatePatPro upPro = new updatePatPro();
2    upPro.ShowDialog();
3    dataFromDB.SubmitChanges();
4    showData();

```

Also the user is allowed to check the notes for this patient by clicking on *View/Add notes* button.

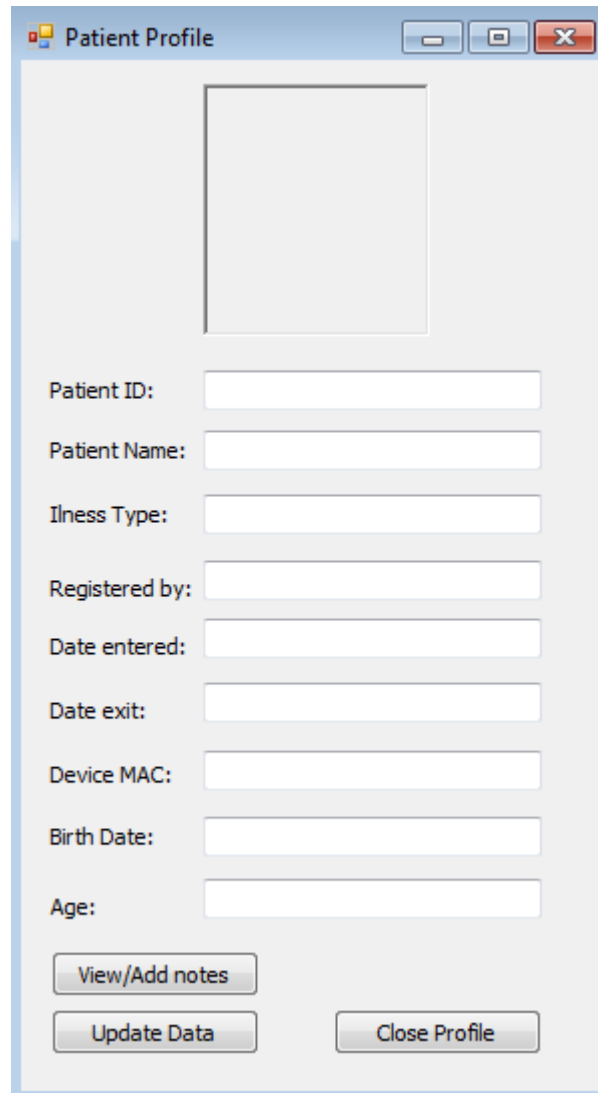
7 - Patient notes class:

When the doctor likes to check or add some notes about the patient condition or progress, the *Patient Notes Form* would allow this to keep up with patients conditions. the following code will be executed when the button Add Note is pressed.

```

1 private void AddNote_Click(object sender, EventArgs e)
2 {
3     patientNotes_tbl newNote = new patientNotes_tbl();
4     newNote.patID = pt.patID;
5     newNote.note = richTextBox1.Text;

```



A screenshot of a software window titled "Patient Profile". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Inside the window, there is a large, empty rectangular box at the top, likely for a patient photo. Below this box, there are several text input fields, each preceded by a label: "Patient ID:", "Patient Name:", "Illness Type:", "Registered by:", "Date entered:", "Date exit:", "Device MAC:", "Birth Date:", and "Age:". At the bottom of the form, there are three buttons: "View/Add notes", "Update Data", and "Close Profile".

| Field Label | Input Type |
|----------------|------------|
| Patient ID: | Text |
| Patient Name: | Text |
| Illness Type: | Text |
| Registered by: | Text |
| Date entered: | Text |
| Date exit: | Text |
| Device MAC: | Text |
| Birth Date: | Text |
| Age: | Text |

Buttons: View/Add notes, Update Data, Close Profile

Figure 5.6: Patient profile form

```

6      dataFromDB.patientNotes_tbls.InsertOnSubmit(newNote);
7      dataFromDB.SubmitChanges();
8      // show notes after adding the new note
9      var dt = dataFromDB.patientNotes_tbls.Where(s => s.patID == pt....
          patID)
10         .Select(x => x.note);
11      listBox1.DataSource = dt;
12 }

```

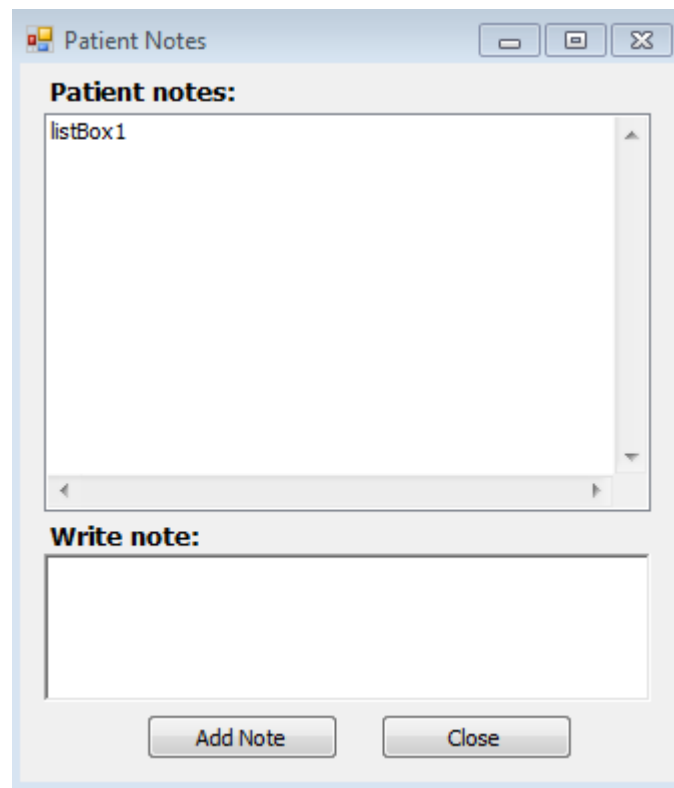


Figure 5.7: Patient notes form

8 - Update patient profile class

When the doctor clicked on *Update Data* button in the *Patient Profile* form, this form will be appeared. The doctor can change some information such as dates or illness type or the device the patient carries. In case it is broken for example, also he can take the chip off if it is not needed to be used by this patient anymore. When the doctor chose to change the chip, the box under it will show the available chips for usage, and the new chip will be connected to this patient while the other will set free. If the doctor chose to take the chip off, it will be set as free

and no chip will be assigned to this patient. This process is described in the following piece of code:

```
1 if (takeOffChp.Checked == true)
2 {
3     Node_tbl MN = dataFromDB.Node_tbls.Single(s => s.macAdd == pat....
        mnMAC);
4     MN.used = "No"; // set the chip free
5     pat.mnMAC = ""; // no chip is assigned for this patient
6
7 }
8 else if (ChngChip.Checked == true)
9 {
10    Node_tbl MN = dataFromDB.Node_tbls.Single(s => s.macAdd == pat....
        mnMAC);
11    MN.used = "No"; // set the chip free
12    pat.mnMAC = PmnMAC.Text; // the new chip is assigned for this ...
        patient
13    MN = dataFromDB.Node_tbls.Single(s => s.macAdd == pat.mnMAC);
14    MN.used = "Yes"; // set the chip in usage
15 }
```

The dates format to be entered is shown next to the text boxes. even so if the user entered it wrong, the software show an error message as shown bellow. After the information is updated, the form will be closed automatically, and the changes will be shown on the Patient Profile form.

```
1 if (DateIn.TextLength != 0)
2     try { pat.dateIN = DateTime.Parse(DateIn.Text); }
3     catch { MessageBox.Show("Date entered wrong format .. " +
4         "changes does not recognized!", "Error!", MessageBoxButtons.OK...
5         ,
        MessageBoxIcon.Error); }
```

9 - Program class

This class is the main entry point for the application. It contains the main function which runs the application as shown below:

```
1 static void Main()
2 {
3     Application.EnableVisualStyles();
```

The image shows a software window titled "updatePatPro" with standard Windows window controls (minimize, maximize, close). The window contains a form for updating patient information. At the top, there is a large rectangular placeholder for a patient photo, with a "Change picture" button centered below it. The form includes several input fields: "Patient Name:", "Illness Type:", "Date entered:" (with a date format hint "mm/dd/yyyy"), "Date exit:" (with a date format hint "mm/dd/yyyy"), "Device MAC:", and "Birth Date:" (with a date format hint "mm/dd/yyyy"). Below the "Device MAC:" field, there are two radio buttons labeled "Change chip" and "Take chip off", followed by a dropdown menu. At the bottom of the window, there are two buttons: "update info" and "Close".

Figure 5.8: Update patient profile form

```

4     Application.SetCompatibleTextRenderingDefault(false);
5     Application.Run(new login());
6 }

```

5.2 Database and Logger files

The system uses logger file to store some actions happened to the system by its users. The logger file stores users following actions: login to the system, logout from the system, adding a new patient, adding a new note for a patient, updating a patient profile, adding a node, and adding a doctor. After logging out from the system two new lines will happen in order to have space between each user actions. The logger file will store the action in the following format:

| date | time | : | action |
|------------|---------------------|---|--------------|
| mm/dd/yyyy | hh:mm:ss (PM or AM) | : | text string. |

The following code lines shows how the login process is put in the logger file:

```

1 using (StreamWriter logger = File.AppendText(@"C:\Users\luma\Desktop\...
    ProjectLogger.txt"))
2 {
3     logger.WriteLine( System.DateTime.Now + " : The user "+username+" ...
        logged in to the system.");
4 }

```

The database was implemented using LINQ to SQL component of .NET Framework that provides a run-time infrastructure for managing relational data as objects. LINQ to SQL maps the data model of a relational database to an object model expressed in the programming language of the developer. When the application runs, LINQ to SQL translates the language-integrated queries in the object model into SQL and sends them to the database for execution. When the database returns the results, LINQ to SQL translates them back to objects that you can work with in your own programming language.

Using LINQ to SQL will minimize the code size and will simplify dealing with database. And it becomes easier to write and read and understand written code because it becomes similar to dealing with C# objects. For example: to add a new node to the system, we declare an object (newNode) of the type (Node_tbl) which is the table name which is containing nodes in the database. Now we can fill the table information using the table columns names, after that we use the command *InsertOnSubmit()* for the insertion process. Then we use the command *SubmitChanges()* to save this change in the database.

```

1  Node_tbl newNode = new Node_tbl();
2  newNode.macAdd = MACadd.Text;
3  newNode.nodeID = NodeID.Text;
4  newNode.floor = floor.Text;
5  newNode.type = nodeTyp.SelectedItem.ToString();
6  newNode.used = used.Text;
7  dataFromDB.Node_tbls.InsertOnSubmit(newNode);
8  dataFromDB.SubmitChanges();

```

similar This code line is used to retrieve the patients name who is holding the mobile node:

```

1  PatientsList.DataSource = dataFromDB.Patient_tbls.Where(s => s....
    mnMAC == mnMAC).Select(x => x.name);

```

5.3 MATLAB code

MATLAB was used in various context in the localization process. We write a code that is used to collect RSSI data from the field in very fast and efficient way during offline phase. It is then used to classify and plot the collected data for a better understanding of how RSSI is distributed in the area. Note that there is no RF planning tools for ZigBee it was difficult to distribute FNs in a way that assure coverage from the 3 FNs in the whole area. But with our MATLAB code this process has become feasible. Another MATLAB code was written to train NN on the collected data to be used in the online phase. However, NN still need more time to be tested and optimized which might be left as future work since we spent most of this semester building the localization system rather than testing the suggested algorithms.

RSSI data collection code

In this code we will explain how we implement RSSI data collection in software. A small program was built to extract 6 RSSI readings from one update packet response. Those readings are stored in a logger files one for RSSI data and the other is used to save all packets that was involved in the process. The user is responsible for just running the part that read RSSI every time he want to take a measurement at specific place. The location coordinates was left zeros to be filled manually since buildings are different in their internal structure. The following code open logger files and prepare program to start accepting RSSI data. The file option 'a' is used to define the program permissions and means the new data will be appended at the end of the file.

```

1  %% --- Defined global variable and packets
2  x = 0;    y = 0;
3
4  update = sscanf(['7E 00 15 10 00 00 00 00 00 00' ...
5                  'FF FF FF FE 00 00 55 50 44 41 54 45 3F F2'], '%2X').';
6
7  RSSIdataFile = fopen('C:\Temp\RSSIdata.txt','a+');
8  packetLogger = fopen('C:\Temp\packetLogger.txt','a+');

```

The program is now ready to start collecting data. First of all we clear the buffer to get rid of any unwanted packet arrived from other nodes then an update packet will be sent and stored in the *packetLogger* file. The response will be read in pretty similar way to what is done in C# and MikroC. When the fingerprinting process is finished, the serial port should be closed as well as the logger files. The following code show how data are collected in brief.

```

1  %% --- Start Here ... !!
2  % Clean buffer before start collecting RSSI
3  if(s.BytesAvailable <= 0)
4      RxData = fread(s,s.BytesAvailable);
5      clear RxData;
6  end
7
8  % Store then send UPDATE packet ...
9  % Wait, read, store then discard MN msg ...
10
11 % read FN2 msg
12 while(~(s.BytesAvailable))
13     % just wait untill first FN2 packet F.2.M.x,P2M,P1M.P21,P3M.P31,P32
14     %         7E 00 16
15     %         90 00 13 A2 00 40 9A 50 48 FF FE 01
16     %         46 32 4D 31 1B 19 11 17 1E 1D BD
17
18 end
19 RxData = fread(s, 3)';    % read in decimal
20 length = 256*RxData(2) + RxData(3);
21 RxData = [RxData, fread(s, length+1)'];
22
23 if(RxData(16)==70)    % if it is from FN2
24     fprintf(packetLogger, ...
25         ['FNsMsg:\t\t%2X %2X %2X %2X %2X %2X %2X %2X %2X %2X ' ...
26         '%2X %2X %2X %2X %2X %2X %2X %2X %2X %2X ' ...
27         '%2X %2X %2X %2X %2X %2X \r\n'], ...
28         RxData);    % store FN packet
29     RSSI = -RxData([21 20 23 22 24 25]); % extract RSSI: RSSI(y,x) = P
30     fprintf(RSSIdataFile, '%6d %6d %6d %6d %6d %6d %6d %6d\r\n', ...
31         x,y,RSSI);
32 end

```

Plotting Heatmaps

After we have finished RSSI collection we need to visualize the data to understand it in a better way. The best way is to plot RSSI values against their coordinates using heat maps. In a heatmap, a data sample point is represented by (x,y,RSSI) where the coordinates are presented on the x and y-axes while the associated RSSI data are visualized using colors. Assuming data are available for a specific room or floor, this code is used to plot the heatmap of that array. Here we plot the RSSI data from FN1 in one room of building B in engineering department. A grid of 3x6 points was read and stored in *FN2wF1* variable.

```
1      %% --- Extract each FN from each direction
2      FN2wF1 = reshape(FN2west(:,3),6,3)';
3      FN2wF1 = FN2wF1(end:-1:1,end:-1:1);
4
5      colormap('hot');      % set colormap
6      imagesc(FN2wF1);      % draw image and scale colormap to values range
7      colorbar;              % show color scale
8      title('Power from FN1 in room 2 while MN was looking to the west')
```

Neural Network Training

NN training is the final step before the system becomes ready to be used in real-time. RSSI data had been collected and they are ready to be applied to the NN for training. Remember that RSSI measurements are no more directional (does not based on four directions since they were averaged) so that only one NN will be presented. Also note that localization will be performed at room levels in real time.

Two types of NN were of equal importance in this context. One for classification (NN as a classifier) and the other NN is for localization (NN as fitting tool). The first one was trained on the collected RSSI data to guess to what room do a specific RSSI data set belongs. This would be applicable to building B since its rooms are shaped by physical walls. If each room contains a FN then its RSSI will be dominant and enough to judge in that a MN is exist in that room. Once the room was determined, the next stage will be locating the MN inside that room. This is the job of the second NN. However, this one was not implemented since the directional sensor is not available.

Figure 5.9 shows NN final structure with one hidden-layer containing 6 neurons, 3 neurons at the input layer (represents RSSI from each FN) and one at the output (represents the room number: 1,2, or 3). Choosing room number representation was tricky. The clustering NN represent the output class (room) where the MN belong through a binary vector that have number of elements equal to the number of available classes (3 classes in our case represents three rooms). The room where the MN is in will have value 1 in its corresponding element in the output vector while other rooms will have 0's. However for fitting NN, the network is required to find a function that produces either number 1,2 or 3. Comparing both choices we find that the

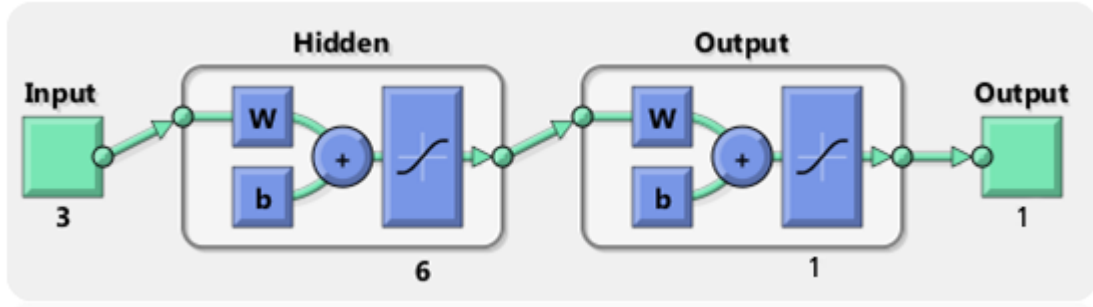


Figure 5.9: Feedforward-Neural Network structure.

performance of the clustering NN will not be dependent on what kind of output representation we are used while the fitting NN do. In the fitting NN if we choose to represent room numbers in different format other than values 1, 2, and 3 such as -1, 0, and 1 then the performance will be different! Despite this disadvantage, we find both networks fall within same accuracy for only three rooms by choosing room representation as (-1,0,1) to indicate room (1,2,3) respectively.

Reduced GUI using MATLAB

A reduced version of the main window was also designed using MATLAB. In fact, we did not intend to duplicate the whole C# code. We were interested in the localization process itself rather than the whole application as a monitoring system for hospitals. For that the database and its all connections was excluded from the MATLAB code, even though a new special database toolbox is available from Mathworks, and a text file was used to simulate the database functions.

MATLAB GUIs could be created using a special toolbox called GUIDE (Graphical User Interface Development Environment), this tool allows a programmer to layout the GUI by selecting and ordering the GUI components without the need of writing any line of code. Once the components are in place, the programmer can interactively edit their properties (such as name, color, size, font and text) and add codes that define the actions they perform. However, we prefer to build the GUI using pure MATLAB code. This approach is more flexible and supports advanced features.

The idea behind creating this GUI is to open the way for using the powerful functions that will not be available in other programming languages including C#. As we mentioned previously, RSSI data needs a lot of processing and filtering before the RSSI becomes directly related to the location of MN. Suppose that one had develop an algorithm using MATLAB, he will face problems in translating his work to C# in order to embed his work in the GUI. Fortunately, C# codes can be imported in MATLAB so that we can combine the main GUI built in C# with MATLAB and continue in this path.

5.4 Embedded C Code for Mobile and Fixed Nodes

Mobile and fixed nodes are nothing more than an XBee module with a PIC MCU. XBee modules support two modes of operation transparent mode (AT) and Application Programming Interface (API) mode. To create simple and fixed links, XBee works nicely in transparent mode. All what is needed is to configure the XBee module once with a destination address using XCTU software, then just bother yourself with the data you need to transmit and receive. This can be applied for mobile nodes; since all what they are required to do is to transmit their three sensor data as a broadcast message whenever they receive update message from the coordinator to identify their locations.

However, when the network consists of multiple nodes each needs to communicate with different destination in different ways based on different events such as our FNs, transparent mode couldn't be used and API mode becomes more convenient. The problem with the XBee is that transmitting to a different destination means you need to *reconfigure* the module to a different destination address before being able to transmit the data to the new destination. This is also apply when RSSI value is to be read.

Reconfiguration in the transparent mode means that the XBee module must enter the AT-command mode using a special sequence of characters (+++), wait for a relatively long *guard-time* (GT=1 second), then the intended command can be applied to configure the XBee with the new parameters (such as setting new destination address or trying to query RSSI value of the last received packet). After changing/reading any parameter, the XBee module must exit AT-command mode in order to be able to transmit data wirelessly to other motes in the network.

The *guard-time* is the required period of silence *before* and *after* the Command Sequence Characters (CSC=+++) of the AT command mode sequence (GT + CSC + GT). The period of silence is used to prevent inadvertent entrance into AT command, and can have a value from milliseconds to seconds. However, reducing GT to the milliseconds range is not preferable and can cause conflicts between command mode (which operates locally through UART port) and data transmission mode (which operates remotely through wireless link).

Since our project is a real time monitoring system, a 2s delay before an RSSI value from only one remote FN becomes ready is too long. The delay will become even worse when the coordinator node waits for more than six RSSI values from FNs. In fact, the large delay is not the only problem transparent mode suffers from. The GT delay causes additional serious problem since no remote XBee module is allowed to send any packet to the FN during this period otherwise the reading RSSI process will fail and FN stop responding. This constrain can't be guaranteed as we have a large active mesh network and nothing prevents other nodes from sending packets to the FN while it reading RSSI value of a certain packet.

Table 5.1: Types of used ZigBee packets.

| Packet Type | Hex API ID | Description |
|---------------------|------------|--|
| AT Command | 0x08 | Used to query or set module parameters on the local device. |
| AT Command Response | 0x88 | Ack for AT command or value of queried parameter. |
| Transmit Request | 0x10 | Causes the module to send data as an RF packet to the specified destination. |
| Transmit Status | 0x8B | Ack to indicate Tx Request status. |
| Receive Packet | 0x90 | When the module receives an RF data packet, it is sent out the UART using this message type. |
| Modem Status | 0x8A | Is sent in response to specific conditions, such as power on. |

In order to overcome these problems we chose to use API mode. Data and commands in API mode travel exactly in the same way but in different frame types as shown in table 5.1. The meaning of each byte in each frame is explained in detailed in appendix A. Note that at the receiver side, it is possible to extract the source address of any received packet whether it was carrying RSSI value from the local XBee (with packet type 0x88) or it was carrying data from a remote module (0x90). This is useful feature since we have a large network with many mobile and fixed nodes. They are continuously sending data and management packets through the network, and it is important to distinguish packets from each others, which can be easily done in API mode.

MikroC PRO software was used to program the second main component in the mobile and fixed nodes which is PIC MCU. MikroC is a powerful, rich-environment tool that uses C language to provide the programmer with the easiest solution for developing embedded systems without affecting the performance. That what distinguishes it from other C compilers used to program PIC microcontrollers. From MikroC we have used at least four libraries to complete this project. They are: *ADC library* to read MNs analog sensors, *UART library* to communicate with XBee through serial port on both mobile and fixed nodes, *C_string library* to check the incoming packets type (in the mobile node which runs the transparent mode), *Conversions library* to convert between ASCII, Hex and Decimal formats, and many other libraries.

Before using the PIC, it should be initialized by defining pins type (whether they analog or digital) pins function (input or output) and data rate of the UART port. For the MN, the

following code define 3 analog inputs to read the attached 3-axis sensor, and the data rate is set to 19200 bps. FNs have the same initialization except no digital input is needed.

```
1  /*
2      This document is for the Mobile Node configuration
3  */
4  void main(){
5      // PIC initialization:
6      ANSEL = 0x07;          // 3 analog pins (x,y,z)
7      CMCON0 = 0x07;         // Disable comparators
8      TRISA = 0x07;          // the 3 analog pins are inputs at port A
9
10     UART1_Init(19200);      // Initialize UART module at 19200 bps
11     Delay_ms(100);          // Wait for UART module to stabilize
12
13     while(1){
14         // The rest part of the program is located here ...
15     }
16 }
```

MikroC PRO UART Library provides comfortable work with the Asynchronous (full duplex) mode given that the PIC MCU have a hardware integrated UART (such as PIC16688 that we used). The library implements two functions to read from UART port `UART1_Read` & `UART1_Read_Text`, and similar two functions to write to the UART port. The first read function reads the incoming data byte by byte while the second one is more powerful and can read a text up to 255 character or until a delimiter is faced. Since MNs use transparent mode (which deals with data as text), it can easily communicate with the PIC using Text read and write functions. However, FNs run API mode which read and write data using frames of hex numbers. Unfortunately, Text read and write functions show a lot of problems with hex data so we return to `UART1_Read` and `UART1_Write` functions.

There is a real challenge with reading incoming packets byte by byte. Assume that a packet of certain length arrives at the XBee module, it will be stored in the XBee RF received buffer. The whole packet is then moved to the UART transmit buffer inside the XBee waiting for PIC to read it. If we use UART read function in a loop that is repeated many times (equal to the packet length) then the result will be unpredictable. This is because the PIC MCU reading speed is much higher than UART data rate. So PIC will read the first byte correctly and perform many read instructions while the next byte does not arrive from the UART port.

If a certain amount of delay is inserted between each two successive readings then we might lose some bytes if the delay was not precise. This is because the PIC has a buffer size equal to only one byte. Another problem will occur if more than one packet arrives at the XBee in very short interval. Note that the XBee will have no problem with storing these packets in its large

buffer until the PIC becomes ready. The problem is at the PIC side since we don't have any indication about the end of the packet by observing its content because the last byte is always the checksum which is variable. This advantage was available in the text read function and suitable with XBee modules that runs the transparent mode. For that we are required to read the first three bytes and wait to calculate the packet size, then read the rest part of the packet. The following code shows how we implement the reading function.

```

1 void Read_Rx_Packet(void){
2     unsigned int length=0;           // packet length
3     char i=0, j=0;
4
5     while(j==0){
6         while(UART1_Data_Ready()==0); // wait for 1st byte
7         RxData[j] = UART1_Read();      // read delimiter (0x7E)
8         if(RxData[j]==0x7E){j++;}      // move on if it is valid delimiter
9     }
10
11     while(UART1_Data_Ready()==0);      // wait for 2nd byte
12     RxData[j] = UART1_Read();          // read length's higher-byte
13     j++;
14
15     while(UART1_Data_Ready()==0);      // wait for 3rd byte
16     RxData[j] = UART1_Read();          // read length's lower-byte
17     j++;
18
19     length = RxData[1]*256 + RxData[2]; // calculate packet length
20
21     for(i=3;i<length+3+1;i++){ // wait for rest of data bytes
22         while(UART1_Data_Ready()==0);
23         RxData[i] = UART1_Read();
24     }
25 }

```

Moving from the UART library to the ADC library to explore the MN code that is used to read the ADXL sensor. Remember that the sensor was not useful in this project since the output voltage does not reflect the change in MN direction. However, we chose to keep this code to be used in the future work with either a better sensor such as HMC6343 (figure 4.4b), or even including any other three sensors to monitor patients health state. In either cases all what is needed is to connect any three sensors to the PIC at pins AN0-AN2 and use them immediately. Note that the MN reads these sensors whenever it receives an update packet from the coordinator. The data is sent back to the coordinator as a string of four characters (since the ADC is 10 bit resolution which produces a max voltage of 1024 mV). The following piece of code shows how to read AN0 analog channel. The other two channels can be read in the same way.

```

1 // Global variable declaration
2 char sensors[30] = "MN1_x:0000 y:0000 z:0000?";
3
4 void main(){
5     // PIC initialization is here ...
6     while(1){
7         // read 1st sensors (x) at AN0 pin:
8         ADCx = ADC_Read(0);
9
10        // prepare the data packet
11        sensors[6] = ADCx/1000 + 48;           // Add 48 to get the
12        sensors[7] = (ADCx/100)%10 + 48;       // ASCII character
13        sensors[8] = (ADCx/10)%10 + 48;        // value of each
14        sensors[9] = ADCx%10 + 48;            // digit.
15        Delay_ms(5);
16
17        // read other sensors ...
18
19        // Send Sensor Reading
20        UART1_Write_Text(sensors);
21    }
22 }

```

After a MN has received update packet from the coordinator, it will send its response as a broadcast message. The neighboring FNs will capture this packet and read RSSI value through software. The following code illustrates how FN1 will responds to the MN packet. Firstly it checks the type of the packet to make sure that the received packet was from a valid mobile node. In order to identify the packet source, it is preferable to check the MAC address of the source rather than checking the data itself. This is because the MAC is fixed and could not be changed. However, anyone can send similar data format to what you expect from the MN. In our case, the MN is assumed to send a packet with this format: "MN1_x:0000 y:0000 z:0000?". In fact, we try to avoid using MAC address to distinguish packets since if one XBee module was damaged for any reasons, we do not need to reprogram its associated PIC to include the MAC address of the new module.

After sending the ATDB command as an AT command packet of type 0x08, the response will come back in a packet of type 0x88 with the 9th byte carrying the value of RSSI. The received value is in hex and the corresponding RSSI can be calculated using this conversion:

$$RSSI[dBm] = -Hex2Dec(Byte9_{packet0x88}) \quad (5.1)$$

The RSSI value is then stored in the *powers* packet as well as the MN id that send the sensor reading packet. The first byte in this packet is always 0x7E and used to indicate the start of a new packet. The next two packets represents the number of bytes the data will

span. Since this packet is to be sent as an RF data, the packet is sent with type 0x10. FN2 will receive it as a packet type 0x90. The 0x01 byte is an arbitrary sequence number that is used to match this packet with its acknowledgment. The next 8 bytes are the 64-bit MAC address of the destination (which is FN2 in this case). The next four bytes are reserved. The actual data starts from byte 17 (given that indexing starts from zero). Byte 0x46 in hex represent the ASCII letter F to indicate Fixed node, it is followed by FN ID (which is 1 in this case since the source is FN1). The 0x4D represent letter M and stands for Mobile node followed by its ID. The next byte is to store RSSI value measured at F1 from Mx. After the packet is ready, the check sum is calculated and the packet is sent to FN2. Remember that according to the packets flow that we describe in table 4.2, FN2 collect all RSSI values from its neighbor 2 FNs before sending the RSSI packet to coordinator as a one unit of 6 RSSI values.

```

1 char powers[] = { // This is unicast packet to FN2, F.1.M.x.Pm
2                   0x7E, 0x00, 0x13,
3                   0x10, 0x01, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x9A, 0x50, 0x48,
4                   0xFF, 0xFE, 0x00, 0x00, //CRC here = 35
5                   0x46, 0x31, 0x4D, 0x00, 0x00, // F1Mx(PF1M)
6                   0x2F};
7
8 void Read_Rx_Packet(void);
9 void main(){
10    // PIC initialization is here ...
11    while(1){
12        Read_Rx_Packet();
13        if(RxData[15]==0x4D){ // if from MN
14            MNid = RxData[17]; // read MN id (if more than one exist)
15
16            // Send ATDB packet(0x08 type) and Read RSSI (0x88)
17            for(i=0;i<sizeof(ATDB);i++){UART1_Write(ATDB[i]);}
18            Read_Rx_Packet();
19            if(RxData[3]==0x88){
20                powers[20] = MNid;
21                powers[21] = RxData[8];
22            }
23            else{powers[21] = 0;}
24
25            // calculate CRC for powers packet and send it to FN2 ...
26        }
27    }
28 }

```

Note that we built a virtual cluster tree topology inside the ZigBee mesh network. The basic unit of this cluster is 3 FNs with one of them acts as a data sink that deliver RSSI data to the coordinator. Note that in order to cover a wider area, this cluster could be repeated as many times as you want. The only limitation on this topology is that if the sink node fails to receive

the packet from one of its cluster members, the whole 6 RSSI values will not be available at the coordinator even if those cluster members can read MN power, and can deliver it to the coordinator. However, this does not create a problem since two RSSI values are useless without the 3rd value since at minimum we need 3 RSSI values from 3 different FNs to be able to predict MN location. So if we make sure that all 3 FNs cover the cluster area, which is always the case, we will not have any problem.

FNs two and three are programmed in a pretty similar way to what we do with FN1. The complete code is available with the associated CD. We just want to add the packet format used to sent FN2 and FN3 data. The meaning of each byte can be extracted in the same way as in FN1. For more information refer to appendix A -ZigBee section- that describe packet types and its format in more details.

```

1      // Powers packet of FN3 = F3Mx(PF3M)(PF31)
2      char powers[] = { // accept ack, BH=0(max), unicast to FN2.
3          0x7E, 0x00, 0x14,
4          0x10, 0x01, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x9A, 0x50, 0x48,
5          0xFF, 0xFE, 0x00, 0x00,          //CRC here = 35
6          0x46, 0x33, 0x4D, 0x00, 0x00, 0x00, // Data is here
7          0x2F};
8
9      // Powers packet of FN2 = F2Mx(PF2M)(PF1M)(PF21P)(PF3M)(PF13)(PF23)
10     char powers[] = { // accept ack, BH=0(max), unicast, F1MxPP
11         0x7E, 0x00, 0x18,
12         0x10, 0x01, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x91, 0x9B, 0x28,
13         0xFF, 0xFE, 0x00, 0x00,          //CRC here = 57
14         0x46, 0x32, 0x4D, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // Data
15         0x2F};

```

5.5 The Central Localization Engin

This part of the monitoring window is represented by the Monitoring Options area in the GUI. The first option here is the update Locations option. This option is executed when the user clicks the corresponding button. To update a location, a serial port connection is needed to interact with the ZigBee nodes network. We used some variables to accomplish the serial port communication as shown below:

```

1      //***** variables for serial port *****
2      SerialPort MyPort;
3      int[] RxData = new int[50];
4      int j = 0;
5      int length = 0;
6      int[] RSSI = new int[6]; // MNx power [F1M F2M F3M F12 F13 F23]

```

```

7         int[] pwrREF = {-47, -48, -51}; // F12 F13 F23
8         bool readyToPlot = false;
9
10        // *****

```

The port to be opened is *MyPort*, the *RxData* is the array to store the packets which are coming from the ZigBee network. The variables *j* and *length* are used while reading packets to make the reading process done properly. The 'RSSI' array is used to store the power values which the packet contains. The first byte contains powers of MN measured at FN1, while the second byte represent the RSSI of MN measure at FN2. The third byte if for FN3. On the other hand, the last three bytes in RSSI array are powers between FN1 and FN2, powers between FN1 and 3, and the power between FN2 and FN3 respectively. Those three values are used to reduce the power variation as described before by comparing them with the reference power level *pwrREF*. The *readyToPlot* variable is used to give a sign that the data is ready to be sent to neural network function for getting the location to be shown on the map.

The updating process is used in a try clause, so if anything wrong happened, the catch clause will stop the operation by showing a message box. The first step in the updating process is to prepare the update message packet, then to clear the buffer in case that has some un-needed data before sending the update message to the serial port for the connection with master node. The following lines describe this process:

```

1    // preparing the update message packet:
2    byte[] update = new byte[] { 0x7E, 0x00, 0x15, 0x10, 0x00, 0x00, 0...
        x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0x00,...
        0x55, 0x50, 0x44, 0x41, 0x54, 0x45, 0x3F, 0xF2 };
3
4    // Clear the Buffer:
5    if (MyPort.BytesToRead > 0)
6    {
7        string temp = MyPort.ReadExisting();
8    }
9
10   // send update packet:
11   MyPort.Write(update, 0, update.Length);

```

After sending the update packet, we need to wait the response packets. We expect two kinds of packets to respond; the first one is the mobile node which we dont need any data from it. Actually, it is used just to get the mobile node address for the system software which needs these addresses to indicate whom patient is holding it. The second packet we expect is the fixed node (FN2) packet; when the fixed node packet arrived, the RSSI array will be populated with

the power values. When we have the power values, the *readyToPlot* variable will set true to indicate that we can send this data to be processed to get the location coordinates. After that the *readyToPlot* variable will be set *false* to prepare the next updating process. If something wrong happened in the previous process, the catch clause will show a message to indicate that there is something wrong, maybe the port is not opened. As the following code shows:

```
1      catch
2      {
3          if (TrackPat.Visible == false)
4              StopTracking_Click(sender, e);
5          MessageBox.Show("something wrong happened, check the serial port ...
                           connection!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
6      }
```

Moving to the next option in the L&TE which is implemented in the *Track Patients* function. When the user chose this option, all other options in the software will be disabled and a *Stop Tracking* button will appear. We disable everything since most resources will be used by the serial port. And to tell the doctor that there is extensive amount of packets is being transmitted to the patient (which will drain his battery's life). So we assume that the tracking process will be done only for a short time interval. The *TrackPat_Click* function will invoke the function which used to update locations using a timer event handler happened every two seconds. The system will continue this process until the stop tracking button is clicked to return the system to its normal case. The following two functions describe this process:

```
1      private void TrackPat_Click(object sender, EventArgs e)
2      {
3          // Disable all components other than what used in tracking
4          updateLocs.Enabled = false;
5          TrackPat.Visible = false;
6          StopTracking.Visible = true;
7          openPatProfile.Enabled = false;
8          ChkPort.Enabled = false;
9          OpnPort.Enabled = false;
10         ClosePort.Enabled = false;
11         menuStrip1.Enabled = false;
12
13         // create timer to perform tracking as multiple updates
14         track = new Timer();
15         track.Interval = 2000; // 2 seconds
16         track.Start();
17         track.Tick += new EventHandler(track_Tick);
18     }
```


Chapter 6

Testing and System's Performance

In the previous chapters, we explain the idea behind the project, and discussed the conceptual design as well as the implementation of both hardware and software parts. In chapter 1 we introduce the complete idea of the monitoring system even if we know that the idea was bigger than a one year graduation project. The plan was to use a system that provide RSSI data directly from ZigBee transceivers and develop new algorithm based on these readings. However, we face many problems with that and we fail to buy such systems. So we have no choice rather than building our own system from scratch together with its GUI. Start from the zero point was difficult and consumes a lot of time so that the suggested localization algorithms do not have enough time to be tested and optimized. For that this project will be considered as a first stage in building a robust localization engine that is able to operate in real-life environment and achieve a good localization accuracy.

This chapter is dedicated to test the performance of the system. The performance will be tested through three parts: the first is to test the monitoring system that was designed for hospitals. The second is to test the system as a fingerprinting device in terms of number of packet lost and required time to collect and store the data. The final part is to test the suggested algorithm to reduce RSSI variation from the environment to enhance the localization accuracy.

6.1 Test The main Software

To test the system, a small scenario has been chosen such that all classes were involved. This starts from opening the software, then adding a new patient and a new node, opening the serial port, checking the patients List then logout from the system.

At this time we have 2 users for this software (System and Doctor1), if another username is used to enter the software, an error message will appear. Figure 6.1 shows the login window with wrong username. When a proper username and password are used, the main window will appear with a map set on its place as the figure 6.2

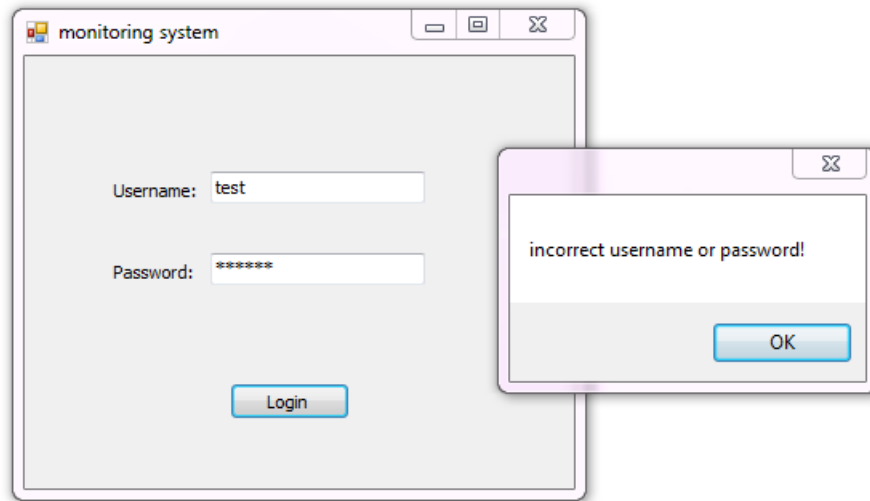


Figure 6.1: A wrong username was entered in the Log in screen.

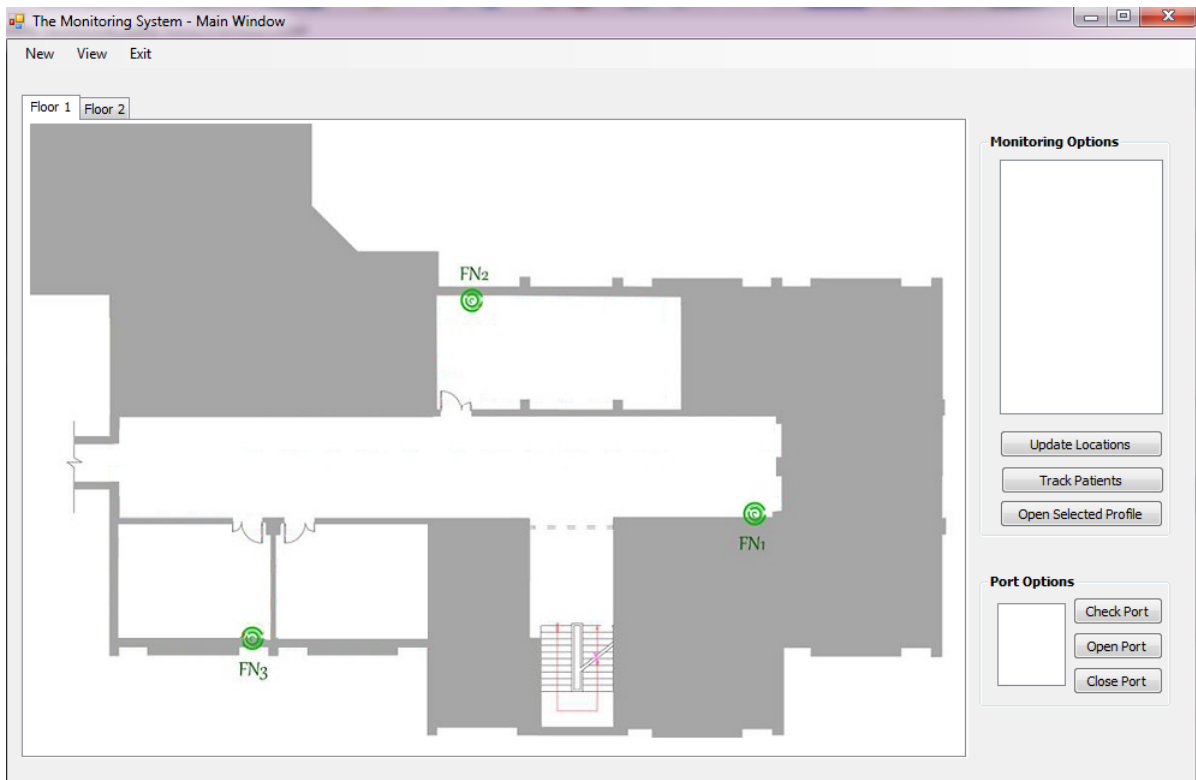


Figure 6.2: The main monitoring window with the map of 6th floor of B-Building.

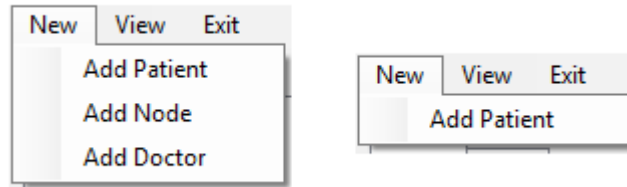


Figure 6.3: Strip menu options.

When the user (System) login to the system, he can see all menu options while when the other user login he can see less options as the shown in figure 6.3. when the user choose to add a new patient, the username who is adding and the system date will put automatically on the corresponding fields. If wrong Card ID is written the following error icon will appear and a small message will tell the error (ID card must be 9 digits). The patient's age will be filled automatically when the user changes the calendar date (figure 6.4). After adding the patient, an information message will be appeared and the form will be closed.

Figure 6.4: Adding New patient. System will check the data for validity.

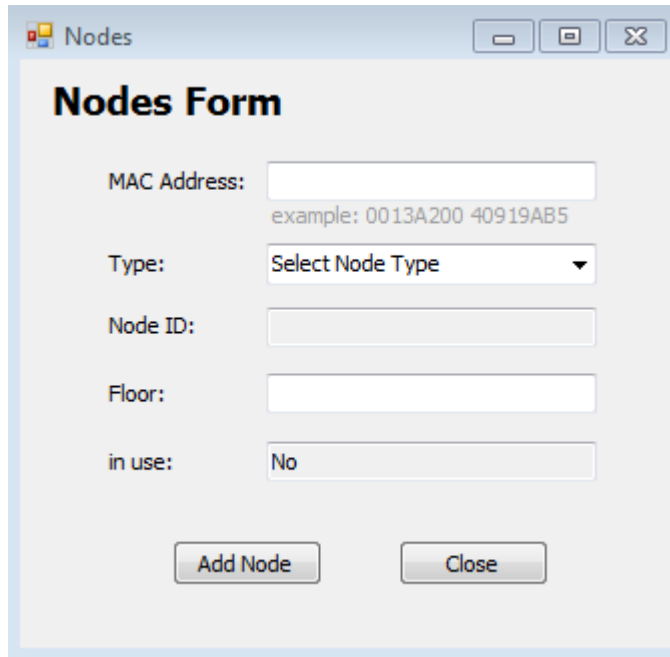
A Windows-style dialog box titled "Nodes" with a standard icon in the title bar. The main content area is titled "Nodes Form" in bold. It contains five input fields: "MAC Address:" with a text box and a hint "example: 0013A200 40919AB5" below it; "Type:" with a dropdown menu showing "Select Node Type"; "Node ID:" with a text box; "Floor:" with a text box; and "in use:" with a text box containing "No". At the bottom are two buttons: "Add Node" and "Close".

Figure 6.5: Nodes Form.

Only the user (System) can add a new node; he can choose the *Add Node* option in the *New* option in the menu strip. The user will not be able to add the node until the node type is chosen. When the user chooses the node type, and the node ID will be put automatically in its field (see figure 6.5). When the user press the *Add Node* button, an information message will appear then the form will be closed automatically.

When the user chooses to update locations while the master node is not connected to the laptop, the following error message will be appeared (figure 6.6).

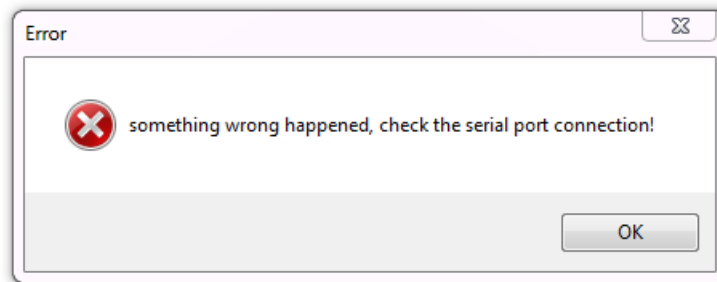


Figure 6.6: Attempts to use serial port while coordinator is not connected.

After the master node is connected to the laptop, the port can be opened when the user click on Open Port button, then an update message should be sent in order to get the mobile node address, so the patient who holding it will be retrieved from the database. When the user click on the close program option, a message will appear that a log out process will happen and the login window will be opened again. And the actions will be stored in the logger file as shown in figure 6.7.

```
5/20/2013 2:07:15 PM The user Doctor1 has logged in to the system.  
5/20/2013 2:09:51 PM The user Doctor1 has added a new patient.  
5/20/2013 2:12:18 PM The user Doctor1 has added a new Node.  
5/20/2013 2:18:13 PM The user Doctor1 logged out from the system.
```

Figure 6.7: Actions performed during the session are stored in a logger file.

A basic test has been done to make sure that the system can determine the location in a room level and draw a static shape in that room to indicate the location place together with an information message. This test was using the RSSI power values without any optimization, with an environment close to the ideal state when we collect the power values. Figure 6.8 shows one of the results.

6.2 Fingerprinting And RSSI Visualization

In this section, the collected RSSI readings will be plotted and compared. We have two main RSSI data sets: one for the Mosque and the other for Building B of the Engineering Department. We will comment on the LOS effect on the RSSI and state a conclusion regarding FNs distribution to achieve better RSSI fingerprint.

Fingerprinting in the Mosque

The process of fingerprinting was described in section 4.3. We collect about 7000 RSSI value using our system based on the new suggested algorithm that is proposed to reduce variation in RSSI. The time required to complete this mission was only 6 hours with no single packet was lost. This reliable and quick system suggest a very good solution that will overcome the main limitation for fingerprinting techniques (Collecting data is known to be very tedious task).

In figure 6.9 the direction of movement was to the west (from FN2 to FN1). Three fixed nodes was distributed as described before. Note that the RSSI readings are approximately in the same range regardless the location in the mosque. As we expected, the LOS effect was so bad and multipath effects destroy any meaningful information about the location. If we examine the first

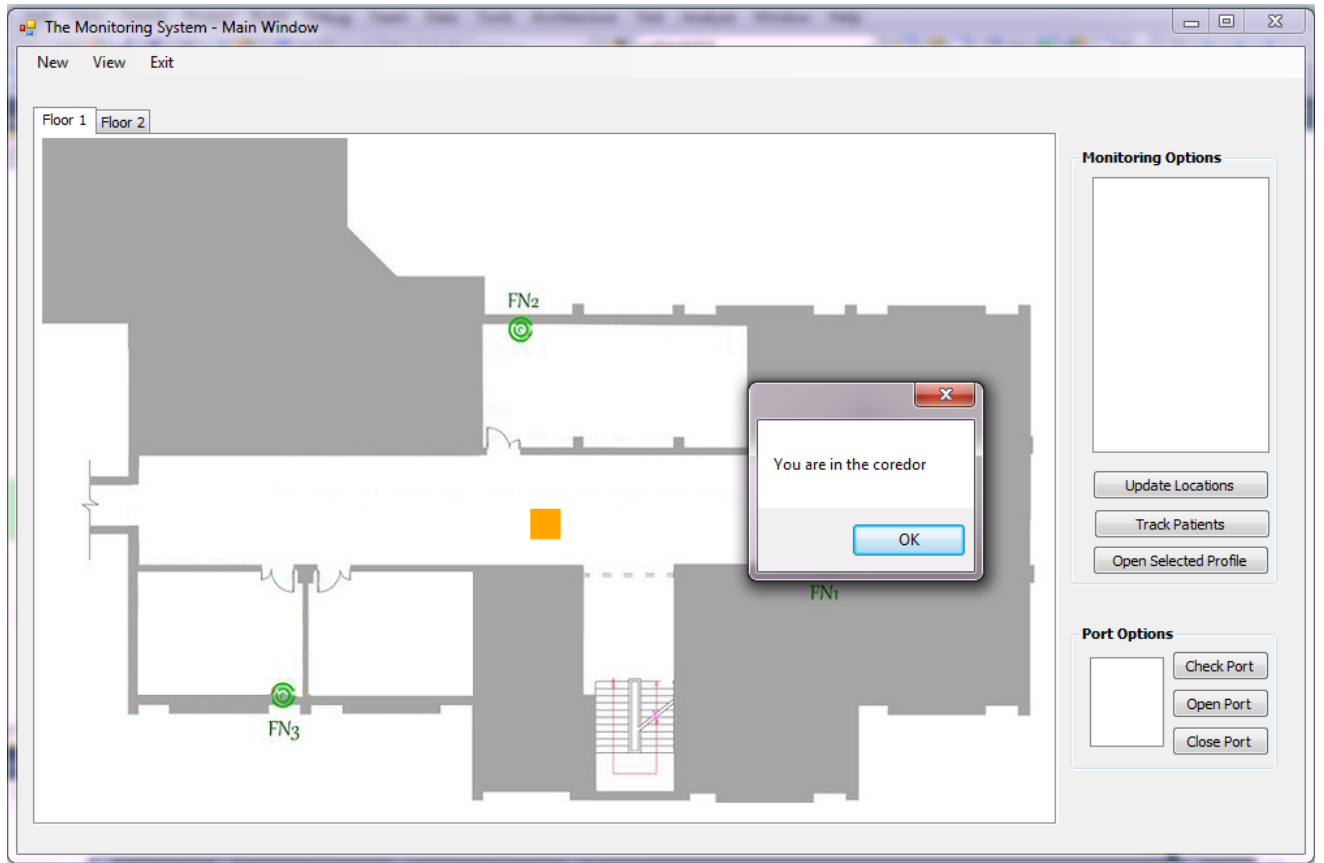


Figure 6.8: Localization scenario to test drawing functionality. No obstacles as presented, RSSI variation reduction algorithm was not implemented.

subfigure, it was clear that the strongest power level is when we are exactly beside FN1. This is also clear in the 3th subfigure when we are standing exactly beside FN3. The total effect of three readings is plotted to see how do NN see the mapping between power and locations. Note that there are differences between adjacent locations, but with some similarities from around the mosque. These similarities could be reduced by increase the number of FNs. Also a feedback from can be added to the FF so that the current location will not only depends on RSSI (which might be similar in two or more different location) but also depends on the previous location. This will filter out any other possibilities except the one the MN is being in.

The second subplot that visualize the RSSI from FN2 is a little bit different since the power of FN2 is not dominant exactly near FN2. This is because we were moving while leaving this FN behind us all the time. Unlike FN1 and FN3, FN2 do not have a clear LOS link to the MN since patient's body represent a constant attenuation of about 10dB. This is explain the importance of the suggested sensor that was assumed to determine the movement direction of the MN. Once the direction is known, the directional NN could be used with better accuracy

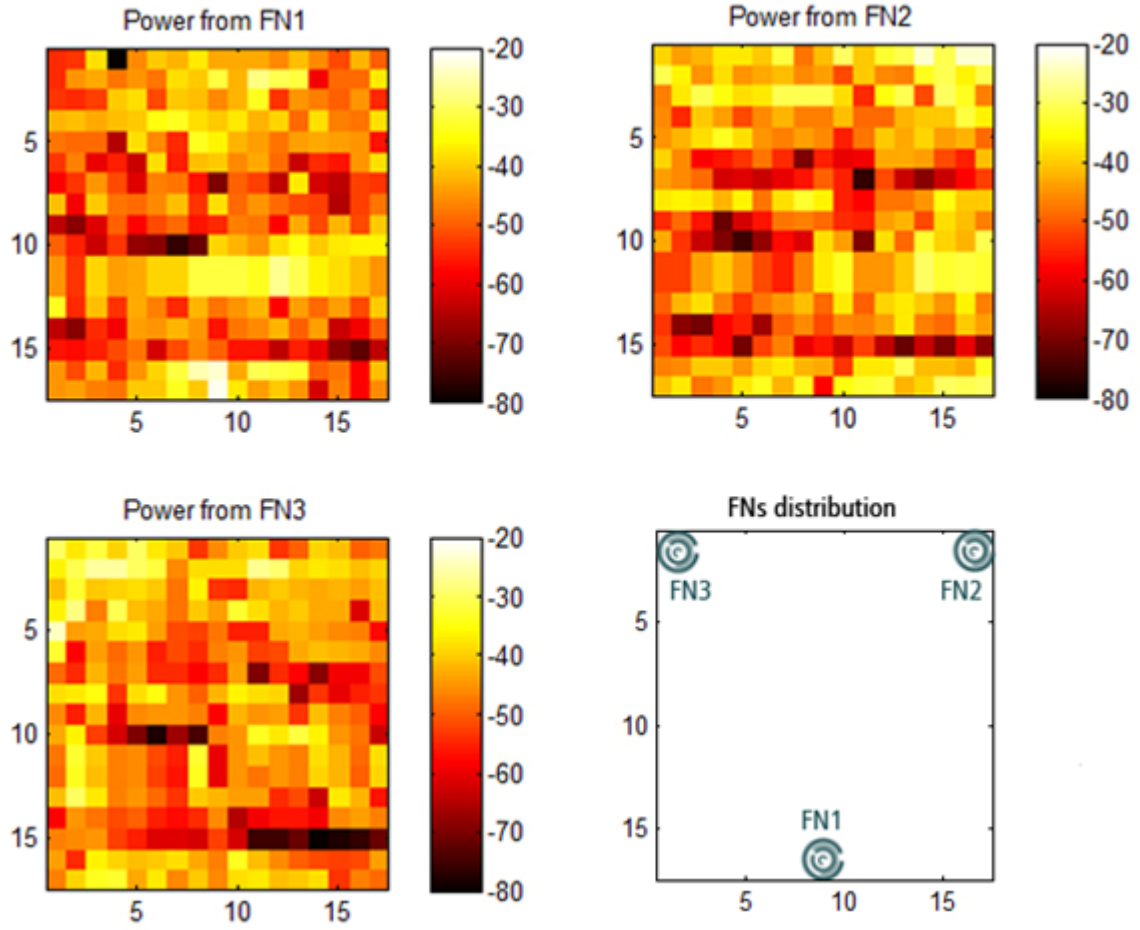


Figure 6.9: RSSI fingerprint at the Mosque while moving in the direction from FN2 side to FN1.

than using one NN regardless of the movement direction.

One final note regarding the mosque scenario is the time needed to complete one update command. Refereeing to section 4.3 again: the time required using the 3rd scenario to complete the update command was only 900ms including: program execution, packets travel time, reading the 3 sensors and plotting the location on the map using a basic localization equation that simulate the optimized neural network (until the last one becomes we tested). Figure 6.10 shows the required time taken from C# system. It is important to note that this time was for the 1st update. We note that this delay was max since XBee needs to make a route discovery the first time it sends a packet to an unknown destination. However, this delay was acceptable and good enough for the tracking to be done smoothly. This time is less that half of the update rate used in the tracking process (which is 2 second) and this is very good.

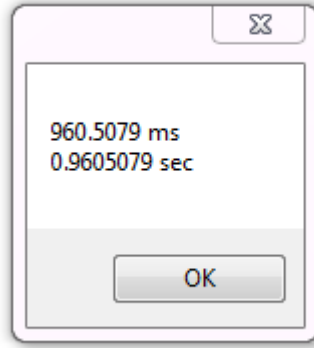


Figure 6.10: Required time to complete 1 location update.

Fingerprinting in the B-Building of Engineering Department

Figure 6.11 shows the RSSI values in the corridor from all FNs, each color point reflect a circle in the map where the power was measured. The FNs locations are also shown. Note that RSSI is distributed in a way closer to the ideal case than we see in the mosque. Subplot 1 is from FN1 and the strongest signals were there. The figure in the middle is from FN2, the strongest signals are in the middle of the corridor since the door of the room at which FN2 is located is there. Hence a LOS can be found there.

In figure 6.12 we present another sample of RSSI from the three FNs. The left three subplots were plotted using a unified scale to see the effect of walls on RSSI of each FN in this room. It is clear that every FN has a range of signal strength that is distinguished from the others. For example, the mean value of RSSI from FN1, FN2, and FN3 in this room is -49.7 dBm, -27.4 dBm, and -77.9 dBm respectively. As expected, FN2 has the strongest signal since it is the only FN that has direct LOS link to the MN in this room. This nice relationship was missed in the Mosque since it has no physical walls to avoid LOS.

In the three subplots that are shown on the right, the same data set was plotted but with the color scale's ranges taken from the data itself (range is defined by the max and min of the each data set). Note that even inside the same room, there is differences between powers with some similarities. Of course, any two locations with similar RSSI values will not be distinguishable from each other so an error might occur. From the collected data we calculate the variance of RSSI in the room from FN1, FN2, and FN3 and the result was 18.2, -34.7, and 56.5 respectively. We notice higher variation from FN3 since its signal is the weakest one, and the weaker the signal the more sensitive it will be and the more variation we will have which lead to a larger variance we will got.

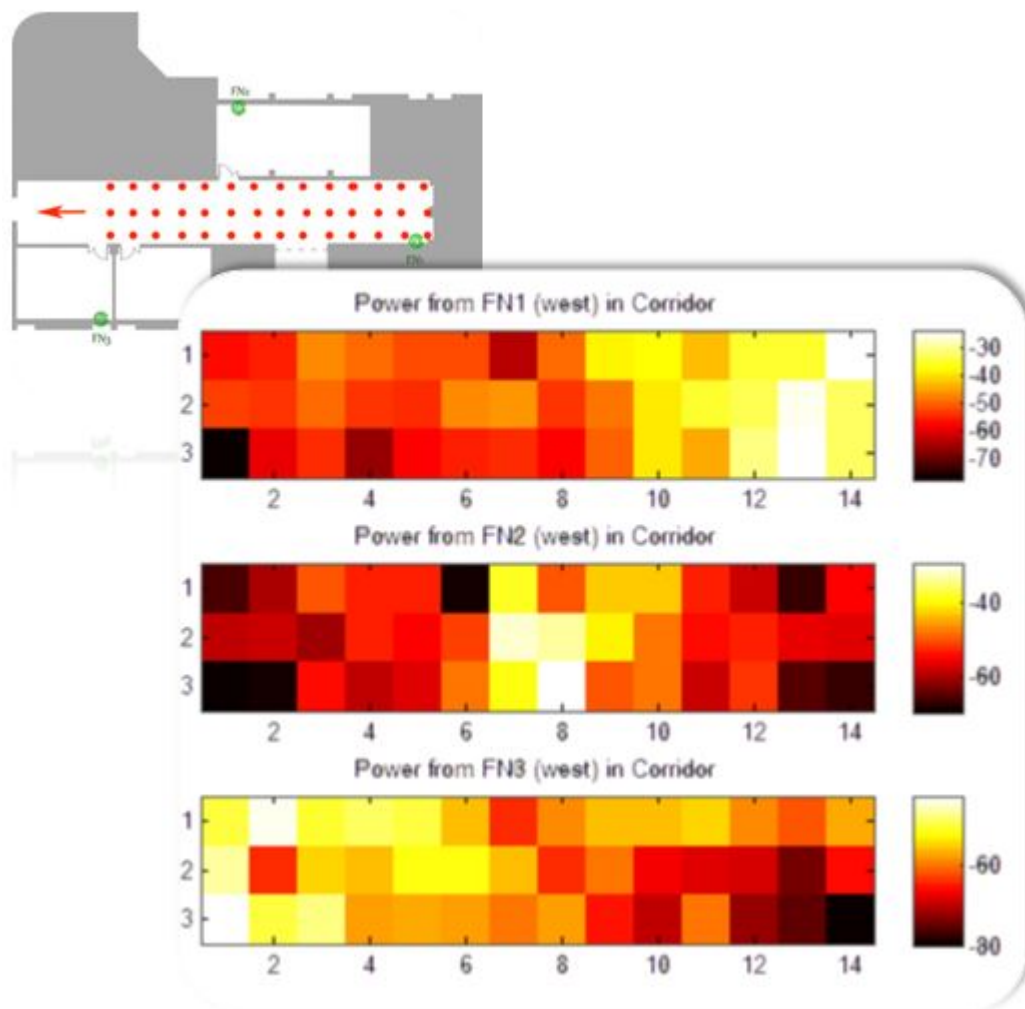


Figure 6.11: The heatmap in the corridor from.

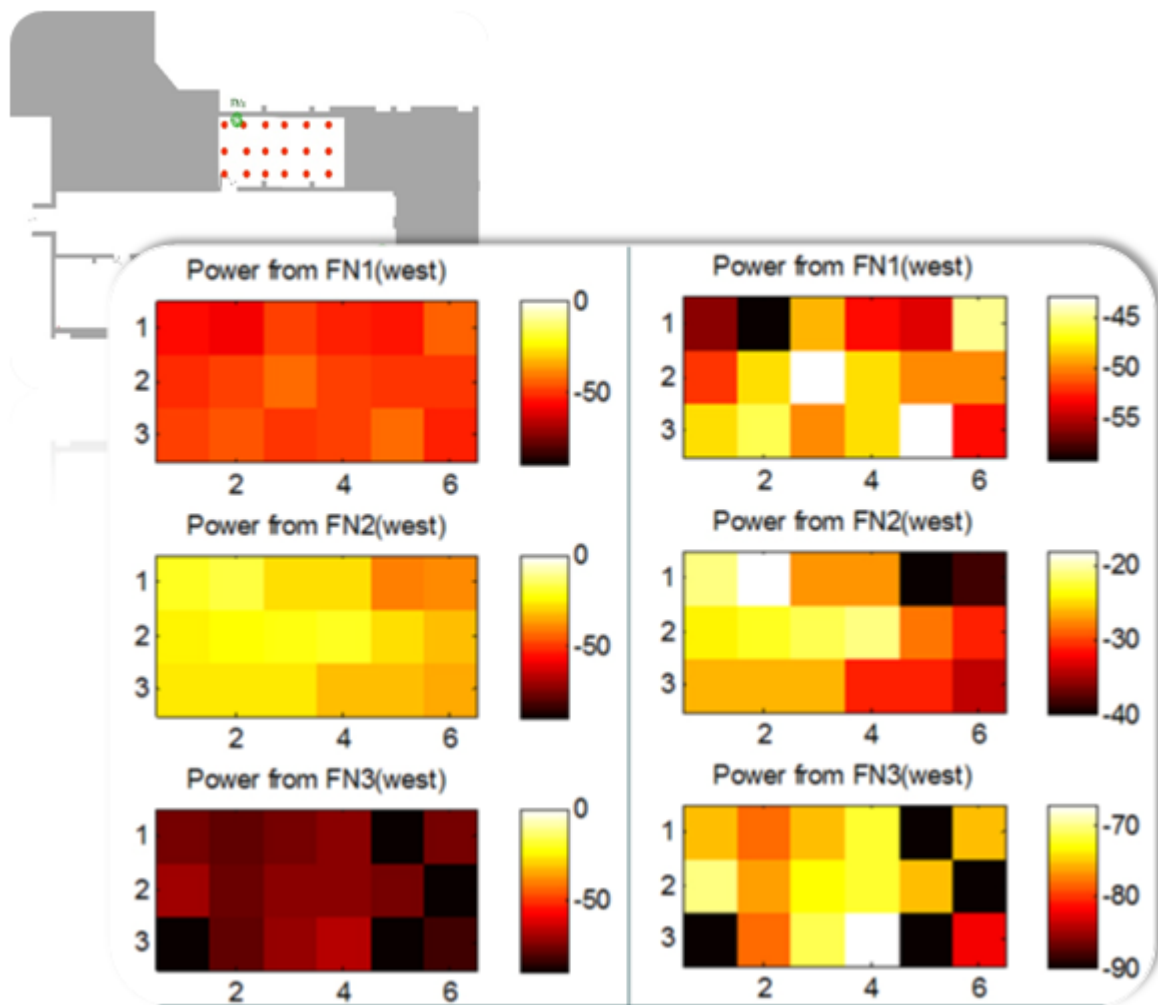


Figure 6.12: The heatmap room where FN2 is located.

6.3 Neural Network Performance

After RSSI data had been collected, they were applied to the NN for training. In this section we will analyze the training curves which will suggest how good NN will be in localization. Remember that RSSI measurements are no more directional so that only one NN will be presented. Also note that localization will be performed at room levels in real time.

Applying Levenberg-Marquardt algorithm to train the network leads us to the following model:

$$output = \tanh(\mathbf{W}^{(2)}).\tanh(\mathbf{W}^{(1)}.P_{in} + b_1) + b_2 \quad (6.1)$$

where;

- P_{in} : is the input power vector from FNs.
- b_1, b_2 : are biases for hidden-layer and output-layer respectively.
- W^1, W^2 : are the weight matrices for hidden-layer and output-layer respectively.

and they have these numeric values:

$$W^{(1)} = \begin{pmatrix} -13.8121 & 18.7784 & -4.5844 \\ 0.0410 & -0.1062 & 0.0590 \\ -0.7069 & -0.1610 & -0.3888 \\ 0.0364 & -0.0647 & 0.0353 \\ -0.1297 & -2.5897 & 0.9628 \\ -0.0551 & -0.2351 & 0.1998 \end{pmatrix}$$

$$W^{(2)} = [-0.0760 \quad 1.2626 \quad -0.0054 \quad -1.099 \quad -0.0644 \quad 0.5733]$$

$$b_1 = \begin{bmatrix} 47.8897 \\ -2.3013 \\ -62.3011 \\ -1.1905 \\ -14.3073 \\ 2.7044 \end{bmatrix} \quad b_2 = [1.7371]$$

Figure 6.13 describe the performance of NN during the training process in the offline phase. Note that at the beginning of the training the error was very high since the weights are randomly chosen and training was not started. By applying training data the error starts to drop down as indicated by the blue curve. A proper division of the data set can be verified from the curve since the validation and test curves are close to each other at the end of the training process. No overfitting problem occur as we can see since the validation curve does not increase while training precedes. However, the network reach its best performance after epochs 29 and no further enhancements have been achieved for 6 epochs so the training was stop at that point.

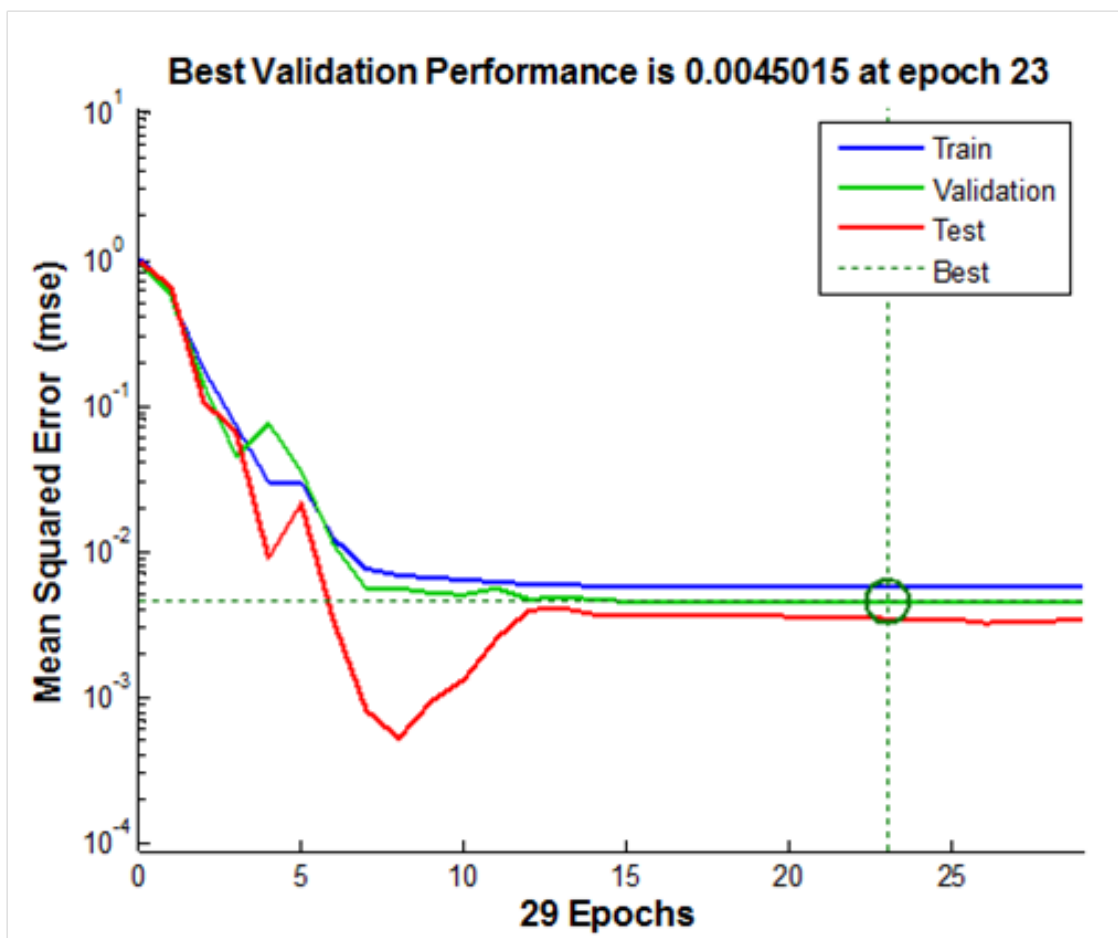


Figure 6.13: NN performance for all three sets during training.

Finally, the correlation between the NN output and the target is given by figure 6.14. Ideally, both should be identical to have correlation value of $R = 1$ which is represented by the dash-line. We find that NN achieved fair prediction capability with a correlation factor of 0.97. Note that there are some ambiguity of certain level about distinguishing the room 2 from the corridor, and this was clear in figure 6.11. The power from FN2 dominates the power of FN1 at some locations, so that NN judge that the MN is inside room 2 instead of the corridor.

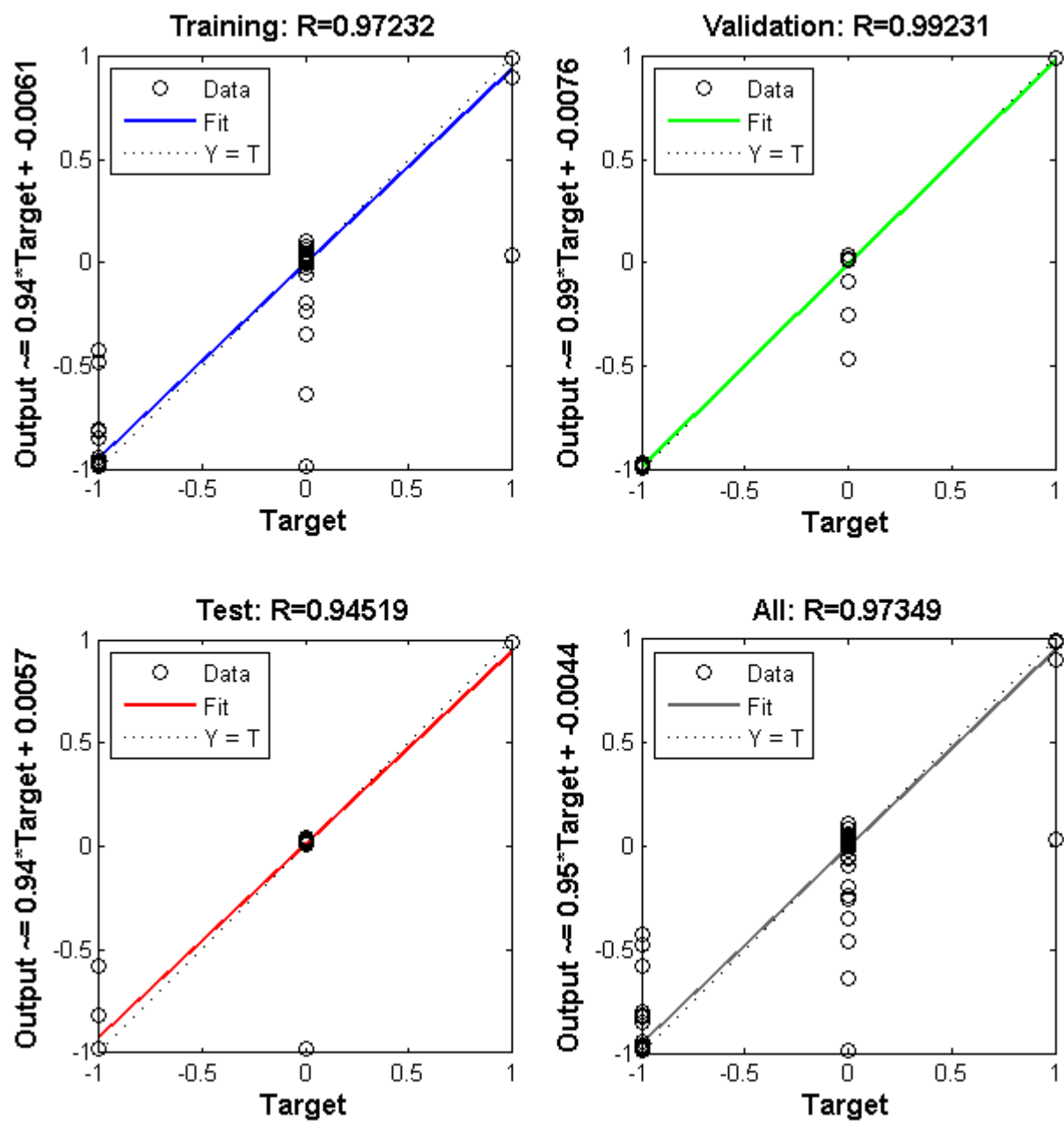


Figure 6.14: Correlation between the target and NN output.

Chapter 7

Challenges and Future Works

In this Chapter we will talk about the biggest challenges we faced during the implementation, and the proposed solution for each one. Then the conclusion will be stated and the chapter will be ended up with the future works.

7.1 Main challenges

Through the preparation of our localization system, we have chosen to build the system using a combination of wireless sensor networks and Neural Networks. The most important role of XBee modules was to provide us with the RSSI measurements to enable localization. Extracting RSSI data wasn't as easy as we expected after reviewing many papers related to this topic. The RSSI values were easily measured by the available hardware between only two nodes in a point to point link but not in a mesh network. For that we started to build our own system that measure the power between the mobile nodes and the whole anchor network.

One popular localization engine is CC2431 from Texas Instruments. CC2431 is a system on chip that implement a hardware location engine. It is able to measure RSSI together with the location of a MN. The localization accuracy provided by this system ranges from 3-5 meters depending on the environment conditions. CC2431 is used in localization researches to develop and test new localization algorithms but unfortunately this system wasn't available for us and that what forced us to waste a lot of time building our own system from scratch.

Our system has many limitations compared with CC2431 since we used a PIC microcontroller to program all nodes in the system while CC2431 uses 8085 microprocessor which is more powerful. In addition CC2431 is equipped with many sensors such as directional sensor which wasn't available for us as mentioned in chapter 4.

Fingerprinting was another challenge in our system. It is well-known that fingerprinting is a tedious process since we have to create RSSI map for the whole location which might need days or weeks to be completed. Combine this with the fact that no two buildings can share

same RSSI map since the structure and environments will affect the RSSI values. This means for every new building we are required to perform this tedious fingerprinting process before the localization system can be applied! However to solve this problem we spent a lot of time building a tool that can measure thousands of RSSI values within few hours!

Another limitation was in X-CTU software that is designed by Digi to interface its XBee transceivers. The problem is that X-CTU was unable to store any measured data or received packet! It can only shows the information on the screen. However, storing the information was essential in order to plug these data in our monitoring system for further processing. So to overcome this problem we design a MATLAB program that can simulate X-CTU's features.

Working on our monitoring system was sequential and cumulative, no step can be done before we successfully finished its previous requirements. Training the Neural Network requires RSSI data to be available which we couldn't get at the beginning of this semester because of all aforementioned problems. We tried to find a sample of power data to train the NN and focus our attention on developing localization algorithms rather than collecting RSSI data, but we hadn't found any useful dataset.

Finally, the most important challenge was having to dedicated lab in the university for graduation projects! It was very bad to spend most of our time working of the corridors, this was a sort of wasting time and killing the creativity. So we wish that the electric department will provide reopen the graduation project lab for next coming students.

7.2 Future Works

There are several ideas that could be implemented on the monitoring system in the future to measure or improves its performance.

1. Adding the Compass Module with Tilt Compensation - HMC6343" sensor can improve its performance very well and we have programmed the PICs to take the sensor readings so the next step is just adding the sensor to the mobile node circuit.
2. Localization system can be tested outdoor to measure the difference in performance between it and indoor environment.
3. Different algorithms could be applied and tested on the system. And it will be very interesting to try capture and combine other types of physical signals. Direct example might be combining image processing field that is based on capturing images from the field with RSSI measurements to improve accuracy.

Bibliography

- [1] Center of Development Disability Health, FACT SHEET: *Working with people with intellectual disabilities in healthcare settings*, Victoria, 2008, www.cddh.monash.org
- [2] S. A. Mitilneos, D. M. Kyriazanos, *Indoor localization with wireless sensor networks*, Progress In Electromagnetics Research Vol. 109, pp. 441-474, 2010.
- [3] M. Jevring, *Automatic Management of Bluetooth Networks for Indoor Localization*, M.Sc in Software Engineering, University of Twente, 2008.
- [4] J. Bachrach and C. Taylor, *Localization in Sensor Networks*, Handbook of Sensor Networks: Algorithms and Architectures, I. Stojmenovic, Ed., 2005.
- [5] A. Pal, *Localization Algorithms in Wireless Sensor Networks: Current Approaches and Future Challenges*, Network Protocols and Algorithms, Vol. 2, No. 1, 2010.
- [6] L. Eslim, H. Hassanein, *Localization in Medical Sensory Systems*, Queen's University, Canada, 2012.
- [7] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, *A survey on sensor networks*, Communications Magazine, IEEE, vol. 40, no. 8, pp. 102–114, Aug 2002.
- [8] H. Liu, H. Darabi, P. Banerjee, and J. Liu, *Survey of wireless indoor positioning techniques and systems*, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions, vol. 37, no. 6, pp. 1067–1080, Nov. 2007.
- [9] S. Nikitaki, P. Tsakalides, *Decentralized Indoor Wireless Localization Using Compressed Sensing of Signal-Strength Fingerprints*, 2012.
- [10] A. Redondi, M. Tagliasacchi, M. Cesara, L. Borsani, P. Tarro, and F. Salice, *LAURA - Localization and Ubiquitous monitoring of patients for health care support.*, 2011.
- [11] A. Bekkelien, *Bluetooth Indoor Positioning*, Ms.C thesis, University of Geneva, 2012.
- [12] A. Pal, *Localization Algorithms in Wireless Sensor Networks: Current Approaches and Future Challenges*, Network Protocols and Algorithms, ISSN 1943-3581, Vol. 2, No. 1, 2010.
- [13] T. S. Rappaport, *Wireless communications: principles and practice*, Prentice Hall PTR, University of Michigan, 1996.

- [14] A. F. Molisch, *Wireless Communications*, John Wiley & Sons Ltd, USA, 2011.
- [15] X. Wang, H. V. Poor, *Wireless Communication Systems: Advanced Techniques for Signal Reception*, Prentice-Hall Professional, 2004.
- [16] J. Torres-Solis, T. H. Falk, T. Chau, *A review of indoor localization technologies: towards navigational assistance for topographical disorientation*, Bloorview Research Institute, Canada, 2010.
- [17] R. Gupta, H. P. Singh, *An introduction to artificial neural networks*, Narosa Publishing House, India, 2001.
- [18] D. Kriesel, *A Brief Introduction to Neural Networks*, 2011,
[http : //www.dkriesel.com/en/science/neural_networks](http://www.dkriesel.com/en/science/neural_networks).
- [19] M. Minsky and S. Papert, *Perceptrons*, MIT Press, Cambridge, Mass, 1969.
- [20] E. Fiesler, *Neural Network Classification and Formulization*, Computer Standereads & Interfaces, vol.16, 1994.
- [21] G. Kumar Jha, *Artificial Neural Networks and its Applications*,
- [22] S. S. Riaz, *The role of ZigBee technology in future data communication system*, India, 2007.
- [23] A. Lonza, *RF management application using ZigBee network*, Germany, 2010.
- [24] S. C. Ergen, *ZigBee/IEEE 802.15.4 Summary*, 2004.
- [25] J. Lee, Yu-Wei Su, C. Shen, *A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi*, Taiwan, 2007.
- [26] W.Stallings, L.Brown, *Computer Security Principles and Practice*, Prentice Hall, USA, 2008.
- [27] Defuse Security, *Salted Password Hashing - Doing it Right*, Last Modified: November 18, 2012, [http : //crackstation.net/ hashing – security.htm](http://crackstation.net/ hashing – security.htm)
- [28] H.Cabrera, B.Bagnall, J.Faircloth, S.Goldberg, M.D, P.Chen, *C# For Java Programmers*, Syngress Publishing, Inc., USA, 2002.
- [29] B. Evjen, J. Glynn, C. Nagel, M. Skinner, K. Watson, *C# 2008*, Wiley Publishing Inc., 2008.
- [30] User manual: *XBee-PRO DigiMesh 2.4 RF Modules*, Digi International, Inc., 2012.
- [31] G. Cybenko. *Approximation by superpositions of a sigmoidal function*. Mathematics of Control, Signals, and Systems (MCSS), 2(4), pp. 303314, 1989.