Winter 2018

Occupancy Detection using Wireless Sensor Network in Indoor Environment

Farah Ferdaus
University of New Hampshire, Durham

Occupancy Detection using Wireless Sensor Network in Indoor Environment

By

Farah Ferdaus

Bachelor of Science in Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, 2015

Thesis

Submitted to the University of New Hampshire

in Partial Fulfillment of

the Requirements for the Degree of

Master of Science

in

Electrical Engineering

December, 2018

This thesis has been examined and approved in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering by:

Thesis Director, Nicholas J. Kirsch, Ph.D.
Associate Professor (Electrical and Computer Engineering)

John R. LaCourse, Ph.D.
Professor (Electrical and Computer Engineering)

Edward Song, Ph.D.
Assistant Professor (Electrical and Computer Engineering)

on 11/28/2018.

Original approval signatures are on file with the University of New Hampshire Graduate School.

DEDICATION

I would like to dedicate this thesis to my family. A special feeling of gratitude to my loving parents, my mother, Shahanara Begum, and my father, Abdullah Faruque for their encouragement and continuous support to achieve my dreams and goals. My beloved better half, B. M. S. Bahar Talukder, who is always inspiring and supporting me in my hard times and providing some great advice when I ran into problems, these are not only helpful but also lifesaving. My siblings, Shafayat Hossen, and Shuhail Hussain, who have always been there to cheer me up and stood by me through the good times and also in bad. They are my greatest sources of inspiration to go through tough times while keeping my head high.

ACKNOWLEDGEMENT

I would like to thank Dr. Nicholas J. Kirsch for giving me the opportunity to work on this project and allowing me to work on something that I am truly passionate about. It is also my privilege to express my deepest and sincere appreciation to Dr. Kirsch for his direction and patience during the work associated with this thesis project. Dr. Kirsch was always available to answer questions despite an extremely demanding schedule. His guidance with thought-provoking objectives and support from the initial to the final level enabled me to develop a profound understanding and gain a wealth of engineering knowledge.

I would also like to express my deepest thanks to the members of my thesis committee, Dr. John R. LaCourse, and Dr. Edward Song for their willingness to serve on the committee. Their constructive and insightful comments helped me to improve my thesis.

I would also like to thank my fellow graduate students Jean Lambert Kubwimana, and Omid M. Kandelusy from the Wireless Systems Laboratory for assisting with the experimental setups. Last but not least, thank you to my family and friends who supported me throughout my education.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

HVAC	Heating, Ventilation and Air Conditioning
GPS	Global Positioning System
WSN	Wireless Sensor Networks
MMSE	Minimum Mean Square Error
GSM	Global System for Mobile Communications
NHAPS	National Human Activity Pattern Survey
PIR	Passive Infrared
CO_2	Carbon Dioxide
CO	Carbon Monoxide
TVOC	Total Volatile Organic Compounds
RH	Relative Humidity
RF	Radio Frequency
RSS	Received Signal Strength
PL	Path Loss
kNN	k Nearest Neighbor

SVM	Support Vector Machine
SMP	Smallest M-vertex Polygon
AoA	Angle of Arrival
ToA	Time of Arrival
TDoA	Time Difference of Arrival
MLE	Maximum Likelihood Estimation
PML	Probability Based Maximum Likelihood
EPML	Enhanced Probability Based Maximum Likelihood
MSE	Mean Square Error
LS	Least Square
SDR	Software Defined Radio
USRP	Universal Software Radio Peripheral
SIM	Subscriber Identity Module
FFT	Fast Fourier Transform
PSD	Power Spectral Density
UHD	USRP Hardware Driver
PSTN	Public Switched Telephone Network
GCC	GNU Compiler Collection

GRC	GNU Radio Companion
USB	Universal Serial Bus
VID	Vendor ID
PID	Product ID
TDMA	Time-Division Multiple Access
SIP	Session Initiation Protocol
PBX	Private Branch Exchange
ODBC	Open Database Connectivity
ARFCN	Absolute Radio Frequency Channel Number
PC/SC	Personal Computer/Smart Card
USIM	Universal Subscriber Identity Module
IMSI	International Mobile Subscriber Identity
MCC	Mobile Country Code
MNC	Mobile Network Code
UL/DL	Uplink/Downlink
RSSI	Received Signal Strength Indicator
LUR	Location Update Request
TMSI	Temporary Mobile Subscriber Identity

IMEI	International Mobile Equipment Identifier
MAC	Media Access Control
MSISDN	Mobile Station International Subscriber Directory Number
SMS	Short Message Service
BTS	Base Transceiver Station
SNR	Signal-to-Noise Ratio
MS	Mobile Station
SACCH	Slow Associated Control Channel
BSC	Base Station Controller
BCCH	Broadcast Control Channel
CLI	Command-Line Interface
NTP	Network Time Protocol
HTTP	Hypertext Transfer Protocol
RAM	Random-Access Memory
CDF	Cumulative Distribution Function
PLMN	Public Land Mobile Network

ABSTRACT

Occupancy Detection using Wireless Sensor Network in Indoor Environment

by

Farah Ferdaus

University of New Hampshire, December 2018

Occupancy detection plays an important role in many smart buildings such as reducing building energy usage by controlling heating, ventilation and air conditioning (HVAC) systems, monitoring systems and managing lighting systems, tracking people in hospitals for medical issues, advertising to people in malls, and to search and rescue missions. The global positioning system (GPS) is used most widely as a localization system but highly inaccurate for indoor applications. The indoor environment is difficult to handle because along with the loss of signals, privacy is a major concern. Indoor tracking has many aspects in common with sensor localization in Wireless Sensor Networks (WSN). The contribution of this work is the demonstration of a non-intrusive approach to detect an occupancy in a building using wireless sensor networks to detect energy from cell phones in a secure facility and perform indoor localization based on the minimum mean square error (MMSE). To estimate the occupancy, the detected cellular signals information such as signal amplitude, frequency, power and detection time is sent to a fusion server, matched

the detected signals by time and channel information, performed localization to estimate a location, and finally estimated the occupancy of rooms in a building from the estimated locations.

CHAPTER 1

INTRODUCTION

The accurate occupancy detection of objects and people in indoor environments has long been considered an important building block for ubiquitous computing applications[1], [2]. In recent years, wireless devices are getting more powerful and pervasive. Current wireless devices often support more than one radio technology, e.g., WiFi, Bluetooth and the Global System for Mobile Communications (GSM). Most research on indoor localization systems has been based on the use of short-range signals, such as WiFi[3]–[5], Bluetooth[6], ultra sound[7], or infrared[8]. The wide availability of GSM networks encourages research on the use of GSM as a common radio technology. In addition, GSM signals appear more stable over time in comparison to WiFi or Bluetooth signals [9], [10]. In this thesis, we also opted for the use of GSM handset signaling. This chapter will present problems and the importance of occupancy detection, related previous work, this project's contribution to science, and the organization of this thesis.

1.1 Motivation

Ubiquitous smartphone and location information enable new features of location-based services around local navigation, retail recommendation, proximity social networking, and location-aware advertising. Recently, the focus is also shifting geographically from outdoor to indoor. The indoor location market will be more enormous than outdoor since people spend more than 87% of the time indoor in the daily activities at the office, restaurant or home[11],[12]. Recent

studies show that the percentage of time spent in indoors is increasing nowadays. On average people spend 90% of their time indoors[13].

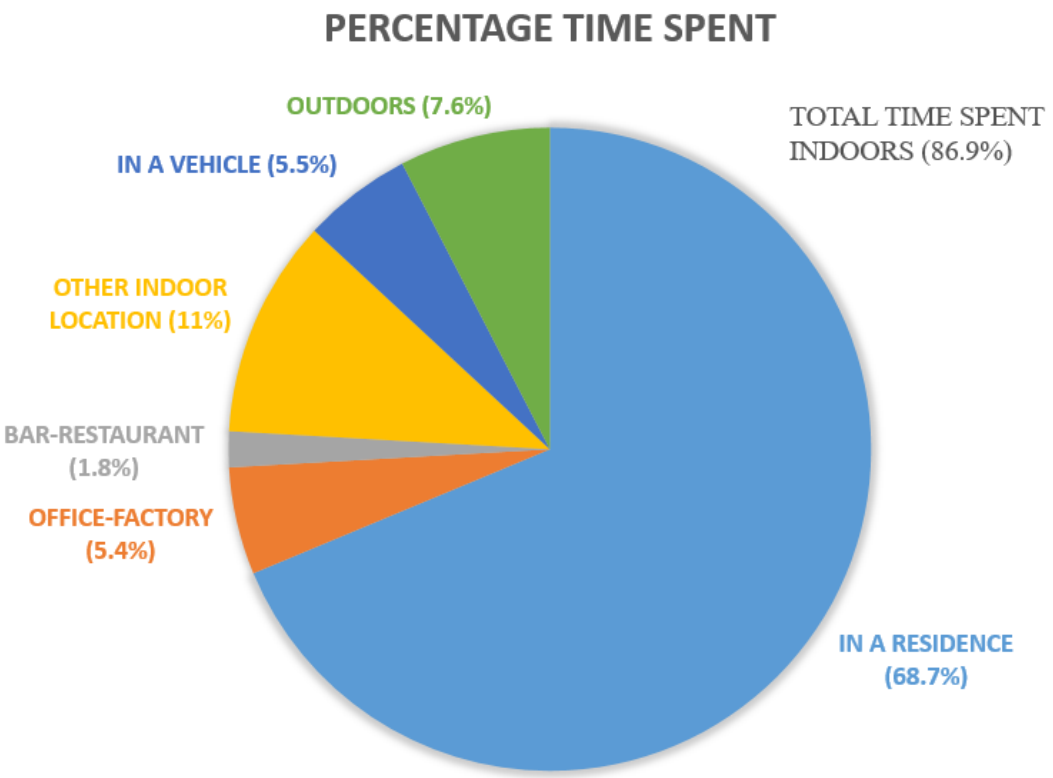


Figure 1.1: This pie chart from the NHAPS study shows that Americans spend 86.9% of time indoors, plus another 5.5% inside a vehicle[12]

Figure 1.1 shows broadly grouped statistics on the mean percentage of time that National Human Activity Pattern Survey (NHAPS) respondents spent in six different locations (residence, office-factory, bar-restaurant, some other indoor location, enclosed vehicle, and outdoors). Of the total time spent by all respondents on the diary day, 69% was spent, on average, in a residence (Figure 1.1). Approximately 87% of the time was spent indoors and 5-6% in a vehicle, with the remaining 7-8% spent outdoors. Time spent indoors (composed of time in a residence, in an office or factory, in a bar or restaurant, or in some other indoor location) and outdoors are represented by

different colored shaded slices. The percentages in the figure are the mean percentages taken over individual percentages for people in the NHAPS sample. Individual percentages were calculated from the time spent in each location over the total amount of time spent, which was equal to 24 h (1440 min) for each individual[12].

In dynamic environments, where the setting and occupancy keep changing, knowing occupancy information, including the number of the occupants and where they are located, can be beneficial in energy management. The other applications are public safety and services, security and emergency response such as search and rescue missions, asset tracking in hospitals etc.[14]. For example, congestion management in public places like malls, and tracking team members and assets on missions in the dark, or in crowded locations etc. Another example would be a section of the building that has more people will require more cooling or heating as compared to a section where a lesser number of people are present. Therefore, occupancy detection in an indoor environment is becoming increasingly important.

Building energy management and the necessity to reduce overall energy consumption is becoming an increasingly important topic. HVAC systems currently account for approximately half of the energy consumed in buildings in developed countries[15]. It is therefore essential to design and operate HVAC systems in an energy-efficient manner to meet low-energy targets. The HVAC need is strongly related to the occupancy of the building due to the air pollution and heat load generated by human metabolism, and their use of electrical equipment[16]–[19]. Conventional rule-based HVAC operation typically relies on a daily static occupancy schedule and real-time measurements of air temperature and/or CO_2 concentration to determine the HVAC need. However, several studies have suggested that significant energy savings can be achieved by using feedback from sensor-based occupancy detection when operating HVAC systems[20]–[29]

and lighting[30]–[32]. These studies demonstrate a significant theoretical energy-saving potential, i.e. when perfect occupancy detection and predictions are assumed. However, simulation results of Pedersen et al. [33] show that the accuracy of occupancy detection and predictions affects the theoretical energy-saving potential significantly. This calls for the development of reliable yet simple and inexpensive real-time occupancy detection approaches to include occupancy information when optimizing real-time HVAC operation.

Occupancy detection system in indoor environments (Indoor localization) is a technique for locating people or objects inside a building. Technologically, outdoor localization techniques cannot be directly moved to indoor[34]. GPS works almost perfectly in the outdoor environment but its accuracy, coverage, and quality deteriorate significantly in small-scale indoor places. Satellite-based GPS data can be difficult or impossible to access when a user is inside, or insufficient to accurately locate someone in a multistory building. This is a challenging problem for two reasons such as participant inclusion and access to trackable signals. The main challenge is the variability and multipath in the environment. Pre-deployed fixed infrastructure can be used to overcome the challenges. A few methods that use a pre-deployed infrastructure are as followed: Microsoft RADAR[3], RFID (Radio Frequency Identification) and the Vision-Based Approach[14]. This research is greatly inspired by the previous works[35]–[38] which is prototyped using OpenBTS, GNU Radio, and software-defined radios to verify the performance of indoor localization in the real-world environment.

1.2 Thesis Objective

Our goal is to develop an indoor occupancy detection scheme to make the system inexpensive with limited resources as well as environmentally friendly. Besides, the primary focus is to develop the detection technique that ensures accuracy and privacy. As the cell phone has become a ubiquitous device (Figure 1.2 shows the number of mobile phone users (in billions) worldwide from 2015 to 2019), therefore, detecting occupancy by tracing the location of the cell phone signal is a smart approach. To achieve this goal, the system requires the use of an RF (Radio Frequency) signal receiver to capture the GSM uplink frequencies.

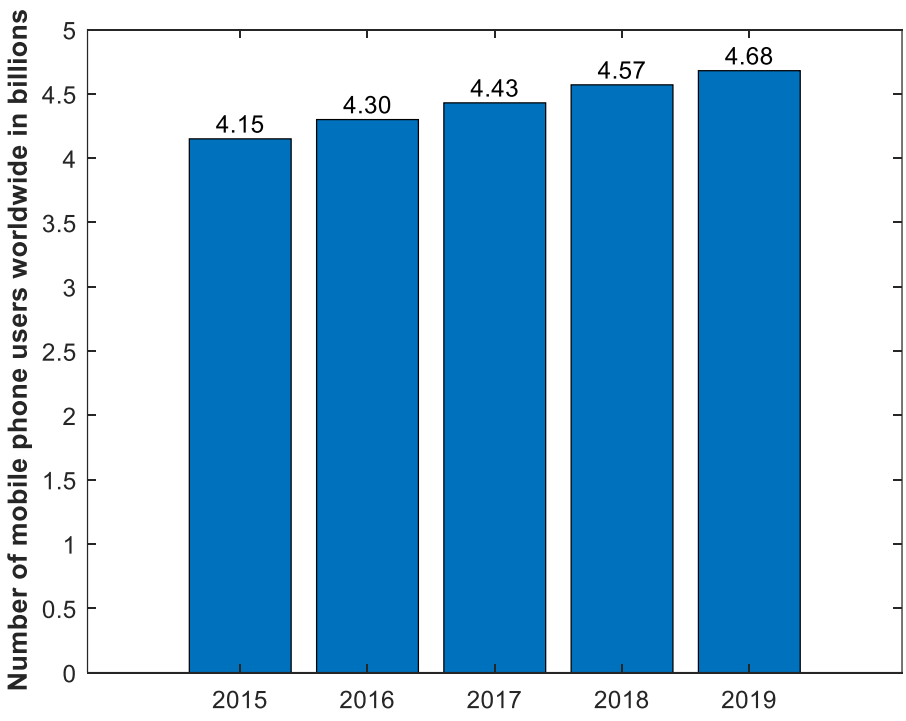


Figure 1.2: Number of mobile phone users worldwide from 2015 to 2019 (in billions)[39]

The design of an occupancy detection service has several challenges that are related to the nature of the wireless medium and the GSM standards. These challenges are: how to capture GSM radio signals, and how to identify the channel information in order to provide the correct services? Facing these challenges requires an uplink receiver that captures, processes and analyzes GSM radio signals generated by the mobile devices. In this thesis, a wireless sensor network is used as a receiver for capturing GSM uplink signal frequencies. Moreover, our conducted measurements also revealed the insights on the GSM communication.

1.3 Related work

Current occupancy detection approaches can be divided into two groups: image-based methods and data-based methods. Image-based methods[40]–[43] rely on camera technology to detect occupancy. However, installing cameras can be perceived as a privacy violation and often represents an additional investment and running cost to a building project. Zhao et al.[44] obtained convincing occupancy detection in offices using a Bayesian belief network, which is a probabilistic graphical model (a type of statistical model) that represents a set of variables and their conditional dependencies via a directed acyclic graph, together with information from e.g. WiFi, GPS location, chair sensor, and keyboard and mouse sensor. However, some occupants may still consider these sensor data to be intrusive. Therefore, an inexpensive and non-intrusive alternative is required for indoor occupancy detection.

Currently, the most commonly used sensor data for occupancy detection is data from passive infrared (PIR) sensors[45]–[49] which are installed primarily for energy efficient operation of lighting. However, relying solely on PIR sensor data as a detection of occupancy is rather uncertain since the sensors do not capture immobile occupants or occupants that are outside the PIR sensor's

field-of-view[50]. Data from indoor climate sensors already used for conventional HVAC control seems like another basis for occupancy detection. The carbon dioxide (CO_2) level in a room is an attractive indicator as it is a direct consequence of human presence and, to some extent, independent of whether the occupants are moving or not[50]. The disadvantage of using the CO_2 mass balance equation is that it requires detailed information about the physical room conditions (e.g. room volume, mechanical air change rate, window/door openings, occupant CO_2 production, outdoor CO_2 concentration) which can be difficult to determine and vary in time, thus making it subject to some uncertainty.

Utilizing sensor data to establish statistical models is another widely used approach[25], [49], [51]–[55]. Data-based occupancy detection based on measurements of CO_2, carbon monoxide (CO), total volatile organic compounds (TVOC), small particles (PM2.5), acoustics, illumination, PIR, temperature and relative humidity (RH) from an open-plan office environment was reported in[25], [49], [55]. The disadvantage of current methods for data-based occupancy detection is that they need prior information to work in practice. The above-mentioned methods based on physical models (mass balance equation) require detailed a priori information about the physical conditions of each room in the building. This type of method, therefore, needs to be set up manually before application. The above-mentioned statistical models need extensive training data and can therefore not be applied right after they are installed.

Thus, an alternative method to occupancy detection that overcomes the practical disadvantage of the model-based approaches is the novel plug-and-play method presented in this thesis to detect indoor occupancy. The proposed method was tested in long-time duration tests and evaluated in terms of its ability to detect occupancy compared to the ground truth.

1.4 Contribution

The contribution of this work is the implementation of an indoor occupant detection scheme by utilizing occupant-carried cellular devices. The proposed architecture is prototyped using OpenBTS, GNU Radio, and software-defined radios. All of the processing, such as detection of the energy of radio frequency from cell phones, transmission of the processed detected signal to a fusion server, performing localization algorithm over the processed data, is done in real-time. Therefore, significant energy savings can be achieved by operating HVAC controllers using a feedback from sensor-based occupancy detection methods. Furthermore, the proposed system model is applicable in any type of room shape and dimension and does not require the placement of the sensors in each room which potentially reduces cost. Finally, these sensors do not decode received cellular signals, so the privacy and identification of occupants are not of concern.

1.5 Thesis Organization

This section provides an explanation of each chapter of this thesis. This section helps the reader navigate this thesis and facilitates efficiency when particular sections are needed for review or reference.

Chapter 2 presents background information pertaining to this thesis project. Fundamental concepts required for understanding the occupant detection scheme using indoor localization and experimental results of this thesis project are explained.

Chapter 3 presents the hardware and software used to develop the prototype architecture and wireless sensor networks for occupancy detection. This chapter is important because it provides

specific information on how this project can be implemented. This chapter can also be referenced in order to facilitate other research projects that involve similar processes.

Chapter 4 presents empirical results and details the measurements made in Kingsbury Hall. It identifies the properties related to the proposed framework and the related assumptions. Tables and figures are provided in this chapter in order to organize results. The analytic information presented in this chapter proves this thesis is a contribution to science.

Chapter 5 concludes this thesis project, summarizing the results and the impact of the project. Also presents a focus for future work on this project. This chapter is important because it provides insight on what research needs to be conducted in order to increase the contribution of the systems presented in this thesis project.

CHAPTER 2

BACKGROUND INFORMATION

Path loss model, Received Signal Strength (RSS), sensor network and localization method are defined and conceptualized in this chapter. This information is presented in support of the experimental process and the results of this thesis project. The fundamentals of wireless communication are essential to this research. The first portion of this chapter explains basic propagation loss in wireless communications and signal propagation. Secondly, indoor localization is explained. Indoor localization is included to show how occupancy detection can be done by relative location estimation. Then, wireless sensor networks are explained so that the best number of sensors and the positioning of sensors for best results is understood. Finally, the Minimum Mean Squared Error (MMSE) is explained. The MMSE explains how this method works for localization problems. These processes are important because Chapter 3 uses these processes to design the wireless sensor network.

2.1 Propagation Loss

The mobile radio channel or environment places fundamental limitations on the performance of wireless communication systems. The transmission path between the transmitter and receiver can vary from a simple line of sight to one that is obstructed by buildings. These obstacles cause the wireless environment or channels to be random and unpredictable as well as complex. Movements by transmitter or receiver obstacles also may impact the signal levels.

There are five basic propagation mechanisms that mainly impact mobile communication by affecting the received power constructively or destructively[56]. These are reflection, diffraction, scattering, shadowing and path loss. These large-scale effects have an impact on the path loss between a transmitter and receiver.

Reflection occurs when an electromagnetic wave impinges upon an obstacle which has a large size in comparison to the wavelength of the propagating wave. Reflection may occur from any surface; from the surface of the earth and from buildings and walls.

Diffraction occurs when the electromagnetic wave is obstructed by a surface that has sharp edges or irregularities. The secondary waves resulting from the obstruction are present throughout the space and behind the obstacle. Diffraction gives rise to bending of waves around the obstacle, even though the line of sight might not exist between the transmitter and receiver. This phenomenon brings about a change in amplitude, phase, and polarization of the incident wave at the point of diffraction.

Scattering occurs when the environment through which the wave travels consists of objects which are smaller in dimension than the wavelength, and where the number of obstacles per unit volume is large. Scattered waves are produced by rough surfaces or irregularities in the channel.

Shadowing is the variation or the uncertainty in the environment. It is a zero mean Gaussian distributed random variable (in dBm) with standard deviation, σ. Shadowing varies from building to building and even floor to floor inside a building.

Path loss (PL) or path attenuation is the difference in power or the loss in the signal strength due to the environment while traveling from the transmitter to the receiver. This loss of power is due to the radio wave propagation and objects in the environment (urban or rural, vegetation and

foliage), propagation medium (dry or moist air), the distance between the transmitter and the receiver, and the height and location of antennas. Theoretical and measurement based propagation models indicate that average received signal power decreases logarithmically with distance, in both the indoors and outdoors[56]. The PL does not take antenna gains or cable losses into consideration. PL is simply the attenuation present in the environment due to effects of free space propagation loss, reflection, refraction, scattering, shadowing, diffraction, aperture-medium coupling loss, and absorption. The PL is expressed as a function of the distance by using the path loss exponent, *n*. The value of *n* depends on frequency as well as the environment and also highly depends on the obstacles present in the path of the signal. In Equation (2.1), d_0 is the reference distance, *d* is the distance between the transmitter and receiver, *PL(d)* is the path loss at distance *d* (in dBm) and *n* is the path-loss exponent which indicates the rate at which the path loss increases with distance, d_0 is the far-field distance.

$$PL(d) \propto \left(\frac{d}{d_0}\right)^n \tag{2.1}$$

Equation (2.1) can be written as follows where *P(d)* is the receiver signal strength at distance *d* (in dBm), $P_0(d_0)$is the received signal strength at distance d_0 (in dBm).

$$P(d) = P_0(d_0) - 10n \log\left(\frac{d}{d_0}\right) \tag{2.2}$$

In free space, *n* can be equal to 2 while in an environment with many obstacles, *n* will have a larger value. Table 2.1 lists the typical path loss exponents in various environments as mentioned in[56], [57].

Table 2.1: Path loss exponents based on different environments[57]

ENVIRONMENT	PATH LOSS EXPONENT, n
Free Space	2
Urban area cellular radio	2.7 - 3.5
Shadowed urban cellular radio	3 - 5
In-building line of sight	1.6 - 1.8
Obstructed in-building	4 - 6
Obstructed in factories	2 - 3

As in this thesis, indoor environments are dealt with, so the effect of materials generally present in indoors are needed to examine. Extensive surveys have been done on the probable value of the PL exponent in various buildings to find out the relation of *n* with power and increasing distance[34]. For example, signal levels greatly vary depending on whether doors are open or closed inside a building. Partitions are of two types: hard partitions and soft partitions. Hard partitions include parts of building structure like piping, walls, floors, windows etc. In contrast to hard partitions, soft partitions include partitions or materials that can be moved and do not span to the ceilings, like metallic devices, chairs, etc. The partition type also has a significant effect on *n*.

2.2 Indoor Localization Methods

The development of real-time locating systems has become an important add-on to many existing location-aware systems. While GPS has solved most of the outdoor locating problems, it fails to repeat this success indoors. A number of technologies have been used to address the indoor tracking problem. Indoor Localization is a system to locate objects or people inside a building using lights, radio waves, magnetic fields, acoustic signals, or other sensory information collected by mobile devices[58]. The research of localization has become a more and more important topic

with the popularity of ubiquitous mobile computing. In an indoor environment, many miniaturized wireless and sensing technologies have shown giant potential in positioning applications. The traditional indoor localization methods are:

- Fingerprinting
- Dead-reckoning
- Triangulation

2.2.1 Fingerprinting

Fingerprinting is a Radio Frequency based scene analysis. This method has two stages i.e. offline (training/survey) and online (query/run-time use)[59], [60]. Offline stage collects features known as fingerprints. Online stage estimates the location by matching online measurements against previously collected fingerprints. There are five common approaches to perform the analysis. These are Probabilistic, k Nearest Neighbor (kNN), Neural Networks, Support Vector Machine (SVM), and Smallest M-vertex Polygon (SMP)[61].

2.2.2 Dead-reckoning

Dead-reckoning is a device-centric and portable method less dependent on installed infrastructure. It starts with a known initial position, then estimates displacement and direction. Error accumulation is done over time and distance traveled. This method usually requires an accompanying alternative error control and reduction techniques[61].

2.2.3 Triangulation

Triangulation method is based on the geometric properties of triangles to determine location. It requires either range and/or direction information from/to reference points. The range-based case essentially uses the angle or distance information to estimate the location and distance measurements mostly through radio signals and ultrasound[3], [7], [62]. Both fingerprinting and dead-reckoning methods require an extra stage, offline stage, and alternative error control and reduction stage respectively, so to avoid the requirement of the additional stage in this thesis, the triangulation method was used. The next section describes more detail about the triangulation method.

2.3 Triangulation

The need for localization has brought forward a multitude of different approaches that vary depending on the type of information to be extracted from the signal and the environment. Based on signal metrics, the most popular and prevalent methods are[63], [64]:

- Angle of Arrival (AoA) (triangulation)
- Time of Arrival (ToA) (trilateration)
- Time Difference of Arrival (TDoA)
- Received Signal Strength (RSS)

2.3.1 Angle of Arrival (AoA)

AoA determines the position of the user by measuring the angle at which the signal arrives at the receiver from the transmitter[34], [65]. Directional antennas have the capability to record the angle of arrival. However, this method is highly prone to multipath fading and requires a Line of Sight (LOS) with the receiver, which cannot be ensured in indoor environments. AoA is inconvenient because it involves geometric relationships that are used to locate the intersection of two lines.

2.3.2 Time of Arrival (ToA)

ToA measures the exact distance by using the travel time of the signal from the transmitter to the receiver. ToA systems are based on the precise measurements of the arrival time of a signal transmitted from a mobile device to several receiving sensors. Signals travel with a known velocity which is equal to the speed of light. ToA uses the equation *Distance = Time x Speed* to determine the location of *Distance*[34]. ToA requires precise knowledge of the transmission start time and must ensure that all receiving sensors are accurately synchronized with a precise time source. The amount of time required for a message from station X to arrive at receiving sensors A, B and C are recorded. Given a known velocity (the velocity of light), the distance can be calculated respectively. ToA is inconvenient and difficult because it involves absolute synchronization of the transmitter and the receiver which is often not possible.

2.3.3 Time Difference of Arrival (TDoA)

TDoA is similar to the ToA as both belong to the trilateration group which involves the intersection of the radii of the circles. ToA requires the synchronization of the transmitters and

receivers while TDoA requires the synchronization of the receivers. TDoA technique uses the relative measurements of time at each receiving sensor. The TDoA is calculated between the locations of sensors B and A as the positive constant *k*. The value of the TDoA between B and A can be used to construct a hyperbola with the foci at the receiving sensors A and B. The hyperbola represents all points on the x-y plane and the distance difference between the foci is *k(c)* meters. The unknown node lies along the hyperbola.

2.3.4 Received Signal Strength (RSS)

RSS is the relative signal strength at the receiver. The higher the RSS, the stronger the signal. Sensors or receivers deployed throughout an area of interest can sense the signal strength from a transmitter[66]. From this RSS[66], and signal path loss the distance between the unknown node and the sensor can be estimated. In this thesis, the RSS method has been used for localization.

A minimum of three sensors is always required for triangulation or trilateration with respect to RSS based methods. Popular localization methods include MLE (Maximum Likelihood Estimation), MMSE, PML (Probability Based Maximum Likelihood), and EPML (Enhanced PML).

As an example of the system that is considered in this thesis, Figure 2.1 shows the considered scenario and the process of estimation of location. The cell phones are the position which is tried to estimate, and the triangles are the receivers or sensors. Now the question that arises is how an existing infrastructure can be used for indoor localization.

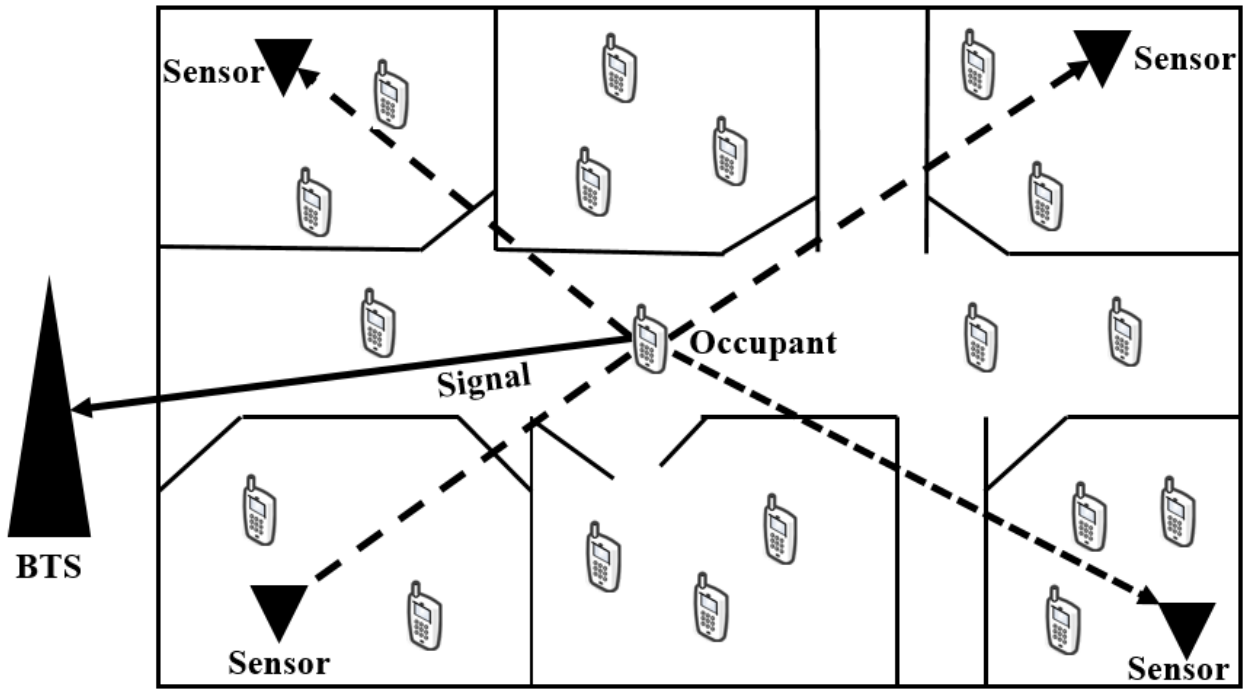


Figure 2.1: The scenario

The main objective in localization is to use the received signal metrics, such as time and signal strength, to estimate the true distance between the receiver and each of the transmitters. The triangulation assumes the sensors to be the centers of circles and the distance between the receiver and transmitter to be the respective radii of the circle. The location is then estimated by determining the point at which all the circles intersect with each other as shown in Figure 2.2. In Figure 2.2 the transmitter positions are marked as A, B, C respectively. Thus, in the simplest form, the triangulation method needs at least three sensors. Once all the distances between the receiver and the transmitters are determined, the triangulation method can be used on them to determine the location of the unknown node $(\hat{x},\hat{y})$. The location of the unknown node would then be the intersection of the circles.

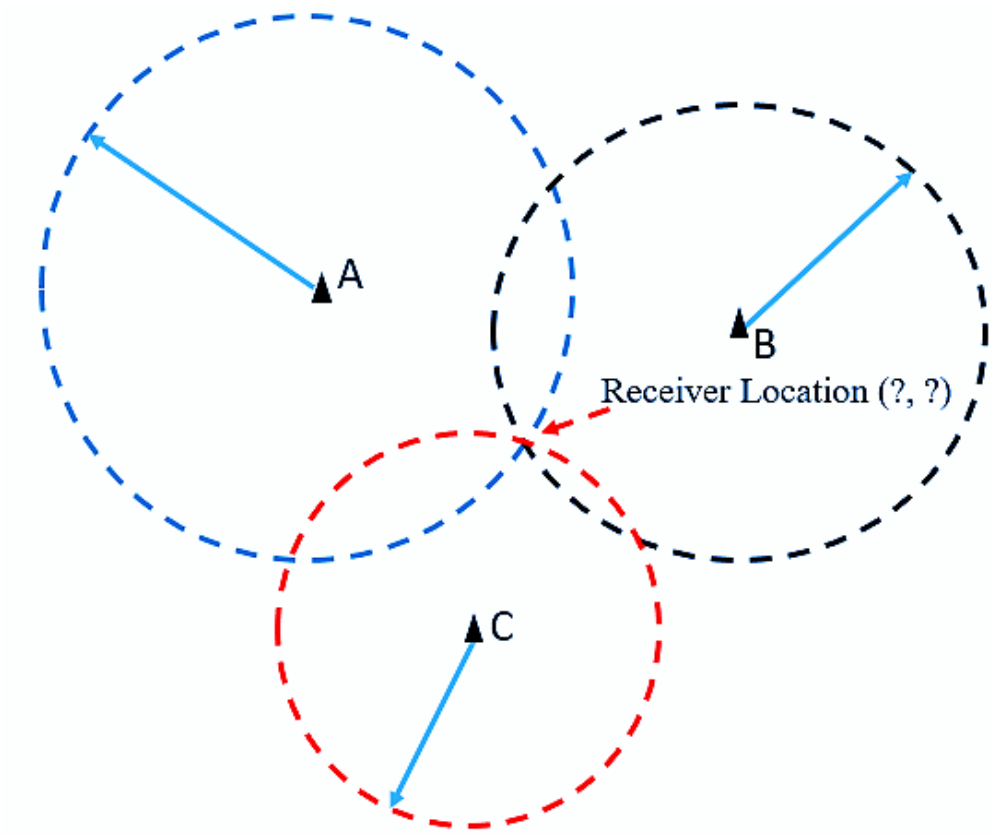


Figure 2.2: Simple Triangulation based on distance measurements between the receiver and transmitter

This simple concept of triangulation has a very basic prerequisite that would only work if the distance measurements are perfect. In other words, all the circles have to coincide or intersect at a single unique point. The distance between the receiver and the transmitter can be determined by the distance equation. Let (X, Y) be the receiver location. Let (x_i, y_i) be the transmitter locations where $i = 1, 2, 3...N$ and N is the total number of transmitters present. The distance between the receiver and each transmitter can be given by the Euclidean distance equation (2.3).

$$D_i = \sqrt{(X - x_i)^2 + (Y - y_i)^2} \tag{2.3}$$

If there had been no shadowing and no variations in the environment, it would have been very easy to come up with a situation like Figure 2.2 where all the circles would have a common intersection *point* or a *region* of intersection such that the unknown point is inside the area of intersection. But in reality, wireless channels have multipath and shadowing. As a result, the region in which the unknown point might be present is larger. It is near impossible to determine or predict the area or region of overlap of the circles and thus it becomes very difficult to estimate the location of the unknown point in the indoor environment.

In Figure 2.3 the brown patched area corresponds to the region of overlap when there is no shadowing, while the green area corresponds to the region of overlap when shadowing is involved. Thus it can be easily said that shadowing increases the area of the region of overlap and makes localization a challenge. Essentially, a person's location is tried to determine based on the signal strength received at the sensors marked as A, B, and C. By sensors, it is referred to antennas connected with software defined radios used to receive the signal strength from the unknown nodes (cell phones location).

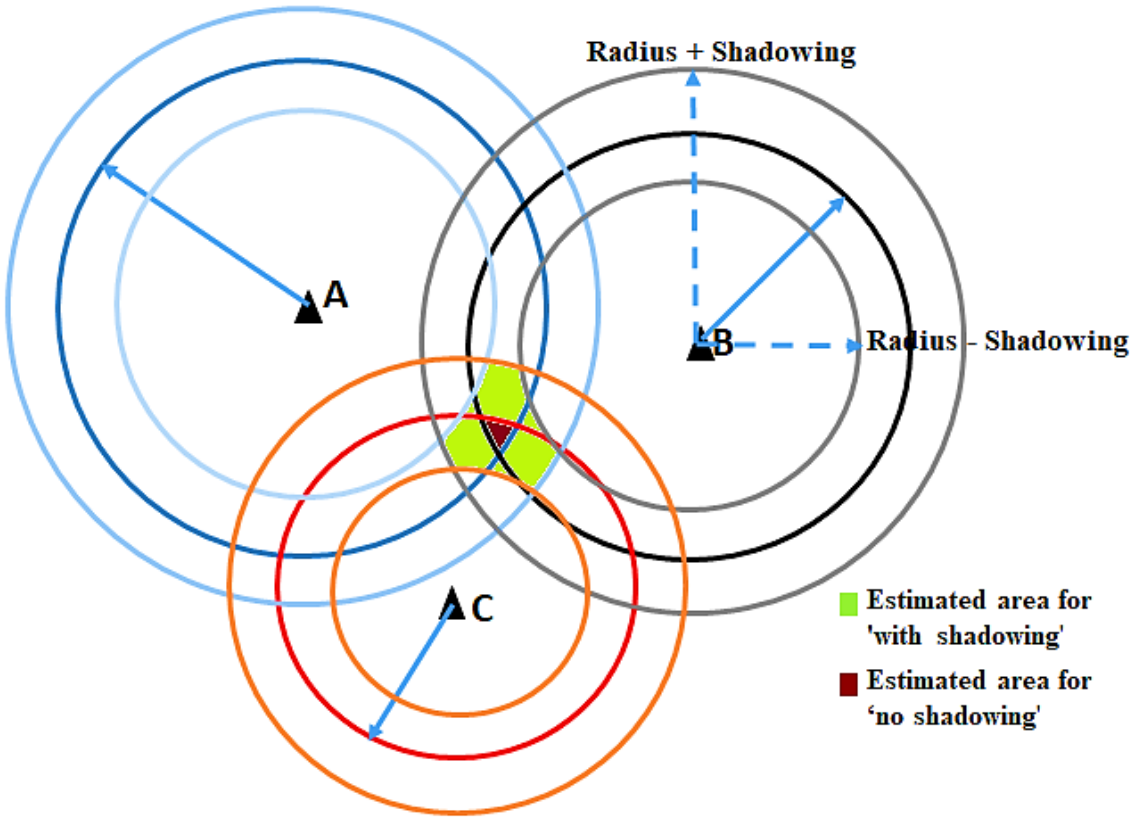


Figure 2.3: The comparison of area of overlap with and without shadowing

2.4 Signal Metrics

The RSS method is based on the fact that the received signal decreases as the distance between the transmitter and receiver increases. Thus, by establishing a relationship or a rate at which the energy decreases, one can easily determine the distance between the transmitter and receiver with the knowledge of the received signal strength.

The log-distance path loss model, as shown in Equation (2.1) and Equation (2.2) indicates that the received signal strength decreases logarithmically with distance, in both outdoor and indoor environments. The relation between the path loss and the received signal strength can

simply be described as $P_r = \{P_t - PL\}$ dBm, where P_r is the received signal power, and P_t is the transmitted signal power. The average received signal strength for an arbitrary separation between the transmitter and receiver, d can be expressed as a function of distance by using a path loss exponent, n[56].

Another inevitable and unavoidable environment variation factor is shadowing. In more precise terms, shadowing is the log-normal distribution that describes the randomness which occurs over a large number of measurement locations which have the same transmitter-receiver separation but have different levels of clutter on the propagation path. This is referred to as *log-normal shadowing*. Log-normal shadowing implies that measured signals have a Gaussian distribution about the distance-dependent mean where the measured signals have values in dBm units

It is possible to find a value of d based on knowledge of the other parameters. But a major problem with RSS is that the power attenuation with increasing distance is not monotonically decreasing. Issues like multipath and shadowing have an effect on the received signal strength. In environments that have various obstacles, there is more randomness which affects the calculation of d on the basis of received signal strength. As a result, Equation (2.2) changes to,

$$P(d) = P_0(d_0) - 10n \log\left(\frac{d}{d_0}\right) + X_\sigma \tag{2.4}$$

Where, X_σ is the zero mean Gaussian random variable with standard deviation, σ (dBm). When plotted on a log-log scale, the modeled path loss is a straight line with a slope equal to *10n* dBm per decade.

2.5 Sensor Network

The number of sensors impacts the performance of the localization method[34]. In this section, the best number of sensors and the pattern in which the sensors should be set for best results are going to be discussed.

Localization methods either use all of the sensors to locate an unknown node or use a subset of sensors which yield the highest RSS. The basic logic behind the localization method is, if the received signal strength is higher, it would mean that the sensor is nearer to the unknown node in comparison to some other sensor, which results in lower RSS. Shadowing is log-normal in nature. The shadowing often results in a higher RSS for a farther sensor or vice versa[34]. That would result in the estimated point moving away from the actual point instead of estimating it better. As a result, it is required to find out the number of sensors that proves to be good for minimum errors in an indoor environment.

To understand the effect of the number of sensors on localization performance, different positions and numbers of sensors were considered for locating an unknown node. To fulfill the triangulation method, first, there had to be at least 3 sensors. Second, the sensors could not be in a line because that would not fulfill the triangulation condition. For the experiment, the sensors were always considered to be pre-deployed and their locations were already known.

In [34]–[36] Ranita Bera et al. describe the method of deploying the sensors to result in a minimum error in indoor localization. There is a significant effect of a) using various numbers of sensors, b) the distance between the sensors and c) the area lying enclosed by the sensors. The test bed was a 100 X 100 meters region divided into a 4 X 4 grid. As a result, each grid width was 33.33 X 33.33 meters as shown in Figure 2.4. More sensors do not necessarily produce better

results rather the excess sensors actually move the estimated point from the actual location of the unknown point. Therefore, the positioning of sensors is more important than the number of sensors.

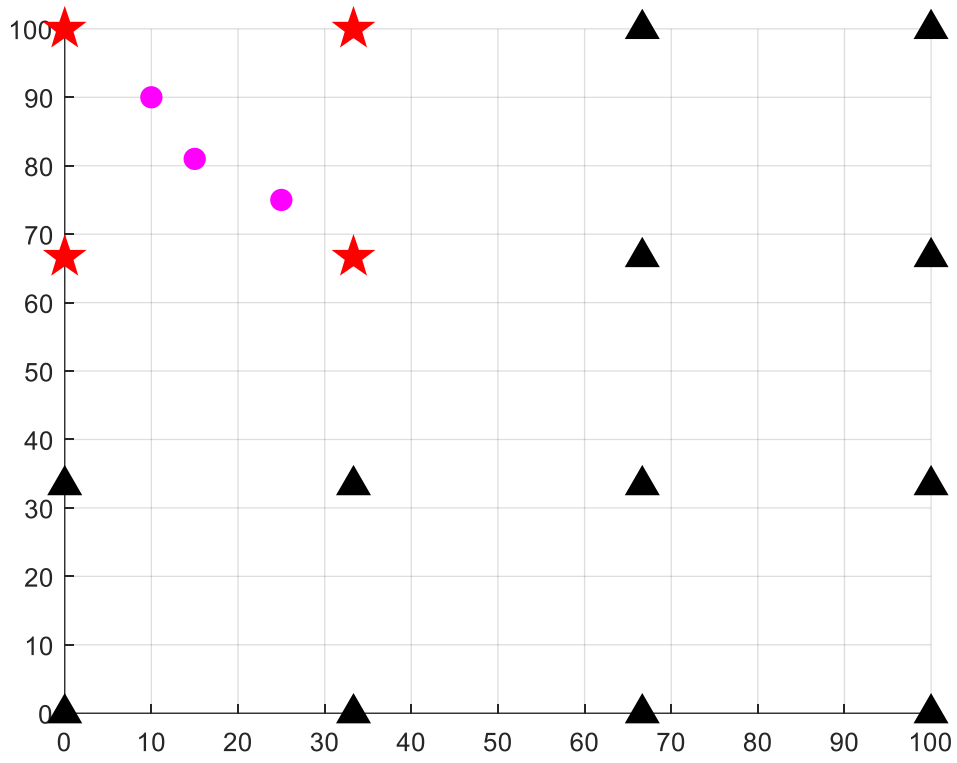


Figure 2.4: Position of sensors

Figure 2.4 shows the four sensors in a square formation that results in the least error in localization[35], [36]. The dots show the location of the unknown nodes. The red stars are the sensors being used for localization while the black triangles represent the remaining sensors that could have been chosen. The four sensors bounded the unknown nodes uniformly which resulted in a uniform low error in localization. Additionally, four sensors placed in a rectangular or square formation was a more feasible scenario than a triangular formation[34]. It is common understanding and knowledge that the centroid region, where the circles of content intersect, has

the least error. As a result, based on the literature review, it can be concluded that sensors bounding a rectangular or square region serves the best.

2.6 Minimum Mean Square Error (MMSE)

Mean Square Error (MSE) is a way of quantifying the difference between values implied by an estimator and the actual value of the quantity being estimated. Mean square error measures the average of the squares of the errors involved in the variables between the value implied by the estimator and the actual value that is being estimated. Suppose, let (x_0,y_0) be the coordinate of the unknown node and $(\hat{x}, \hat{y})$ is the estimated coordinates of the unknown node in totally m times estimations. The MSE is as follows:

$$MSE = \frac{1}{m}\sum_{j=1}^{m}\left(x_0 - \hat{x}_{0,j}\right)^2 + \left(y_0 - \hat{y}_{0,j}\right)^2 \tag{2.5}$$

Now, minimizing the mean square error would mean that the error is minimum and the estimated point nearest to the actual point. This is known as the MMSE estimate.

In [67], Yung-Fa Huang et al. describe the method of MMSE based localization. The RSS at the receiver is attenuated with the distance in wireless communication channels. Moreover, the shadowing effect will fluctuate the RSS with a log-normal distribution. Thus, the RSS at the receiver can be obtained by equation (2.6) where $P_r(d)$ is the received power at an arbitrary distance, d and P_t is the transmitted power. G_t is the transmitter antenna gain and G_r is the receiver antenna gain. Again, λ is the wavelength; d is the distance between the transmitter and receiver and n is the path loss exponent[68]. The shadow fading, L is assumed to be the log-normal distribution model.

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^n L} \tag{2.6}$$

After obtaining the power of RSS, P_r in the unknown node, the estimated distance can be calculated by the following equation (2.7). Where, the transmitter antenna gain, G_t and receiver antenna gain, G_r both is set to 1.

$$d = \left(\frac{\lambda}{4\pi}\right)^{2/n} . \left(\frac{P_t}{P_r}\right)^{1/n} \tag{2.7}$$

After the minimum three distance is obtained, the location of the unknown node is further estimated by the MMSE method[64], [69]. To construct the MMSE method for localization problems, the estimation error equation is formulated for the unknown node (x_0,y_0) by

$$e_n(x_0, y_0) = d_n - \sqrt{(x_n - x_0)^2 + (y_n - y_0)^2} \tag{2.8}$$

Where x_0 and y_0 are the coordinates of the unknown node to be estimated, x_n and y_n are the known coordinates of the *n*th reference node, d_n is the estimated distance for the distance between the *n*th reference nodes and the unknown node. When the number of reference nodes is *N*, the index *n* would be 1, 2, ..., *N*. There are *N* simultaneous equations in (2.8).

To minimize the estimation errors, let $e_n(x_n, y_n) = 0$. Then through mathematical operations in (2.8), the (*N-1*) simultaneous equations are obtained by

$$d_n^2 - x_n^2 - y_n^2 - (d_N^2 - x_N^2 - y_N^2) = 2(x_N - x_n).x_0 + 2(y_N - y_n).y_0 \tag{2.9}$$

By vector form, equation (2.9) can be rewritten as follows:

$$\boldsymbol{w} = \boldsymbol{X}\boldsymbol{b} \tag{2.10}$$

Where

$$\boldsymbol{X} = \begin{bmatrix} 2(x_N - x_1) & 2(y_N - y_1) \\ \vdots & \vdots \\ 2(x_N - x_n) & 2(y_N - y_n) \\ \vdots & \vdots \\ 2(x_N - x_{N-1}) & 2(y_N - y_{N-1}) \end{bmatrix} \tag{2.11}$$

$$\boldsymbol{w} = \begin{bmatrix} d_1^2 - x_1^2 - y_1^2 - (d_N^2 - x_N^2 - y_N^2) \\ \vdots \\ d_n^2 - x_n^2 - y_n^2 - (d_N^2 - x_N^2 - y_N^2) \\ \vdots \\ d_{N-1}^2 - x_{N-1}^2 - y_{N-1}^2 - (d_N^2 - x_N^2 - y_N^2) \end{bmatrix} \tag{2.12}$$

And

$$\boldsymbol{b} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{2.13}$$

respectively. Using the Least Square (LS) method, the estimated coordinates of the unknown node is obtained by

$$\widehat{\boldsymbol{b}} = \begin{bmatrix} \hat{x}_0 \\ \hat{y}_0 \end{bmatrix} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}.\boldsymbol{X}^T\boldsymbol{w} \tag{2.14}$$

Thus the performance of localization can be investigated by mean square error mentioned in equation (2.5).

CHAPTER 3

PROTOTYPE ARCHITECTURE AND SENSOR NETWORK

The contribution of this work is to present an indoor occupancy detection technique using wireless sensor networks. The modeled infrastructure comprised of both hardware and software element enables the wireless sensor network system to detect the RF energy from cellular devices. This feature makes the system highly desirable for real-time applications.

In order to fully evaluate the performance capabilities of the sensor network in indoor occupancy detection, the framework was implemented using OpenBTS, GNU Radio, and software-defined radios. This chapter presents the software and hardware used in the execution of this thesis project.

The implementation process of the proposed prototyped architecture is very important to the outcome of this thesis project because the occupancy detection method is tested in a real-world application. Thus, in order to run the simulations, it is extremely important to build and install the system model precisely. The following chapter also covers each step of the software installation process to run the proposed system.

3.1 System Overview

In this project, RF energy from occupants carried cellular devices to a base station is captured by sensors installed at known locations around a building. In our current framework, the receiver continuously scanned uplink channels of GSM band to check the presence of the signal from the

user. Each time a sensor detects a signal, information about the received power, time and channel of reception, and sensor number was sent to a fusion server. After the detection of RF energy, localization was performed using RSS and sensor information by the server. The server monitored the time and channel stamps of each entry from multiple sensors and matched entries with near-identical time and channel stamps. When three or more entries with near-identical time and channel stamps were matched then the information of those matched entries was fed into the localization algorithm, which estimates the occupant location. This collection occurs continuously, and as multiple occupant locations were collected and updated in the server, an estimate for room occupancy is made. Figure 3.1 illustrates how the cellular signals from occupants were captured by sensors around a building, then the sensors sent data to a fusion server, and the fusion server estimated occupancy from this information.

Ideally, the sensor must continuously detect and capture the RF energy of every nearby cellular device on every possible cellular channel. Sensors must be chosen or designed such that as many cellular channels can be monitored simultaneously as possible.

In the United States, cellular devices use the 700MHz, 750MHz, 800MHz, 850MHz, 1700MHz, 1900MHz, and 2100MHz bands. The bandwidth of each of these bands varies but is as high as 108MHz[38]. Each band is divided into multiple channels, each with a fixed bandwidth. Monitoring every band simultaneously with a single receiver would require an infeasible sampling rate with current technology. Realistically, multiple receivers monitoring each band, or a single receiver constantly sweeping over all bands are possible solutions. A single receiver constantly sweeping over all bands is a less expensive solution, however, this solution may miss the signal of a cellular device on one band while monitoring another, and therefore may underestimate occupancy. For our testbed, the search is restricted to a single band for proof-of-concept.

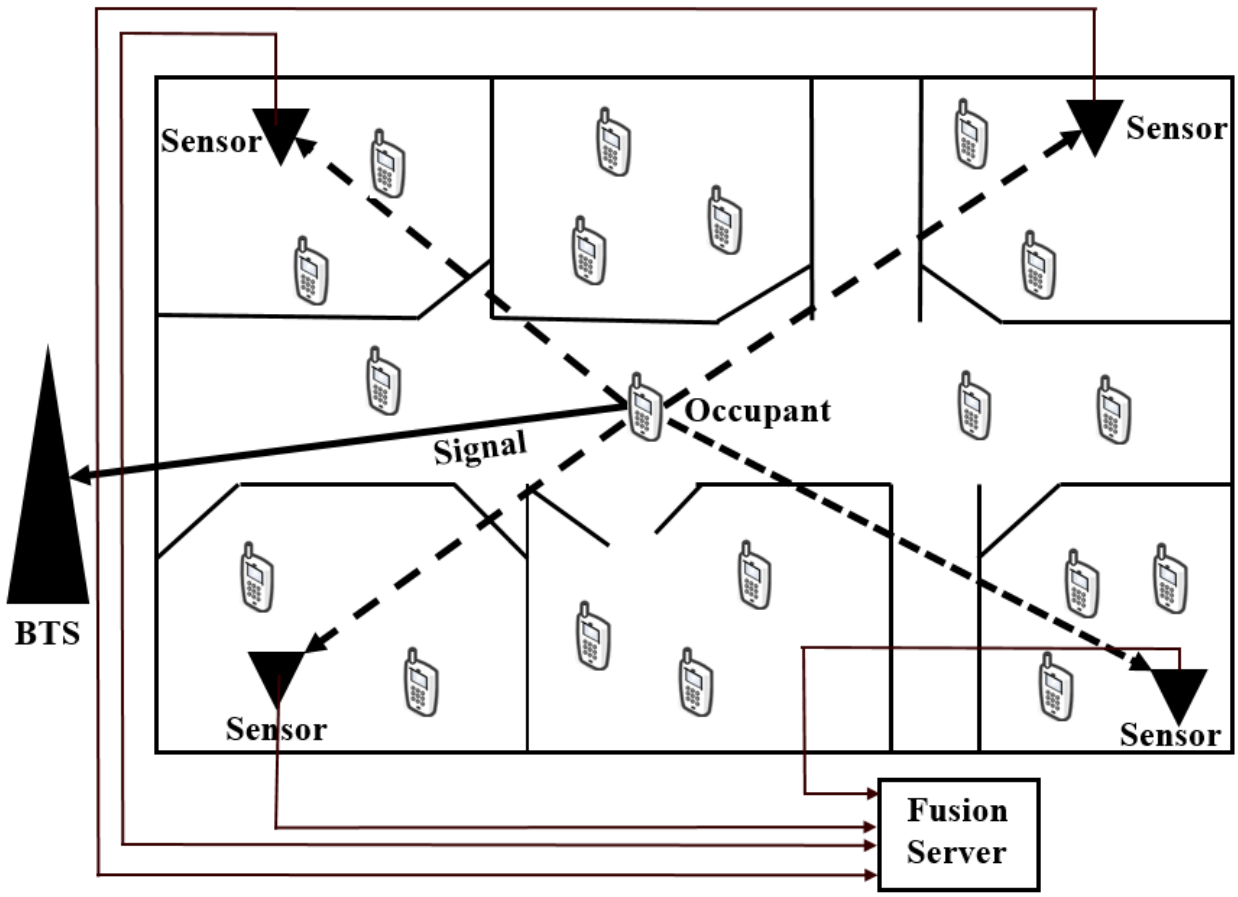


Figure 3.1: System Schematic

The remainder of this section includes details on hardware and software used in our testbed platform.

3.2 Hardware

For this project, an Ettus Research USRP (Universal Software Radio Peripheral) B200 software defined radio board was chosen as a BTS as well as a receiver (sensor). This model has RF coverage from 70MHz to 6GHz, therefore covering all cellular bands. It can handle an instantaneous bandwidth of 56MHz, which is adequate for most cellular bands (but not all). It is powered over USB and transfers data to an associated compact and efficient host computer using

the USB 3.0 standard. This board also contains a Xilinx Spartan 6 XC6SLX75 FPGA, which can be programmed to perform signal processing on the received signal[70]. The USRP B200 is equipped with an Ettus VERT900 antenna, designed for use in 824-960MHz and 1710-1990MHz frequencies, thereby covering the majority of cellular (GSM) bands. A software defined radio (SDR) was chosen due to the ability to reprogram the device; a production system would be lower cost and designed specifically for this sensing application.

Figure 3.2: Intel NUC (Left) and Ettus VERT900 antenna connected with Ettus B200 (Right)

In addition to the receiver, the sensor must process the received signal, record the power, time and frequency of this signal, and send this information to a fusion server. While much of this will eventually be handled by an FPGA, a host computer connected to the USRP B200 currently handles the signal processing. The Intel NUC D54250WYKH was chosen for its small size, low power usage, processing power, and USB 3.0 capability. Figure 3.2 shows the VERT900 antenna

connected with USRP B200 paired with Intel NUC. This computer has a core i5 dual-core processor with turbo capability to achieve 2.6 GHz, 8GHz RAM, a 128GB SSD, and a WiFi adapter. The computer runs Ubuntu Linux, and controls the USRP B200 and processes the incoming signal using MATLAB. However, GNU radio can be a cheap alternative for such kind of application. The detected cellular signal power, time, and channel information were sent over an SSH/TCP/IP connection to the fusion server. From the server side, the collected data was analyzed using Python and stored in a database.

For testing purpose, our mobile network was modeled with the help of Open Source GSM Infrastructure (OpenBTS) implemented with a USRP B200 board which is equipped with two VERT900 antennas, one for transmitting GSM downlink frequencies and another for receiving GSM uplink frequencies, and customized Super SIM. OpenBTS is a C++ application that implements the GSM stack. The combination of OpenBTS and software-defined radios change the way of thinking about mobile networks and allowing the construction of complex radio networks purely in software.

3.3 Software

The important step in determining the location of the cell phones is the detection of the uplink signal transmitted to a base transceiver station (BTS) and process the detected signal accordingly to perform localization using RSS, the channel of reception, detection time and sensor information. For this project, MATLAB is used to process the detected signal and send the processed signal to the fusion server over a TCP connection.

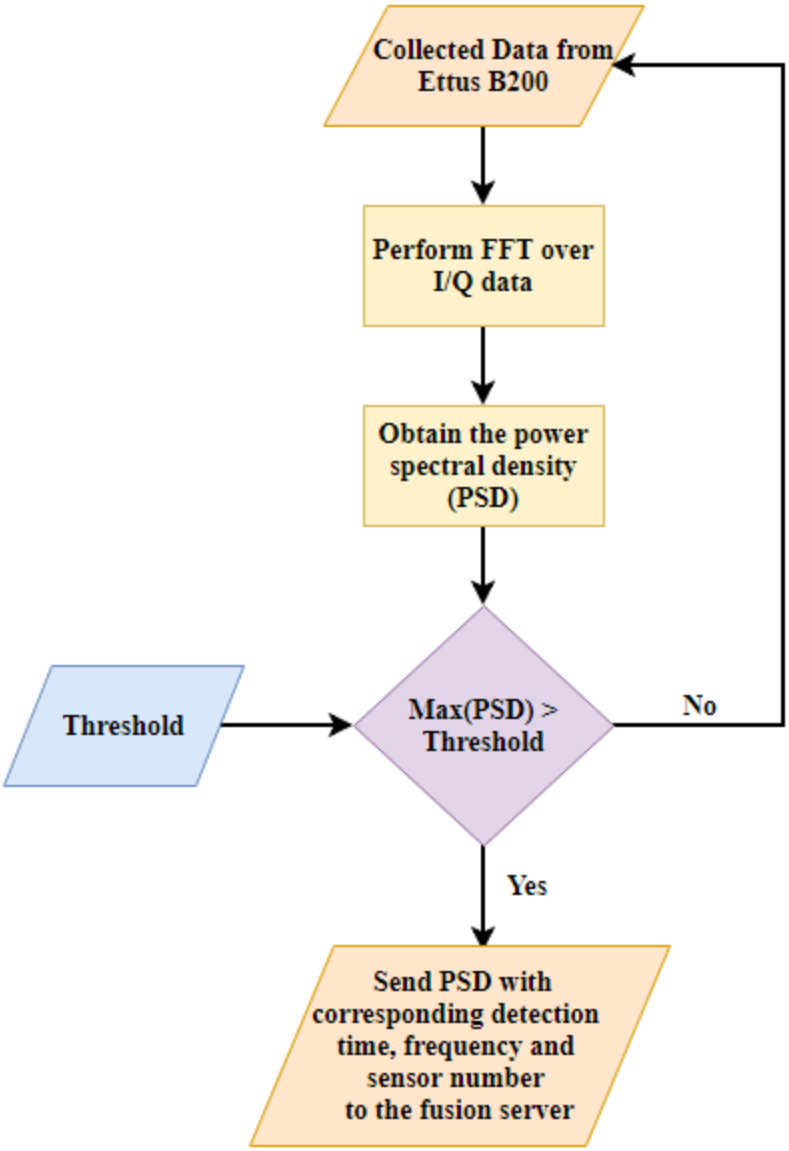


Figure 3.3: Signal processing technique flowchart performed by each sensor

In Figure 3.3 the main processing steps of RSS detection and extraction of detected information is illustrated. The whole processing is performed in MATLAB from the sensor side. Initially, the sampled I/Q data comes from USRP B200 board. Then Fast Fourier Transform (FFT) is performed over the sampled data, from which power spectral density (PSD) is calculated. After that, the maximum PSD is compared with a threshold, environmental noise level, to decide the

presence of RSS. Finally, the PSD and corresponding channel of reception, detection time and sensor information were sent to the fusion server over an SSH/TCP/IP connection.

The fusion server was received the detected cellular signal information from each sensor. The data was received over a TCP/IP connection as a string, which is then parsed and stored in a database. Python is used to analyze the processed data and stored in a database for localization. In the server, the entries were matched with near-identical time and channel stamps by depending on the time and channel stamps of each entry. At least three entries with near-identical time and channel stamps are required to match, to feed the received power and sensor information into the MMSE based indoor localization algorithm. MATLAB is used to investigate the MSE performance of the localization. Thus a location estimation can be made. As each cellular device may communicate with a cellular BTS periodically, duplicate occupant entries may occur; this must be accounted for using a statistical model. Figure 3.4 illustrates the databases and functions of the server as a schematic.

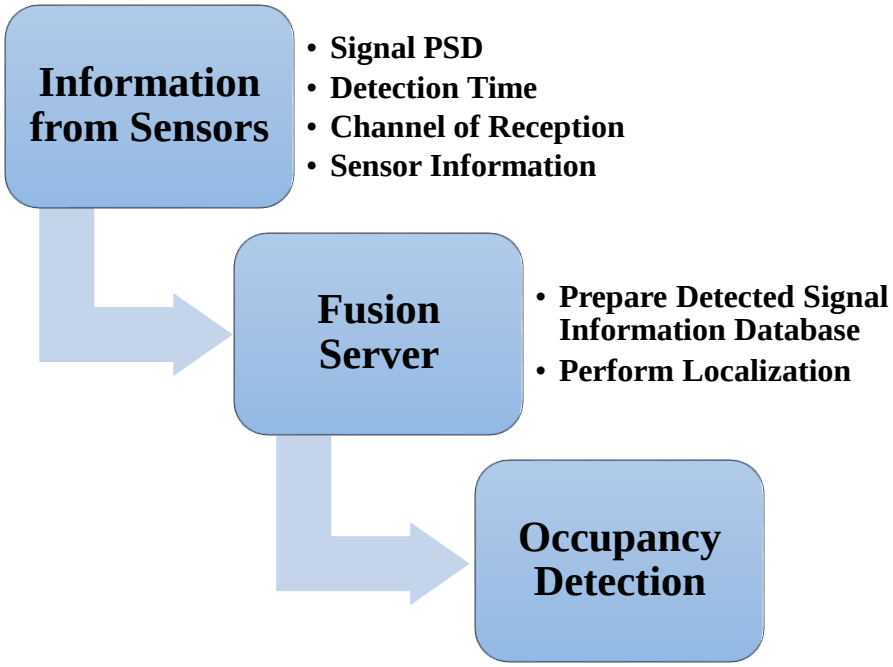


Figure 3.4: Fusion Server Schematic

The following section demonstrates each step of the software installation process to run the proposed prototype architecture.

3.4 OpenBTS Installation

This section explains the building and installing of software and dependencies followed to develop the proposed framework. In order to efficiently rebuild this system, it is important that each mentioned version of the software is present during the installation process.

Several prerequisites are required before the system will be capable of building, installing, and running OpenBTS. Due to the compatibility issue, special care is required to select the appropriate version of the operating system (OS) and Open-source toolchain. USRP Hardware Driver (UHD) is fully supported on Linux, using the GNU Compiler Collection (GCC) and should work on most major Linux distributions.

Although OpenBTS implements most of the complexity involved in building a mobile network in software, radio waves must still be transmitted and received somehow. Below are the hardware/software components required to procure implementing this capability in a development setting.

- Linux Desktop/Server
- Software Defined Radio
- Antennas
- Test Phones
- Test SIMs (Subscriber Identification Module)
- Smart Card Writer

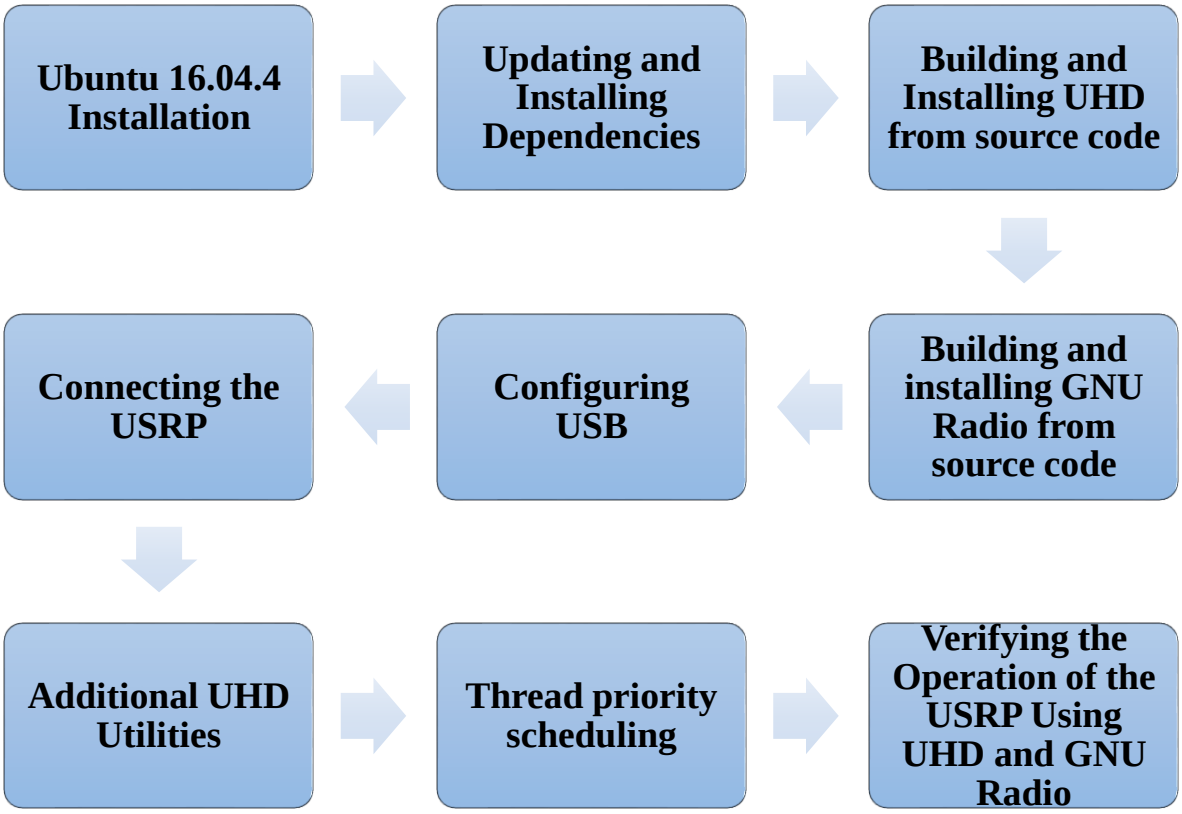


Figure 3.5: Prerequisites required for building, installing, and running OpenBTS

Figure 3.5 shows the steps required to follow to make the system prepare for building, installing, and running OpenBTS. Furthermore, ***Appendix A*** presents a step by step in detail procedure for installing the required prerequisite software and applications to develop the OpenBTS system. Ubuntu Desktop 16.04.4 LTS is required as the operating system. After installing the dependencies, the UHD and GNU radio should install from source code. Finally, it is required to verify the operation of USRP using UHD and GNU Radio. The following section describes the in detail installation process of OpenBTS.

3.4.1 Building, Installing and Running OpenBTS

The installation of a single instance of OpenBTS on a single computer with a single radio is described here. A complete installation of OpenBTS comprises the following components:

- **OpenBTS itself:** This is the GSM implementation from the Time Division Multiple Access (TDMA) part of Layer 1 up through Layer 3 and the Layer3/Layer 4 boundary.
- **Transceiver:** This is the software radio modem, implementing the lower part of Layer 1. OpenBTS starts the transceiver automatically.
- **A SIP (Session Initiation Protocol) PBX (Private Branch Exchange) or softswitch (Asterisk):** This component connects speech calls. This is not packaged with OpenBTS.
- **Sipauthserver:** This is the SIP registration and authorization server, used to process location updating requests from OpenBTS and perform corresponding updates in the subscriber registry database.

- **Smqueue:** This is the store-and-forward text messaging server. It needs to be started independently of OpenBTS. Smqueue is not required in installations that do not support text messaging.

In Figure 3.6 black links are the network connections (SIP), red links are the file system connections and the blue link is the Open Database Connectivity (ODBC) (network/local DB lookups) [71].

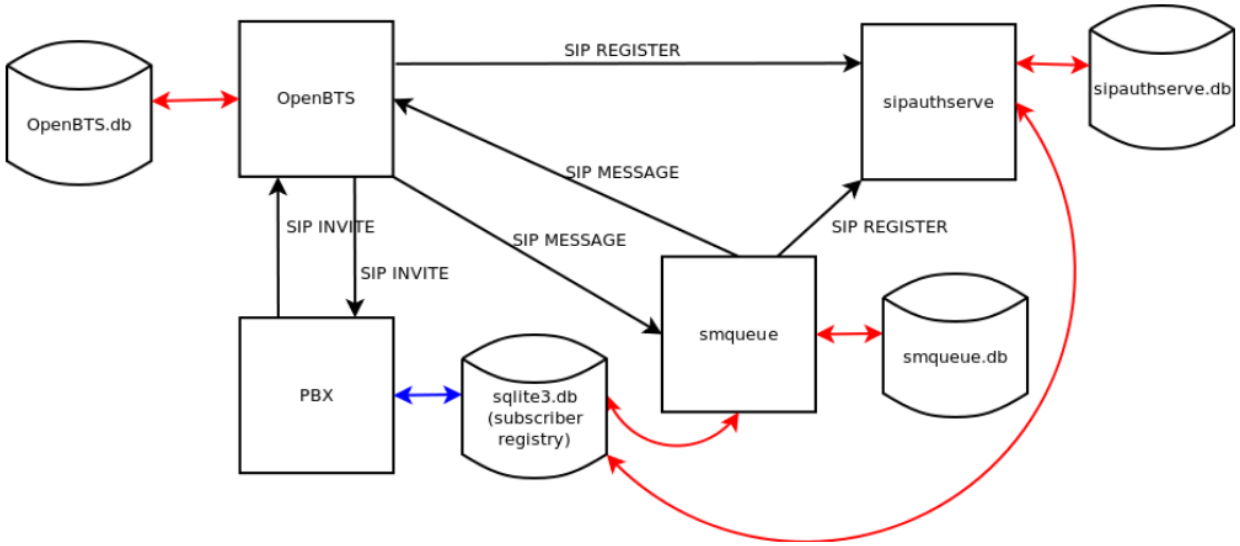


Figure 3.6: OpenBTS system connections

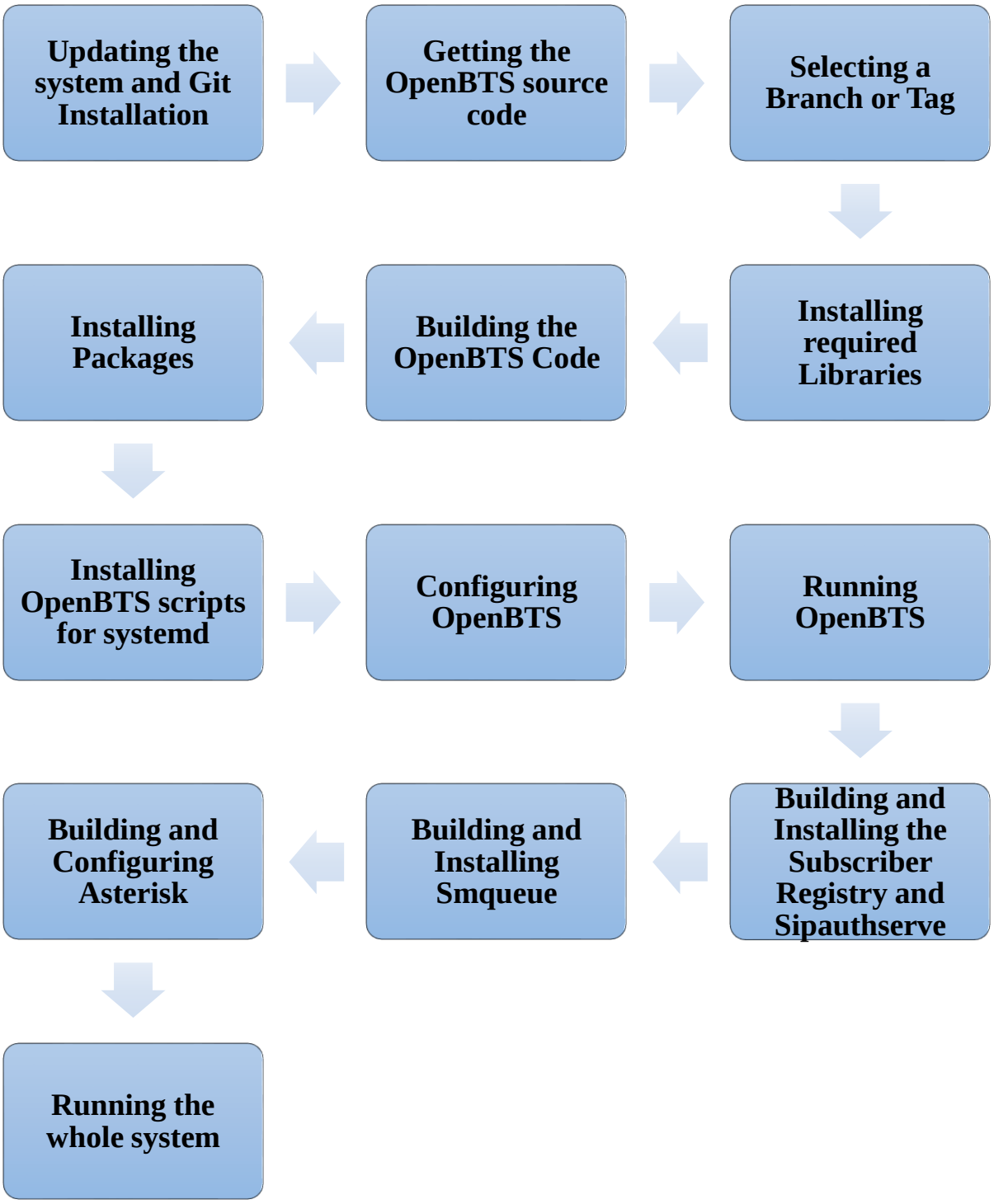


Figure 3.7: Required steps for building, installing, and running OpenBTS

Figure 3.7 shows the steps required to follow for building, installing, and running OpenBTS. Furthermore, ***Appendix B*** presents a step by step in detail procedure for installing the software and applications to develop the OpenBTS system. Before installing the Git, the system needs to be updated. After that, the source code of OpenBTS is downloaded. Before building OpenBTS, the desired branch or tag for compilation need to be chosen. After installing the required libraries, the OpenBTS should be ready to install from source code. Finally, it is required to build and install/ configure the Sipauthserve, Smqueue, and Asterisk.

3.4.2 Testing Radio Frequency Environment Factors

This section will guide to verify the uplink. In a GSM network, two separate radio frequencies are used, so that the BTS and handsets can communicate simultaneously in both directions. GSM uses Frequency Division Multiple Access (FDMA) to establish full duplex communication. The selected ARFCN determines which pair of frequencies will be used. The path from the BTS to the handset is known as the downlink and the path from the handset back to the BTS is known as the uplink.

To get the best performance it is required to ensure the cell has good uplink/downlink (UL/DL) balance. More specifically, it should not be UL- or DL-limited. OpenBTS has several parameters for this purpose and that should be tuned for specific hardware.

After installation of OpenBTS following the steps mentioned in ***Appendix B***, the next thing to look out for when setting up a new network is excess radio interference or *noise* from other sources on the uplink. If the uplink is too noisy, the signals from handsets cannot reliably be demodulated into usable information. OpenBTS shows the current level of noise by using the `noise` command. The following procedure requires a working OpenBTS setup with at least one

handset connected. Run the *noise* command in the OpenBTSCLI several times to get the worst value (largest) for the noise Received Signal Strength Indication (RSSI) of the setup. It should be a negative value.

```
OpenBTS> noise
noise RSSI is -79 dB wrt full scale
MS RSSI target is -50 dB wrt full scale
INFO: the current noise level is acceptable.
```

In this example, the detected environmental noise RSSI is –79 dB (lower numbers are better and mean less noise is present) and the configured target RSSI level for handsets is –50 dB. This means that the BTS can, at best, receive 29 dB more energy from the handsets than the environmental noise. A very good margin meaning uplink reception issues due to noise should not be a problem.

Smaller margins between these two numbers will produce different informational messages. For example, having a margin of 10 dB or less will report:

```
WARNING: the current noise level is approaching the MS RSSI target, uplink connectivity will be
extremely limited.
```

A margin of zero or less will report:

```
WARNING: the current noise level exceeds the MS RSSI target, uplink connectivity will be
impossible.
```

If either of these warning messages is reported, necessary action will be needed to take by reducing uplink noise and/or increasing the handset transmit power. Furthermore, if the handset

can detect the downlink signal but can no longer connect then noise should be the first thing to check.

To get the required RSSI level which means BTS can receive a cell phone signal without errors, 6-8 dB has to be added to the noise RSSI value. This is called target RSSI in OpenBTS and is set with *GSM.Radio.RSSITarget* value in the *config*. If noise RSSI is -79dB, then set the target RSSI to -50dB:

```
OpenBTS> config GSM.Radio.RSSITarget -50
```

3.4.2.1 Reducing Noise

If the base station radio setup does not include a frequency duplexer, the number one source of noise on the uplink can actually be the downlink signal. Without proper duplexing to filter it out, the downlink signal is usually the closest energy source to the uplink both physically and by frequency. Even without a duplexer, there are ways to reduce noise on the uplink.

3.4.2.1.1 Antenna alignment

A quick duplexer of sorts is simply aligning the antennas so that they do not readily feed into each other. If rubber duck style antennas are used, tilt them to form a 90-degree angle. Thus the radiation pattern for these antennas will then be perpendicular as shown in Figure 3.8.

If the antennas are parallel to each other, the signal can efficiently flow from the transmit antenna to the receive antenna, but when the antennas form a 90-degree angle, the signal is being transmitted on a different plane than it is being received on. The change of noise level can be

observed by running the *noise* command before and after the adjustment. This simple adjustment can reduce noise by as much as 10 dB.

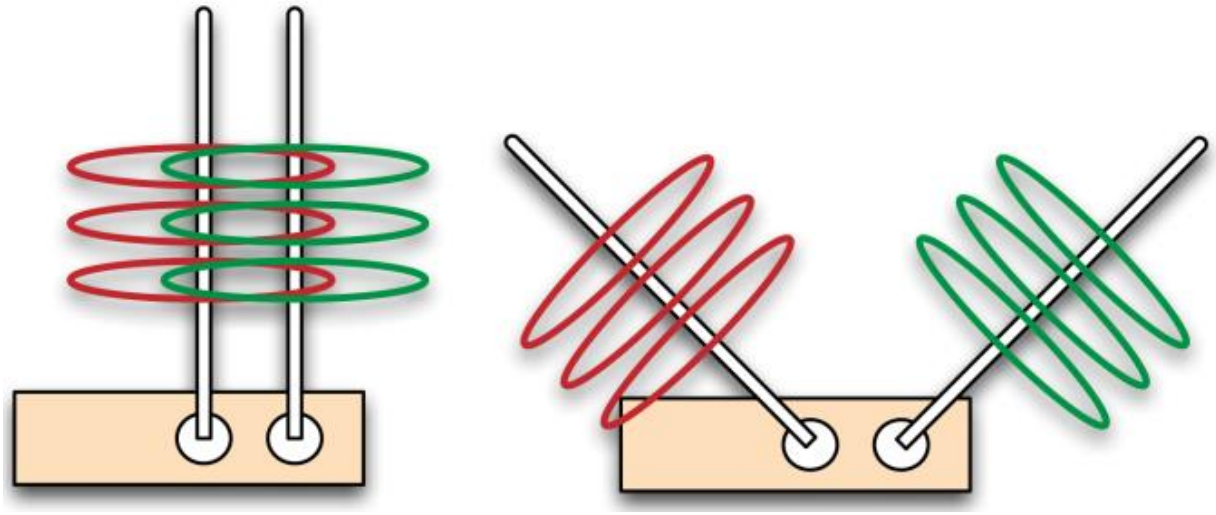


Figure 3.8: Antenna alignment

3.4.2.1.2 Downlink transmission power

The alignment step above reduced the flow of energy from the transmit antenna to the receive antenna. This received energy may still be too high for the uplink to be usable. Decreasing the downlink transmission power will further clean up the uplink. The coverage area lost by decreasing the downlink power is not significant in a lab environment. Cleaner signals are preferable to strong ones. To see the current level, run the *power* command with no arguments. The power is reported in decibels of attenuation:

```
OpenBTS> power
current downlink power -10 dB wrt full scale
```

To decrease the downlink transmission power, for example by 20 dB, enter the following:

```
OpenBTS> power 20
current downlink power -20 dB wrt full scale
```

The downlink is now transmitting with 20 dB less power. To observe the improvement, run the *noise* command. If anyone wants to limit the radius of the BTS it is better to do so by decreasing the BTS transmit power i.e. by increasing attenuation with the *power* command.

3.4.2.2 Boosting Handset Power

Handsets can also be told to use more power by adjusting the *GSM.Radio.RSSITarget* and *GSM.Radio.SNRTarget* keys. Although, the default values for these keys should be sufficient in most situations. However, if large fluctuations in received power are encountered, it may be necessary to increase these values to provide a larger buffer in allowable power differences. Boosting the power for all handsets will drain their batteries more rapidly but uplink signals will be more reliable. There are always trade-offs.

Finally, if the noise level is still too high then it is recommended to change the ARFCN to a less noisy one.

3.4.3 Making Connection

This section describes how to enter the identity parameters for the handset to connect to the network after both the downlink and uplink have been verified. To make sure all settings have been applied, it is required to restart OpenBTS.

The only step left before actually connecting to the test network is to find and enter the handset's identity parameters so it will be accepted onto the network.

3.4.3.1 Finding the IMSI

International Mobile Subscriber Identity (IMSI) is the main identity parameter to be searched for. This is a 14 - 15 digit number stored in the SIM card and is analogous to the handset's username on the network.

Handsets will not usually divulge the IMSI of their SIM card. It can sometimes be located in a menu or through a field test mode, but this method of determining a SIM's IMSI is very cumbersome to explain. Luckily, OpenBTS knows the IMSIs it has interacted with.

To force an interaction between a handset and the test network, a location update request (LUR) operation on the network will be performed, which is analogous to a registration. This is nothing more complicated than selecting the network from the carrier selection list.

Before attempting any LURs, the sipauthserve daemon which is responsible for processing these requests is needed to start (if not running).

Now, again following the steps in ***Searching for the Network*** section, bring up the carrier selection list and choose the test network. After a short time, the handset should report a registration failure.

It may also receive an SMS (Short Message Service) from the test network indicating that registration has failed. This message automatically includes the IMSI, thus skip to ***Adding a***

Subscriber section. However, this feature does not work on all hardware so continue on, if an SMS was not received.

OpenBTS remembers these LUR interactions in order to perform IMSI/Temporary Mobile Subscriber Identity (TMSI) exchanges. IMSI/TMSI exchanges swap the user-identifiable IMSI for a TMSI and are used to increase user privacy on the network. The exchanges are disabled by default (modify `Control.LUR.SendTMSIs` to enable); however, the information is still there to inspect using the `tmsis` command. To view all recent LUR interactions with handsets, run the `tmsis` command:

```
OpenBTS> tmsis
IMSI              TMSI  IMEI              AUTH  CREATED  ACCESSED  TMSI_ASSIGNED
901990000000018   -     012546629231850   0     78s      78s       0
001010000000002   -     351771054186520   1     80h      95s       0
001010000000003   -     351771053005400   1     80h      108s      0
```

Entries are sorted by time, with the top entries corresponding to the most recent interactions. The recent handset should be the top entry on this list. The most recent interaction with `AUTH` set to 0 because the LUR failed due to the handset not being a known subscriber. The other entries in this example are additional test handsets that have successfully performed a LUR as indicated by the `AUTH` column being set to 1.

3.4.3.2 Finding the IMEI

In a busy environment, it can be difficult to ascertain which handset hardware corresponds to which entry on this list. To match an IMSI to a specific piece of hardware the International

Mobile Equipment Identifier (IMEI) can be used. It is the unique identifier given to the handset's physical radio hardware, analogous to a MAC (Media Access Control) address on an Ethernet interface.

A handset's IMSI is usually printed under its battery cover or somewhere very near the SIM itself. On many handsets, the IMEI can also be accessed by dialing `*#06#` on the keypad.

The IMEI value is typically only used for reporting and detecting stolen hardware in production environments. Here it serves as a convenient way to determine which SIM is in which handset. The final digit of the IMEI may not match what OpenBTS displays. It is a check digit and is shown as a zero in OpenBTS.

3.4.3.3 Adding a Subscriber

To create a new subscriber account on the test network, a couple of fields are still needed but are freely selectable. These are Name and Mobile Station International Subscriber Directory Number (MSISDN). The Name field is merely a name for this subscriber to remember which handset or which person it is associated with. The MSISDN field is nothing more complicated than the subscriber's phone number. This can be any number as long as not connected to the public telephone network.

The program for adding subscribers is `nmcli.py`. It is a simple client for the NodeManager APIs and allows to change configuration parameters such as add subscribers, monitor activity etc.

`nmcli.py` is already present in the development directory. To access the file, required to move to `dev/NodeManager` directory:

There are two ways to add a subscriber using `nmcli.py`. The first creates a subscriber that will use cached authentication:

```
$ ./nmcli.py sipauthserve subscribers create name imsi msisdn
```

The second creates a subscriber that will use full authentication:

```
$ ./nmcli.py sipauthserve subscribers create name imsi msisdn ki
```

If a spare SIM is used by another provider, in that case, it is not possible to access the secret key, `Ki` which is stored in the SIM so use the first invocation style. If a programmed SIM is used, `Ki` will be known, in that case, use the second invocation style of `nmcli.py`.

In this example, a programmed SIM is used in a “BLU” cell phone and assigned `0000001` number.

```
$ ./nmcli.py sipauthserve subscribers create "BLU" IMSI901990000000018 \ 0000001
raw request: {"command":"subscribers","action":"create","fields":
"name":"BLU","imsi":"IMSI901990000000018","msisdn":"0000001","ki":""}}
raw response: {
"code" : 200,
"data" : "both ok"
}
```

3.4.3.4 Connecting

Now when the test network is selected in the connection menu, the LUR should succeed. This can be confirmed with the `tmsis` command in OpenBTS. The `AUTH` column will now have a "1" in the entry corresponding to the IMSI. This verifies the successful registration to the own private mobile network.

```
OpenBTS> tmsis
IMSI              TMSI  IMEI              AUTH  CREATED  ACCESSED  TMSI_ASSIGNED
901990000000018   -     012546629231850   1     11m      56s       0
001010000000002   -     351771054186520   1     80h      8m        0
001010000000003   -     351771053005400   1     80h      9m        0
```

At this stage, the handset has access to the network, it is ready to perform some tests. ***Appendix C*** presents a step by step in detail procedure for testing of messaging and calling capability of the network.

3.4.3.5 Measuring Link Quality

The `chans` command is a handy tool available on the OpenBTS CLI. This tool can be used to objectively quantify link quality instead of basing it on user perception. An active call is needed to see anything useful with this command. The active channel is used to estimate link quality between the mobile and the BTS and to tune receiver (Rx) gain and transmitter (Tx) attenuation.

In this example, a call was placed to the *2602* test tone extension and after 33 seconds (indicated in the "Time" column) the `chans` command was executed. The handset was moved approximately 20 meters farther away from the radio being used and another sample with the

chans command was taken. There are a lot of fields and all of these are very useful. For now, the focused fields are signal-to-noise ratio (*SNR*), *TXPWR*, *RXLEV_DL*, and *FER*.

```
OpenBTS> chans
CN  TN   chan    transaction  Signal  SNR   FER    TA    TXPWR  RXLEV_DL  BER_DL  Time   IMSI
         type    id           dB            pct    sym   dBm    dBm       pct
0   1    TCH/F   T103         60      65.1  0.00   -0.9  5      -48       0.00    0:33   4600...
                                          ^^^^^^^
                                       Good UL quality
OpenBTS> chans
CN  TN   chan    transaction  Signal  SNR   FER    TA    TXPWR  RXLEV_DL  BER_DL  Time   IMSI
         type    id           dB            pct    sym   dBm    dBm       pct
0   1    TCH/F   T103         22      35.4  13.00  -0.9  9      -87       0.00    4:32   4600...
                                          ^^^^^^^
                                       Bad UL quality
```

In both readings, there is a single active channel. The *SNR* column represents the SNR of the uplink as measured by the BTS; higher is better. As the handset is moved away, this number upgrades. *TXPWR* column represents the uplink transmit power that the handset reported. In the second reading, this number has jumped from 5 to 9 dBm, meaning the handset used more power to transmit the signal to the BTS. This would explain why there was a change in *SNR* measured at the BTS.

The network independently instructs the handsets to transmit with different power levels depending on how well the BTS receives their uplink signal. This is so all signals are received at the BTS with about the same strength, making it easier to demodulate.

BTS, however, use the same transmission power on the downlink for all handsets. This can be observed in the *RXLEV_DL* column. This column represents the downlink signal level that the

handset reported. In the second reading, this number has gone down from –48 dBm to –87 dBm as the handset moves farther away from the BTS. It is receiving the downlink signal with less strength because it is now farther away.

The power control is implemented for both the uplink direction (the power used by the MS for transmission) and the downlink direction (the power used by the BTS for transmission). For the uplink, the Base Station Controller (BSC) calculates the power to be used by the MS and sends this to the BTS. The BTS then sends this information to the MS in the Slow Associated Control Channel (SACCH) header. Two key parameters are signaled to the MS in the SACCH messages, namely "ordered MS power level" and "ordered timing advance". For the downlink, the BSC calculates the power to be used by the BTS and sends it to the latter. The BTS then uses this power value for transmission to the MS[72].

The uplink and downlink power control applies for each dedicated channel and depends upon the power requirements based on the distance between the BTS and the MS and the air interface conditions. Further, as an implementation option, the BSC can send power-related parameters to the BTS whereby the latter can autonomously implement the uplink and downlink power control logic locally.

rxgain adjustment is required to provide good reception at the edge of the BTS reception area, while at the same time avoiding saturation. The easiest way to estimate reception quality is to use *FER* value at the *chans* CLI command output. Zero *FER* means good reception, non-zero *FER* means errors on uplink channel[73]. A complete listing of these fields can be retrieved by running *help chans* command.

In this stage, a full working framework has established that can able to perform messaging and calling.

3.5 Sensor Configuration

For this project, MATLAB is used in every sensor to process the detected signal (I/Q data from USRP B200) from the cell phone and send the processed signal to fusion server over a TCP/IP connection. It is possible to design and prototype software-defined radio (SDR) systems using USRP with MATLAB. With this support package from Communications Toolbox and a USRP radio, practical SDR systems can be designed and verified. MATLAB Support Package for USRP Radio includes:

- Use of USRP as a standalone peripheral for live RF data I/O, including functions and system objects for connecting MATLAB to UHD-based USRP radios.

Before installing MATLAB, it is required to install the prerequisites to make the system prepare. Figure 3.5 shows the steps required to follow to install the prerequisites. Furthermore, ***Appendix A*** presents a step by step in detail procedure for installing the required prerequisite software and applications to develop the system. Finally, ***Appendix D*** presents a step by step procedure that is required to install the Communications Toolbox Support Package for USRP Radio in MATLAB

3.6 Network Time synchronization

The computer clocks in servers, workstations and network devices are not inherently accurate. Most of these clocks are set by hand to within a minute or two of actual time and are rarely checked after that. Many of these clocks are maintained by a battery-backed, clock-calendar

device that may drift as much as a second per day. Having any sort of meaningful time synchronization is impossible if such clocks are allowed to run on their own.

3.6.1 The Importance of Time Synchronization for the Network

In modern computer networks, time synchronization is critical because every aspect of managing, securing, planning, and debugging a network involves determining when events happen. Time also provides the only frame of reference between all devices on the system network. Without synchronized time, accurately correlating log files between these devices is difficult, even impossible. Following are just a few specific reasons:

- Tracking security breaches, network usage, or problems affecting a large number of components can be nearly impossible if timestamps in logs are inaccurate. Time is often the critical factor that allows an event on one network node to be mapped to a corresponding event on another.
- To reduce confusion in shared file systems, it is important for the modification times to be consistent, regardless of what machine the file systems are on.
- Billing services and similar applications must know the time accurately.
- Some financial services require highly accurate timekeeping by law[74].

For this thesis, time synchronization is essential. OpenBTS and all of the sensors are required to be time synchronized. The signal from the cell phone is required to sort based on the transmitting time and the detection of the cell phone signal by the sensors is also required to sort based on the receiving time. Finally, based on the matched timestamp, the location of the cell phone is estimated from the transmitted and received signal power. ***Appendix E*** presents a step by step procedure that

is required to install `htpdate` to syncs time over http protocol and also explains the superiority of `htpdate` over other time-setting software.

CHAPTER 4

EMPIRICAL RESULTS

In this chapter, the performance of the indoor occupancy detection is presented in a real-world environment. The measurement campaign was conducted in a complex scenario with 6 sensors placed in a non-uniform, polygonal room sizes on two different places. The first experiment was conducted on the second floor of the south wing of Kingsbury Hall at the University of New Hampshire. The second one was conducted on an office (Simpson Gumpertz & Heger (SGH)) located in Waltham, MA 02453. In this scenario, sensors were placed in locations as close to exterior walls of the building and corners of the room as possible. For mobile communication, transmit power from the cell phone is not a constant rather it is a function of a distance between the BTS and the location of the cell phone. By considering this phenomenon, the BTS was placed in such a way that the cellphone can transmit as much constant power as possible. All measurements and programming were done with the following devices:

- **BTS:** Ettus Research USRP B200 board equipped with two Ettus VERT900 antennas and Dell Optiplex 9010 computer that has a core i7 dual-core processor, 8GHz RAM, a 1TB SSD, a Tp-link (Tl-wn881nd) wireless adapter and the Operating system (OS) is Ubuntu Linux.
- **Sensors:** Ettus Research USRP B200 equipped with an Ettus VERT900 antenna and Intel NUC D54250WYKH minicomputer that has a core i5 dual-core processor with turbo capability to achieve 2.6 GHz, 8GHz RAM, a 128GB SSD, a WiFi adapter, USB 3.0 capability, and the OS is Ubuntu Linux.

- **Server:** Dell Optiplex 9010 computer that has a core i7 dual-core processor, 8GHz RAM, a 1TB SSD, and the OS is Microsoft Windows.

4.1 Kingsbury Measurement Campaign

In Kingsbury, the sensors were placed on the ends of ceiling-mounted overhead service carriers in three laboratories (S210, S211, S215), and on the top of a cabinet in other two laboratories (S216, S220). Figure 4.1 shows a map of the measurement region illustrating sensor positions, room boundaries, and measurement points. The coverage radii of the BTS was around 30.48 meters. The location of the sensors and the measurement points were selected based on the coverage area of the BTS.

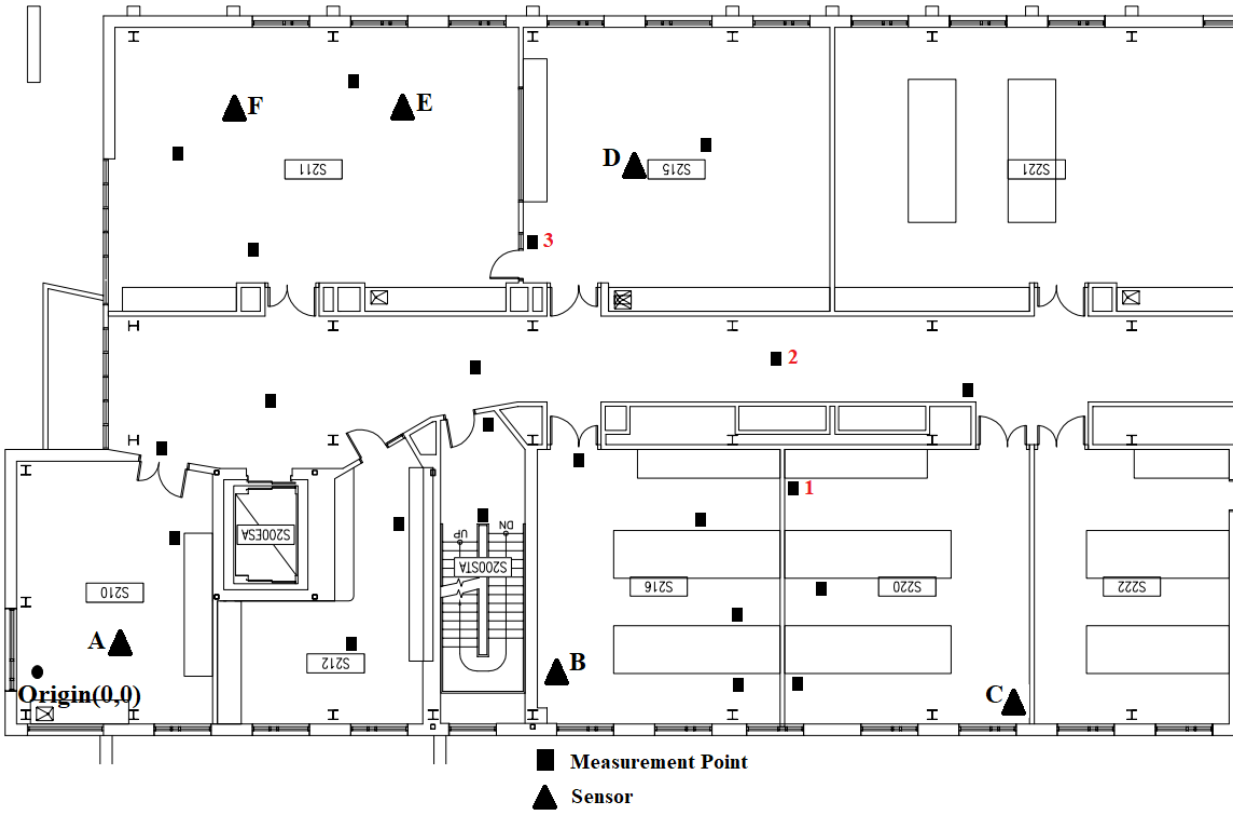


Figure 4.1: Map of Kingsbury Hall, South Wing, Second Floor, including sensors and measurement points

In the measurement area, there were 22 unknown nodes (measurement points) indicated by black rectangles to be examined for six reference nodes (sensors marked by A – F) are also shown in Figure 4.1. The coordinates of the sensors and other parameters are shown in Table 4.1.

Table 4.1: Simulation Parameters

Parameters	Value
Number of Sensors	6
Coordinates of the sensors (m)	A (3.66, 1.28) B (14.6, -0.35) C (28.96, -1.22) D (17.65, 14.51) E (12.8, 16.15) F (6.4, 16.15)
Number of unknown nodes	22
Path loss exponent, *n*	5.8
The estimated distances (m)	5.98 – 28.58

The RSS power measurements were taken simultaneously by all six sensors and processed by a central controlling computer (Server) which performed the location technique on the collected data. A BLU Z150 1.8" GSM quad band (850/900/1800/1900) 2G compatible unlocked cell phone was used as a transmitter. The RSS measurements occurred at the sensors with the transmitted signal from the cell phone located at each of the 22 measurement points indicated in Figure 4.1. The cell phone was placed at each measurement point, approximately 1.13 m off the ground to minimize the effect of multipath and variability in the environment. For each measurement point,

a test call was made so that all 6 sensors can detect the RSS. After detection, the detected signal was processed by each of the 6 sensors simultaneously and sent the detection information to the server.

In the server side, based on the signal detection time, the received signals (RSS) were sorted for each of the 6 sensors. Also, the transmit signals of the cell phone were sorted based on the calling time. After that, the estimated distances were calculated using the modified version of Equation (2.6). Equation (2.6) can be rewritten as follows:

$$\frac{P_r(d)}{P_t} = \left(\frac{G_t G_r \lambda^2}{(4\pi)^2 L}\right) \times \frac{1}{d^n} = \frac{C}{d^n} \tag{4.1}$$

Where $P_r(d)$ is received power, and P_t is the transmitted power from Mobile Station (MS). G_t is the MS antenna gain and G_r is the wireless sensor antenna gain. Again, λ is the wavelength, d is the distance between the MS and the sensor, n is the path loss exponent and L is the shadow fading.

For reference distance, $d = d_0$ the equation (4.1) changes to as follows,

$$\frac{P_{r_0}}{P_{t_0}} = \frac{C}{{d_0}^n} = k \tag{4.2}$$

Now, For an arbitrary distance, $d = d$ the equation (4.1) can be rewritten as

$$\frac{P_r}{P_t} = \frac{C}{d^n} = k.\frac{{d_0}^n}{d^n} = k \times \left(\frac{d_0}{d}\right)^n \tag{4.3}$$

After obtaining the power of RSS, P_r in the unknown node from the sensor and based on the MS transmit power, P_t the estimated distance can be calculated by as follows:

$$d = d_0 \times \left(\frac{kP_t}{P_r}\right)^{1/n} \tag{4.4}$$

Furthermore, the estimated coordinates were calculated using equation (2.14). Finally, the performance of localization was investigated by MMSE using equation (2.5) at each of the measurement locations.

These measurement points were chosen to be relatively uniform over the area. All of these measurements were fed into the script for processing by the location technique. The value of σ is inherent in the environment, so it is constant. A value of σ for the building was not determined in this measurement campaign thus data from the measurement campaign cannot be plotted versus σ.

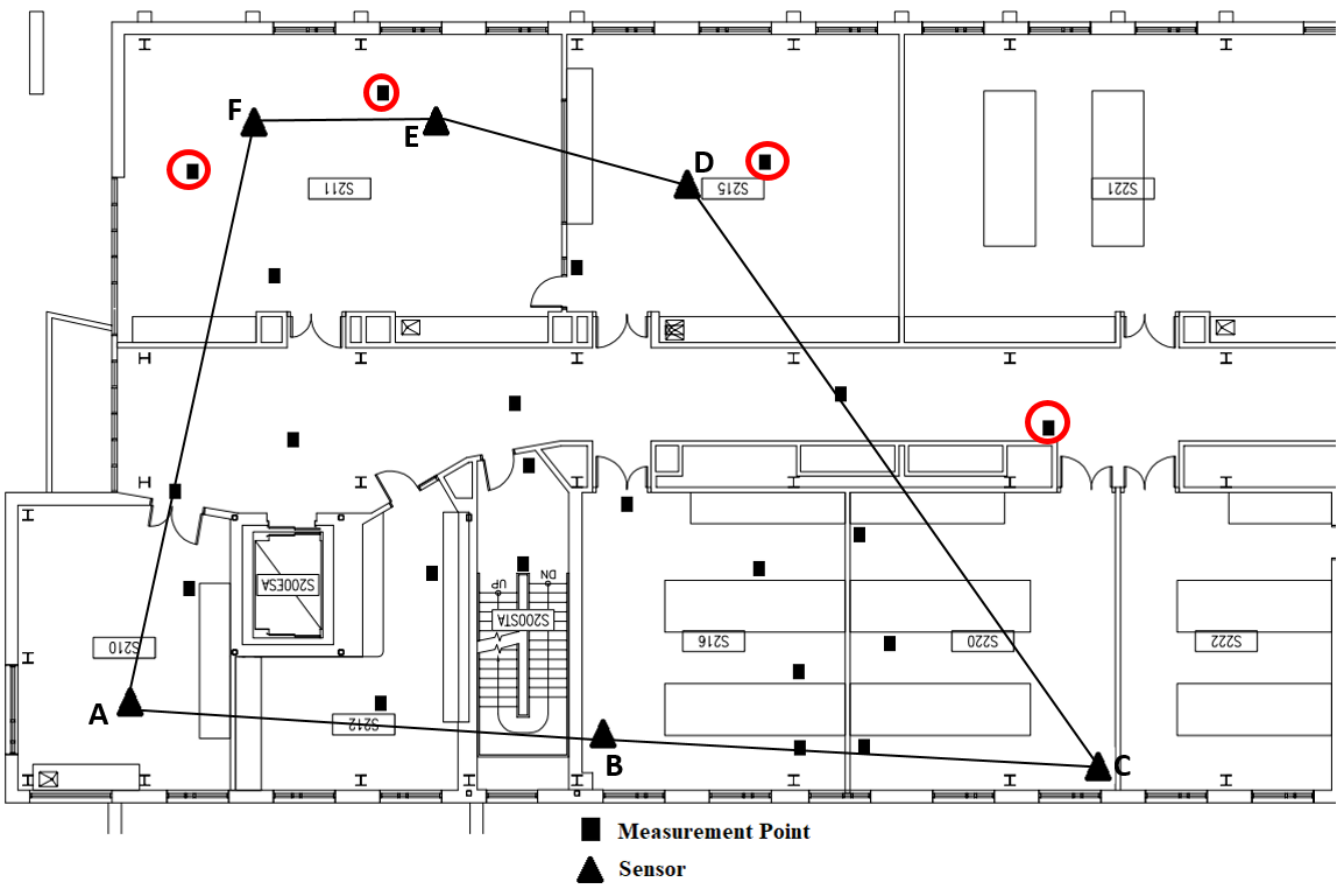


Figure 4.2: Map of Kingsbury Hall, South Wing, Second Floor, including measurement points bounded by 6 sensors

Observation 1:

There is a significant effect of the area lying enclosed by the sensors. The minimum error region is always the centroid of the polygon formation formed by connecting the sensors. As a result, the measurement points, those were inside the bounded region by the 6 sensors shown in Figure 4.2, had experienced less error compared with the points those were outside of the bounded region. Figure 4.3 shows the MMSE corresponding to the distance of the unknown node from the origin versus measured distances of the measurement points from the Kingsbury measurement campaign for both cases.

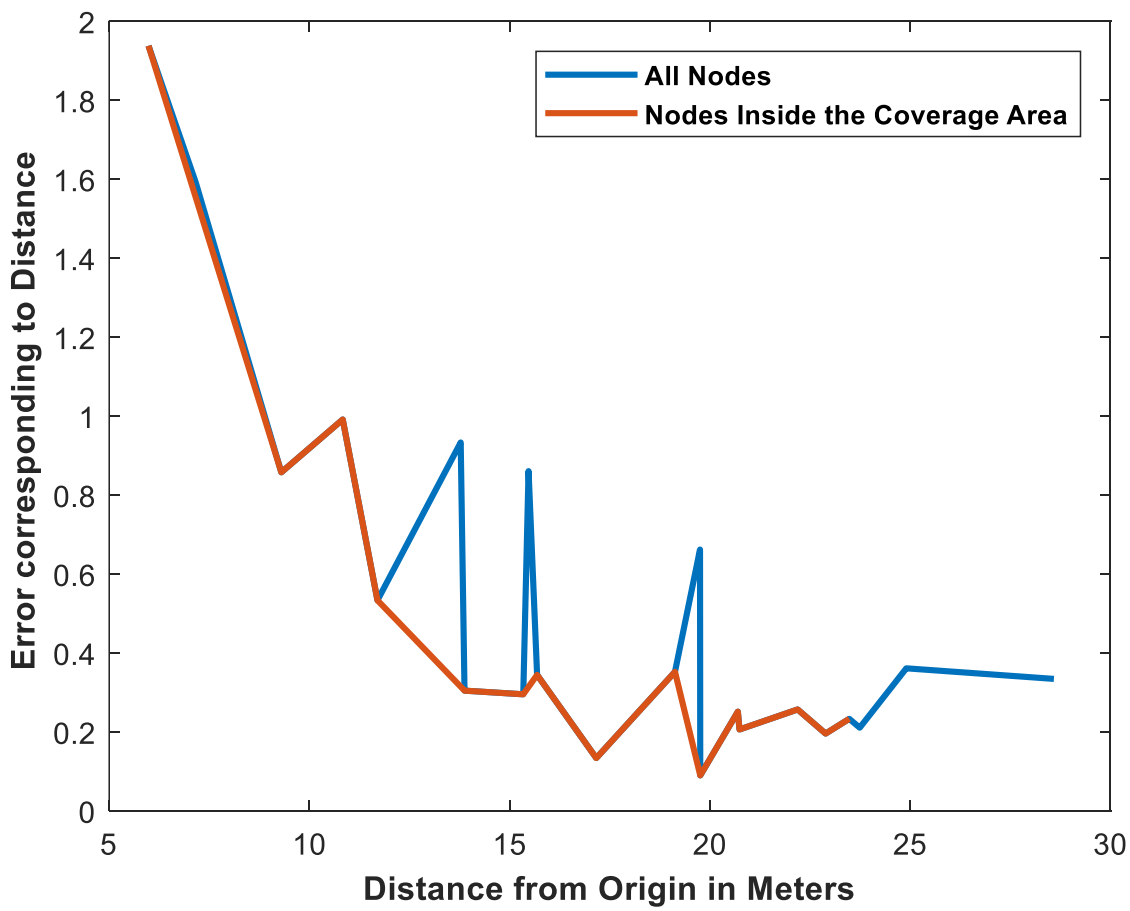


Figure 4.3: The localization of MSE corresponding to distance for different positions of the unknown node from the origin with 6 sensors

The red lines are the MMSE corresponding to the distance of the unknown node from the origin versus measured distances for the points (unknown nodes) those were inside of the bounded region and the blue lines are the MMSE corresponding to the distance of the unknown node from the origin versus measured distances for the points those were both inside and outside of the bounded region. The unknown nodes those were clearly outside of the bounded region by 6 sensors are yielding the greater error (abrupt spikes).

Observation 2:

Furthermore, Figure 4.3 shows that MMSE produces the greatest error when the unknown node is closer to the origin (indicated by a black circle in Figure 4.1) and gradually performs better for farther distances from the origin. This is obvious that for closer distances, the error in determining the actual location will be dominating compared with the farther distances. Although these errors are relatively unexpected and would not be too high like Figure 4.3. This could be due to a slight bias of measurement points away from the walls of rooms (due to a significant number of desks and bookshelves against walls, preventing measurement points against walls in many rooms) and the randomness in the environment, shadowing, and path loss exponent.

Observation 3:

It is already mentioned that the number of sensors impacts the performance of the localization method. To understand the effect of the number of sensors on localization performance, different numbers of sensors were considered for locating an unknown node. Localization technique was tested with 3, 4, 5 and 6 sensors. The two key points are: there had to be at least 3 sensors to fulfill the triangulation method and the sensors could not be in a line to fulfill the triangulation condition.

Table 4.2: Combinations of sensors

Rank	Number of Sensors	Combination of the sensors used for localization	Error in localization for unknown node (marked as "1" in Figure 4.1) (in meters)	Error in localization for unknown node (marked as "2" in Figure 4.1) (in meters)	Error in localization for unknown node (marked as "3" in Figure 4.1) (in meters)
1	6	A, B, C, D, E, F	4.4854	5.0067	6.7409
2	5	A, B, C, D, F	4.3044	4.7443	6.4335
3	4	A, C, D, F	6.2607	7.4536	7.6958
4	4	B, C, D, F	2.4144	2.6645	6.2480
5	3	B, C, D	0.6738	3.6171	8.8724

In the experiment, pre-deployed sensors were considered always and the sensor locations were already known. The results have been grouped according to the resulting error. The various numbers of sensors were tested on the basis of the resultant error for each case to look for a combination that tends to result in a minimum error in indoor localization. According to the resulting error, the best number of sensors and the positioning of sensors were selected for further calculation. The selected unknown nodes location, shown in Figure 4.1, were used for all cases. Table 4.2 shows a few combinations of sensors with the corresponding error in localization.

From the above table and their respective errors, certain conclusion can be made. There is a significant effect of using various numbers of sensors and the distance between the sensors.

- By considering rank 1, 2 and 4, when 5 or 6 sensors in the formation as shown by Figure 4.1 were used for localization, the excess sensors actually move the estimated point from the actual location of the unknown point. Therefore, more sensors do not necessarily

provide better results. Also, another advantage will be the setup cost can be reduced by using a lesser (optimum) number of sensors.

- The difference between rank 3 and rank 4 is that the shape of the bounded region was trapezium (shown in Figure 4.4) in rank 3 whereas the shape of the bounded region was parallelogram (shown in Figure 4.5) in rank 4. As the shape of the bounded polygon is different so the separation between the sensors is also different thus resulted in an increase in error in localization. Furthermore, rank 4 configurations have yielded better results than rank 3, though both cases have 4 sensors. Thus, it proves that the positioning of sensors is more important than the number of sensors.

- The main striking observation is the two best scenarios rank 4 (4 sensors) and rank 5 (3 sensors). Now it is necessary to cross-check these scenarios to check if these cases are true for all unknown points. As a result, three unknown nodes were selected to test the efficacy of rank 4 and 5 scenarios. It is observed that the four sensors bounded the unknown nodes uniformly which resulted in a uniform low error in localization. Additionally, four sensors placed in a parallelogram formation is a more feasible scenario than a triangular formation.

- It is common understanding and knowledge that the centroid region, where the circles of content intersect, has the least error.

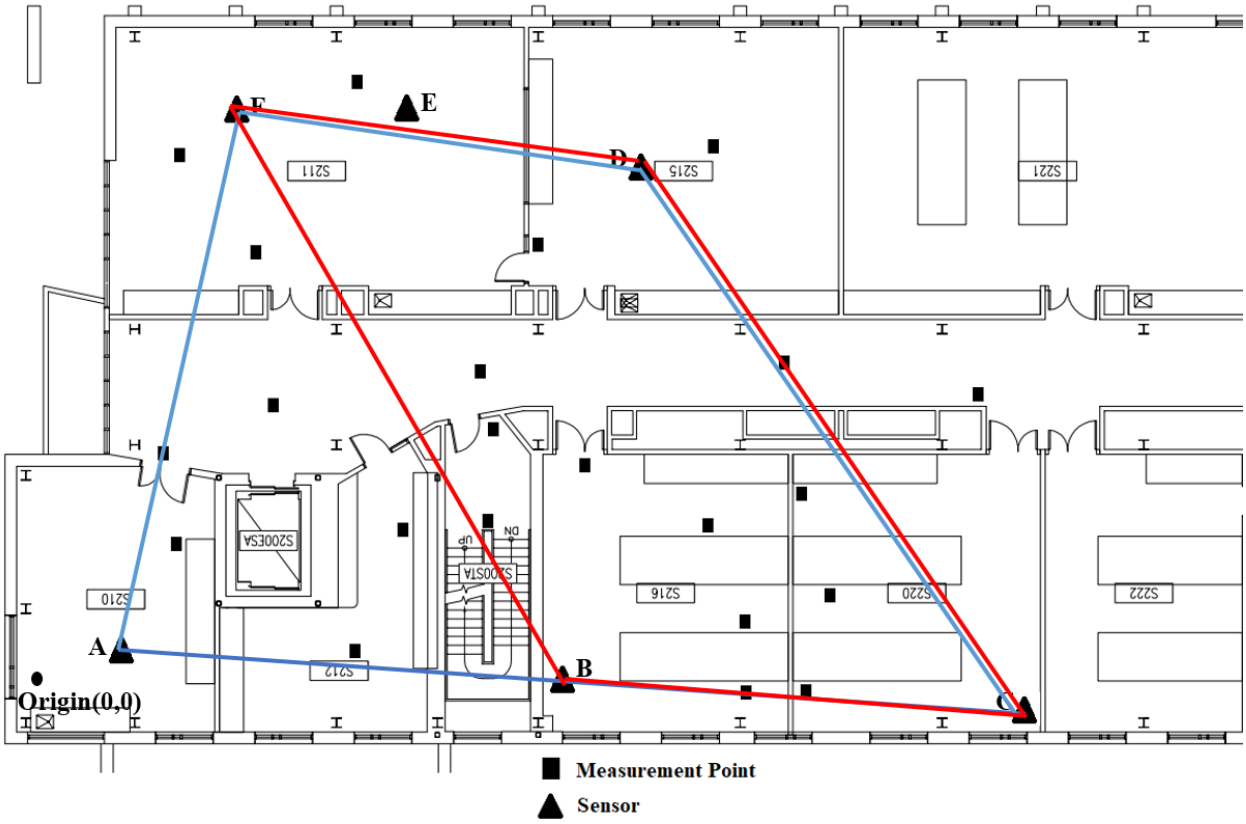


Figure 4.4: Floor plan of Kingsbury Hall, Second Floor, including measurement points bounded by 4 sensors in two different combinations

Finally, based on the above observations, it can be concluded that four sensors bounding a parallelogram region serves the best.

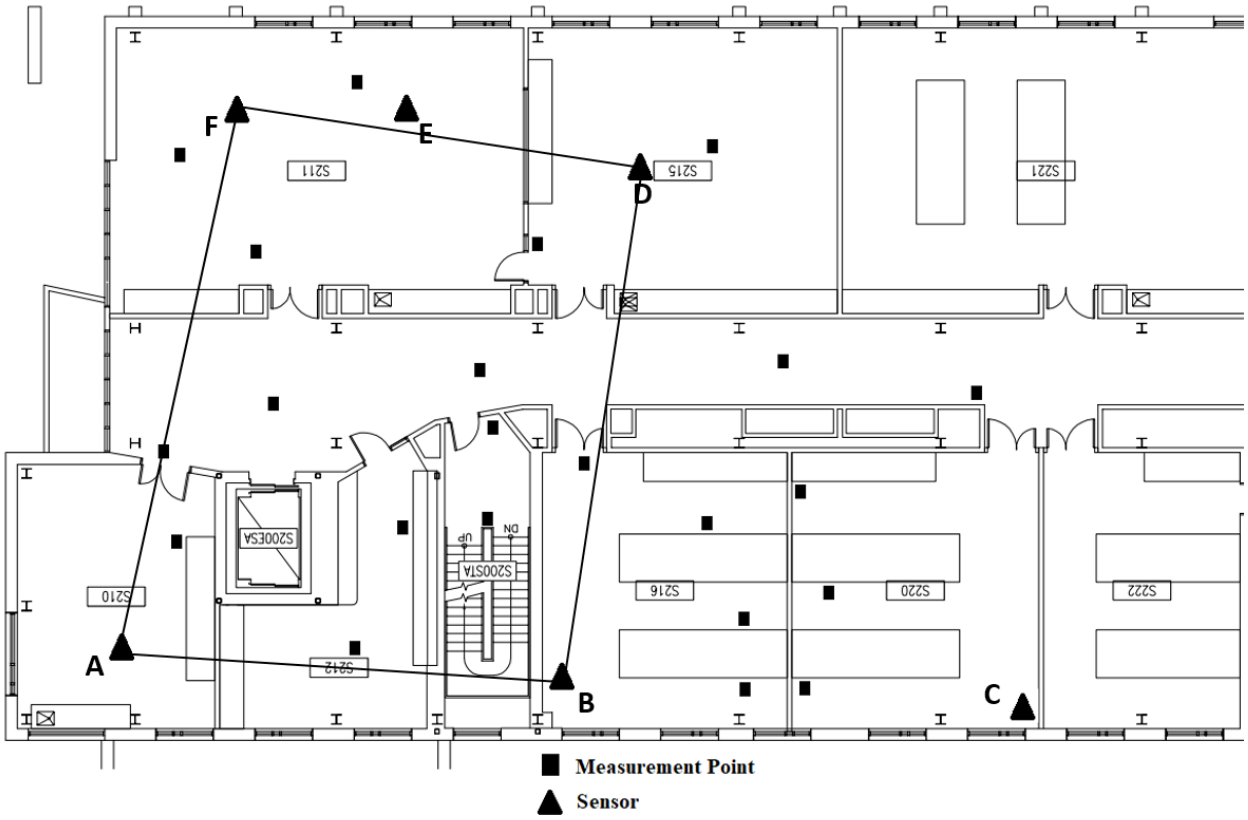


Figure 4.5: Floor plan of Kingsbury Hall, Second Floor, including measurement points bounded by 4 sensors for ABDF combinations

Figure 4.6 shows the MMSE corresponding to the distance of the unknown node from the origin in determining the actual location versus measured distances of the measurement points enclosed by the 4 sensors (ABDF combination shown in Figure 4.5) to understand the effect of distance from the origin on localization performance. It is again observed that MMSE produces the greatest error when the unknown node is closer to the origin and gradually performs better for farther distances from the origin. In Figure 4.6 almost all of the error values are lower compared with Figure 4.3.

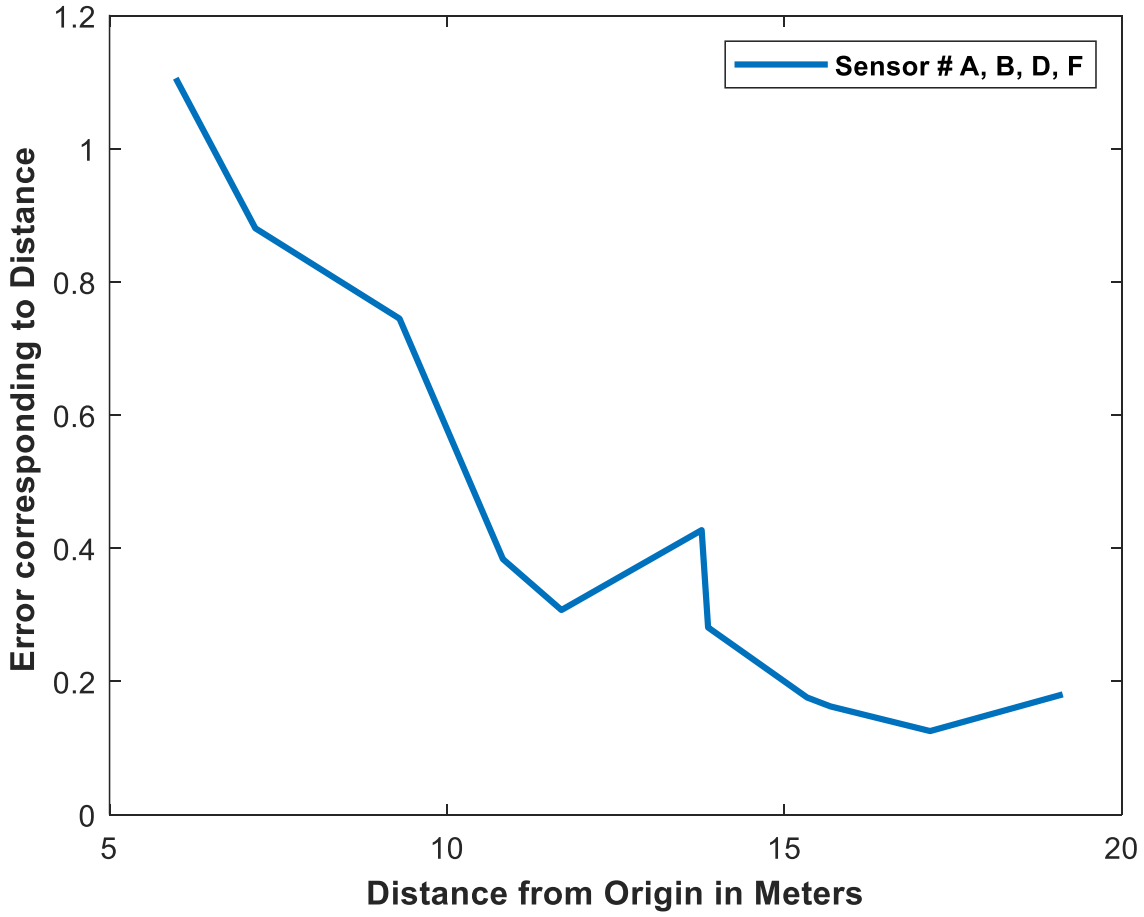
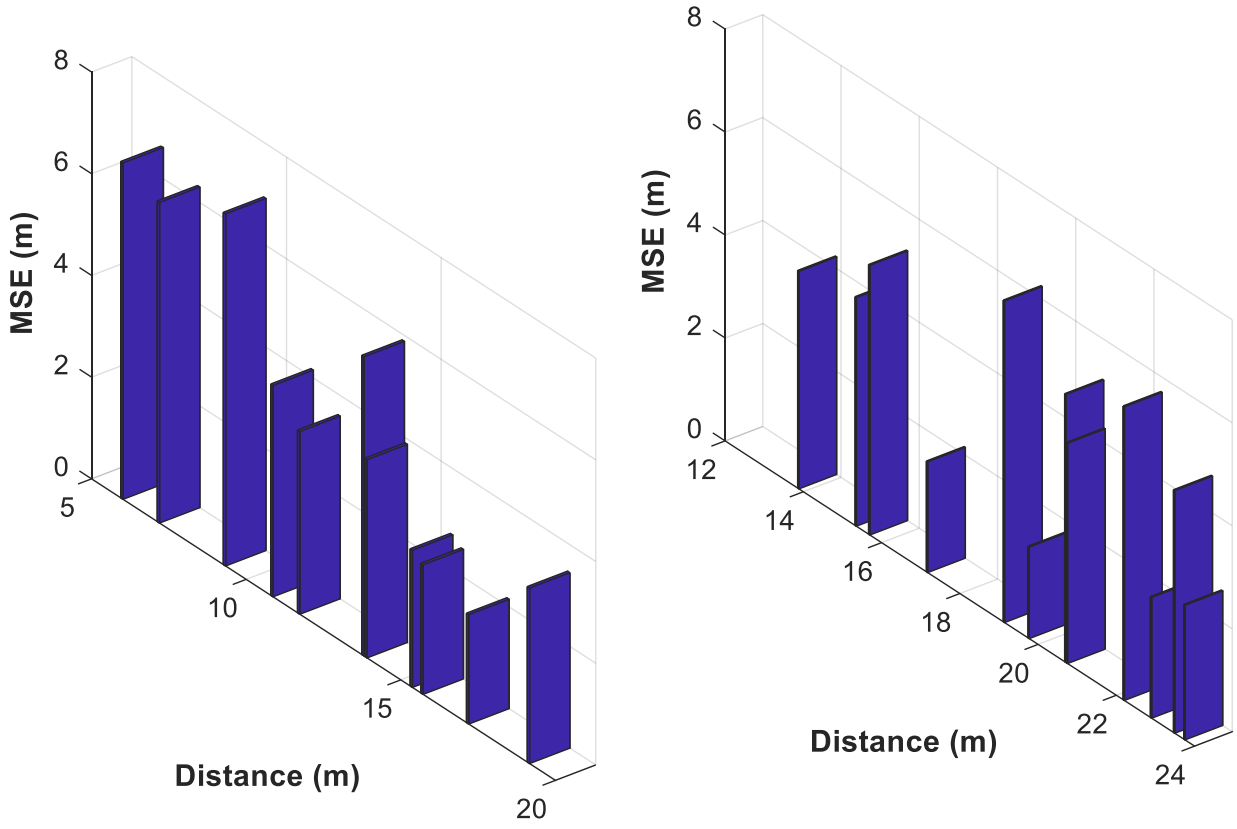


Figure 4.6: The localization of MSE corresponding to distance for different positions of the unknown node from the origin with 4 sensors

Furthermore, the probability of selecting the correct room using the MMSE location method is presented. Figure 4.7 (a) shows the probability distribution of the estimated error using one set of 4 sensors (ABDF shown in Figure 4.5), and (b) shows the probability distribution of the estimated error using another set of 4 sensors (BCDF shown in Figure 4.4). The reason for choosing two parallelogram regions (ABDF and BCDF) bounded by 4 sensors is to include all of the 22 measurement points, as the measurement points were selected randomly based on the coverage of our own BTS. For both of the cases, the region bounded by the sensors is a parallelogram, so producing less error. It is observed that the MMSE based localization method

can be applied to correctly estimate the room that a cell phone is in. Therefore, MMSE would be particularly useful for room occupancy estimation



(a) (b)

Figure 4.7: Probability distribution of estimated error (MMSE) in Kingsbury Hall Measurement Campaign with 4 sensors (a) for A, B, D, F combination and (b) for B, C, D, F combination

Table 4.3 shows the results of the measurement campaign with 4 sensors for both ABDF shown in Figure 4.5 and BCDF combinations shown in Figure 4.4. For both combinations, the probability of estimating the correct room is significantly closer that would be expected based on the previous observations. Although the probability of correct room estimation is relatively unexpected and would not be low like Table 4.3. This could be due to a slight bias of measurement

points away from the walls of rooms (due to a significant number of desks and bookshelves against walls, preventing measurement points against walls in many rooms).

Table 4.3: The probability of Correct Room Estimation in Kingsbury Measurement Campaign with 4 sensors

	B, C, D, F combination	A, B, D, F combination
Pr (Estimate correct room)	0.5	0.4

The Cumulative Distribution Function (CDF) of the estimated error (MMSE) of the measurement points bounded by the 4 sensors from the Kingsbury measurement campaign shows in Figure 4.8. The red line is the CDF of estimated error (MMSE) using BCDF combination shown in Figure 4.4 and the blue line is the CDF of the estimated error (MMSE) using ABDF combination shown in Figure 4.5. Based on the plots and their respective errors the conclusion can be made that MMSE performs better than expected majority of the time. The CDF of the MMSE is almost identical for both of the sensor combinations (ABDF and BCDF). The slight difference is for the effect of the different distances between the sensors.

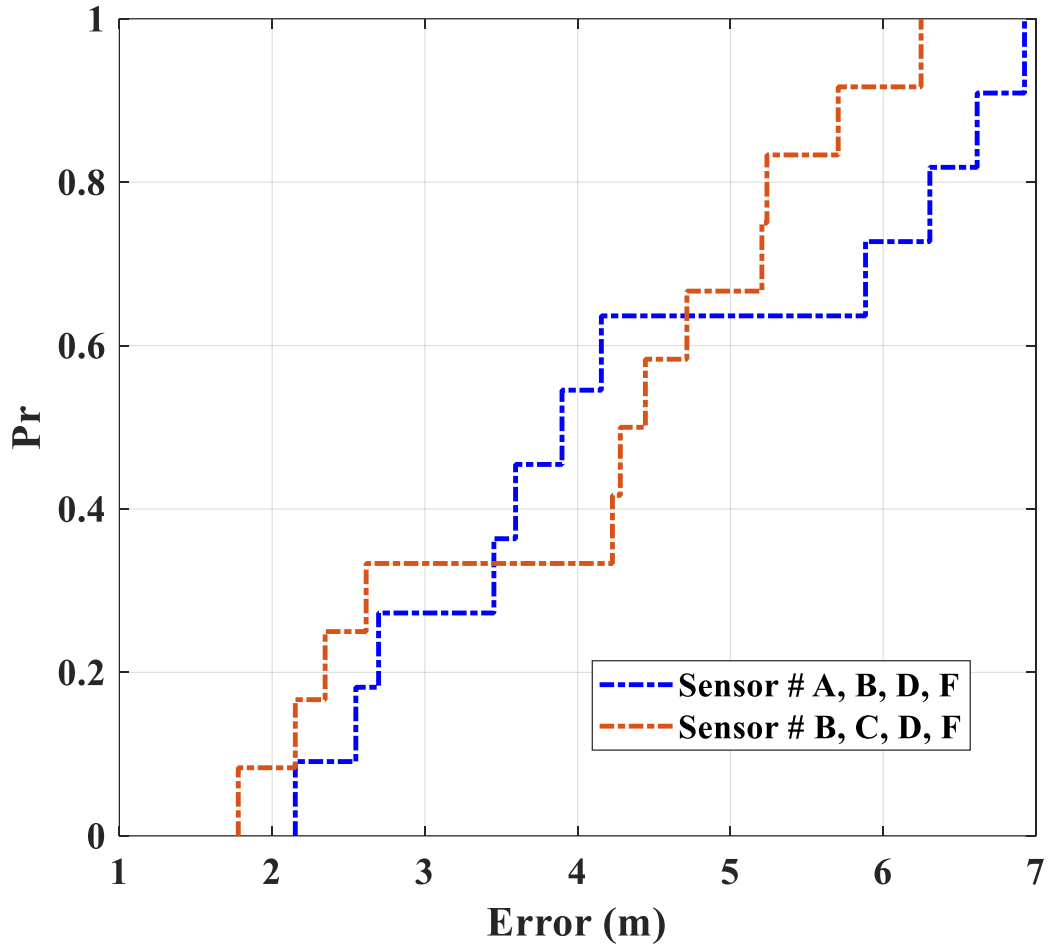
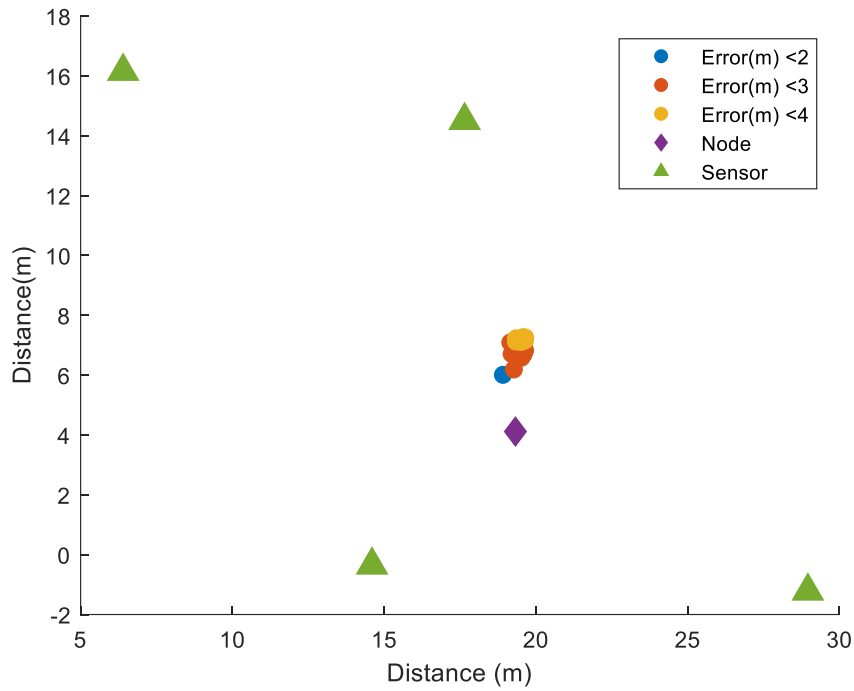
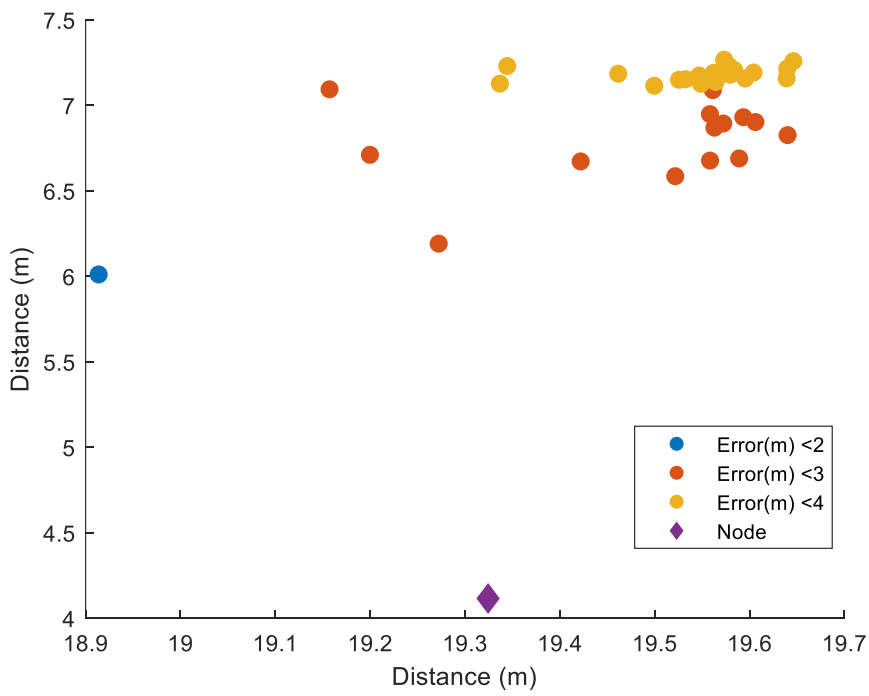


Figure 4.8: CDF of estimated error in Kingsbury Hall Measurement Campaign with 4 sensors

Figure 4.9 (a) shows the position of the sensors along with the actual and estimated position of an unknown node using 4 sensors for BCDF combination shown in Figure 4.4. Furthermore, Figure 4.9 (b) shows only the actual and estimated position of that unknown node to visualize the error distribution pattern more clearly. It is observed that using the MSE algorithm, the maximum error in determining an unknown node is less than 4 meters and the minimum error in determining the same unknown node is less than 2 meters.



(a)



(b)

Figure 4.9: MSE distribution in Kingsbury Hall Measurement Campaign with 4 sensors of an unknown node (a) the position of the sensors along with the actual and estimated position of the unknown node (b) only the actual and estimated position of the same unknown node

4.2 Office Measurement Campaign

SGH has a large office area around 45.72 x 152.4 meters shown in Figure 4.10 and a portion (~45.72 x 45.72 meters) of the office space was chosen during the experiment shown in Figure 4.11. The sensors were placed in locations as close to exterior walls of the building and corners as possible.

Figure 4.10: Map of SGH office

Figure 4.11 shows a map of the measurement region illustrating sensor positions, room boundaries, and measurement points. Based on the coverage area of the BTS, the location of the sensors and the measurement points were selected. In the measurement area, there were 23 unknown nodes (measurement points) indicated by black rectangles to be examined for six reference nodes (sensors marked by A – F) are also shown in Figure 4.11. The coordinates of the sensors and other parameters are shown in Table 4.4.

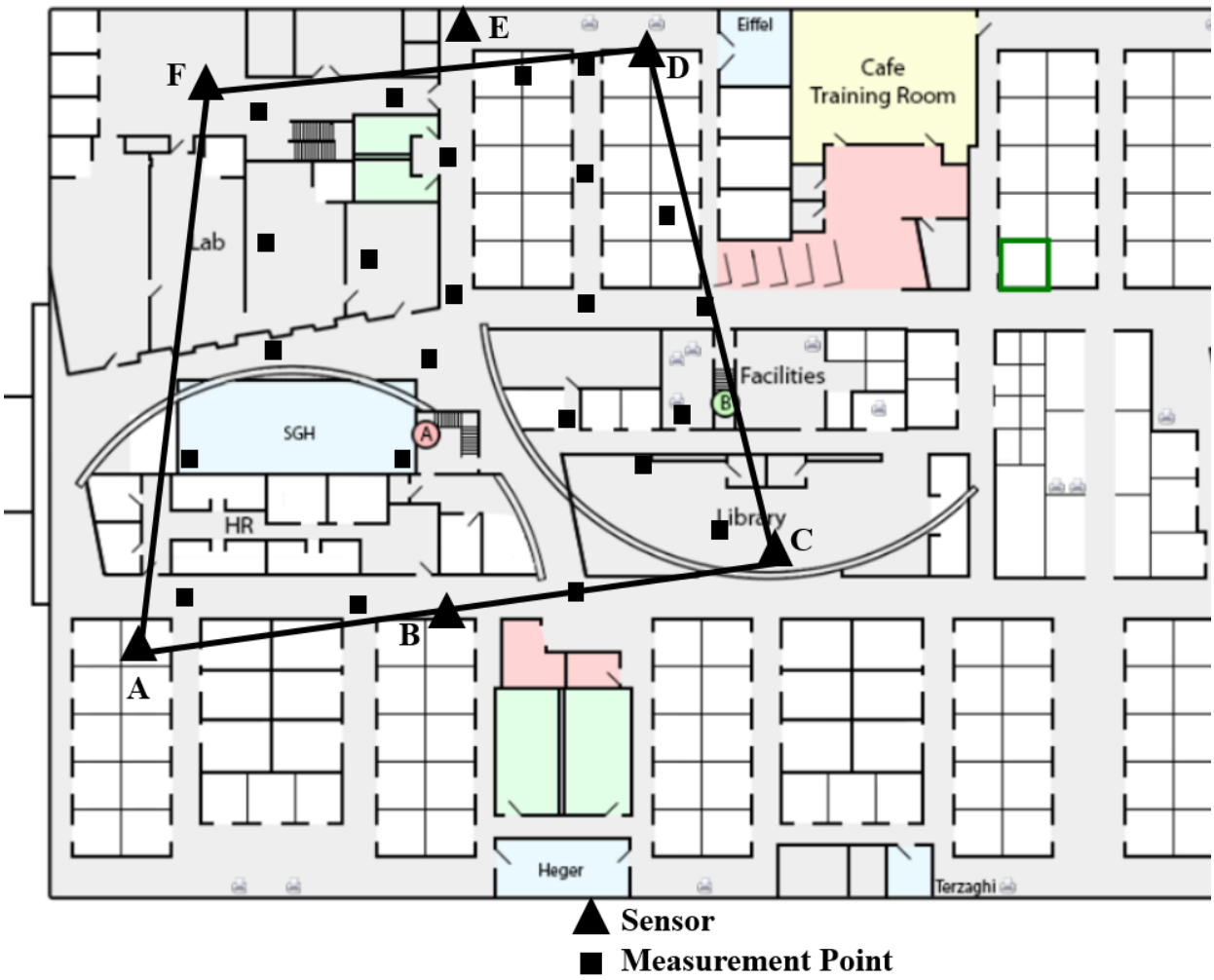


Figure 4.11: Partial Map of SGH office including sensors and measurement points

Like Kingsbury measurement campaign, the RSS power measurements were taken simultaneously by all six sensors and processed by a server which performed the location technique over the collected data. Like previous, the cell phone was placed at each measurement point, approximately 1.13 m off the ground to minimize the effect of multipath and variability in the environment.

Table 4.4: Simulation Parameters

Parameters	Value
Number of Sensors	6
Coordinates of the sensors (m)	A (-1.52, -3.05) B (17.61, -1.02) C (36.58, 2.88) D (29.97, 33.53) E (19.64, 35.66) F (4.40, 31.55)
Number of unknown nodes	23
Path loss exponent, *n*	4.1
The estimated distances (m)	6.01 – 45.87

Figure 4.12 shows the MMSE corresponding to the distance of the unknown node from the origin in determining the actual location versus measured distances of the measurement points enclosed by the 4 sensors (ACDF) from the SGH office measurement campaign to understand the effect of distance from the origin on localization performance. Like Kingsbury measurement campaign, it is again observed that MMSE produces the greatest error when the unknown node is closer to the origin and gradually performs better for farther distances from the origin. In Figure 4.12 almost all of the error values are quite similar to Figure 4.6 which validates the detection technique is applicable in any type of indoor environment.

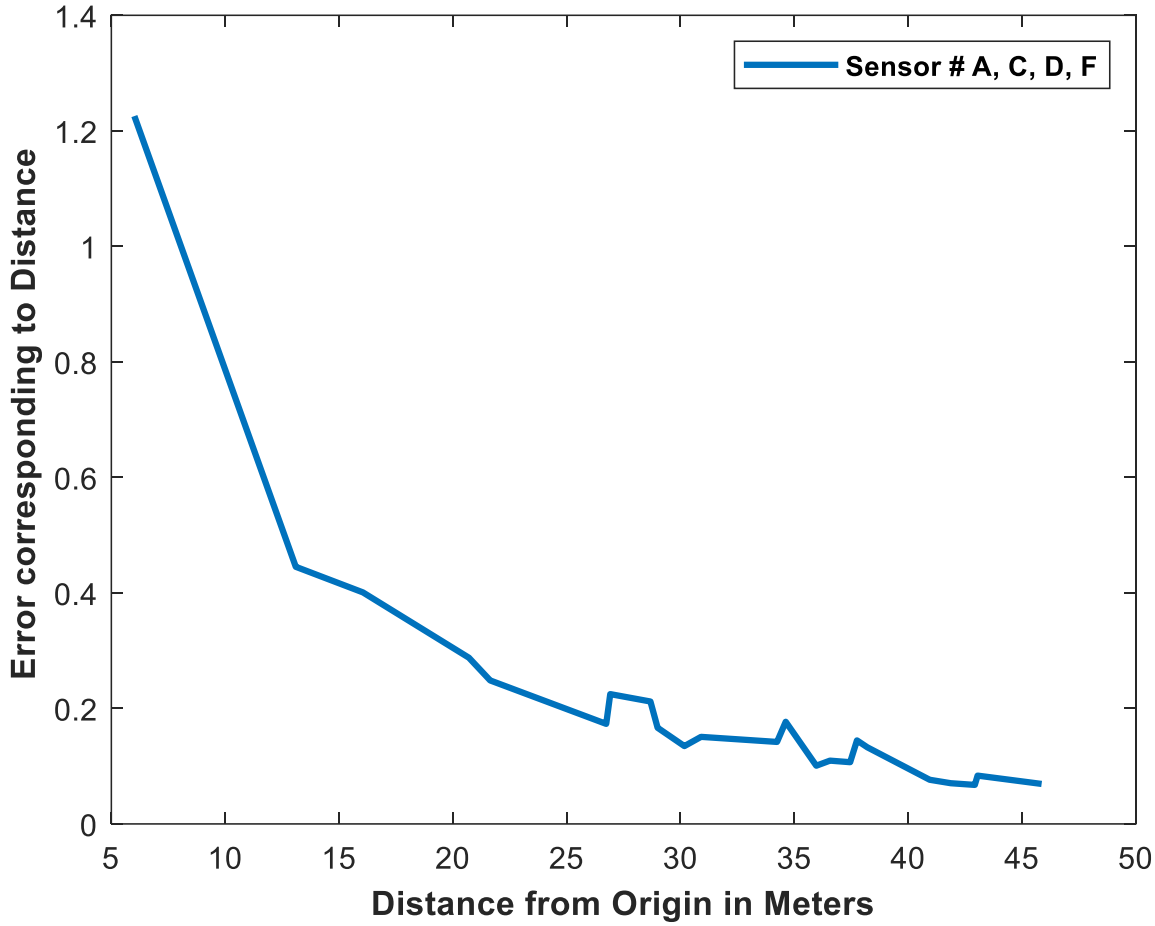


Figure 4.12: The localization of MSE corresponding to distance for different distances with 4 sensors

Furthermore, Figure 4.13 shows the probability distribution of the estimated error using 4 sensors (ACDF). Here as the region bounded by the sensors is a parallelogram, so producing less error. It is observed that using MMSE, it is possible for correctly estimating the room that a cell phone is in. Therefore, MMSE would be particularly useful for room occupancy estimation. The results are also similar to the Kingsbury measurement campaign which was expected.

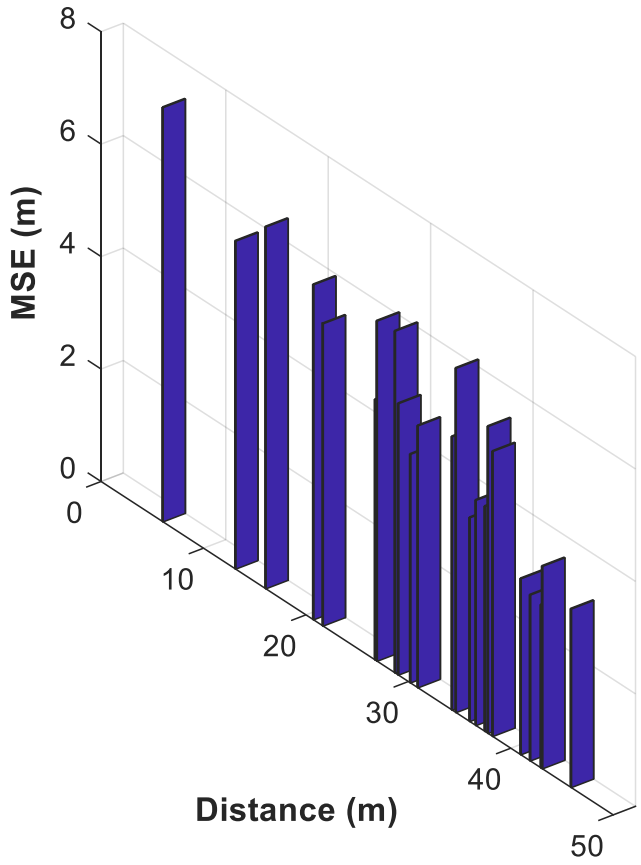


Figure 4.13: Probability distribution of estimated error (MMSE) in SGH office Measurement Campaign with 4 sensors (ACDF combination)

Observation 4:

From the results of the two measurement campaign, it is observed that the probability of estimating a position using the MMSE algorithm is ~55% which is not sufficient enough for room occupancy detection. However, the performance analysis of an MMSE based indoor localization with WSN using RSS was presented in [34], [35], [66] where it is observed that the simulation results (shown in Figure 4.14) were pretty similar to our calculated results collected from the real-

world environment scenario which validate the proposed framework. Therefore, the performance analysis of our proposed model is acceptable.

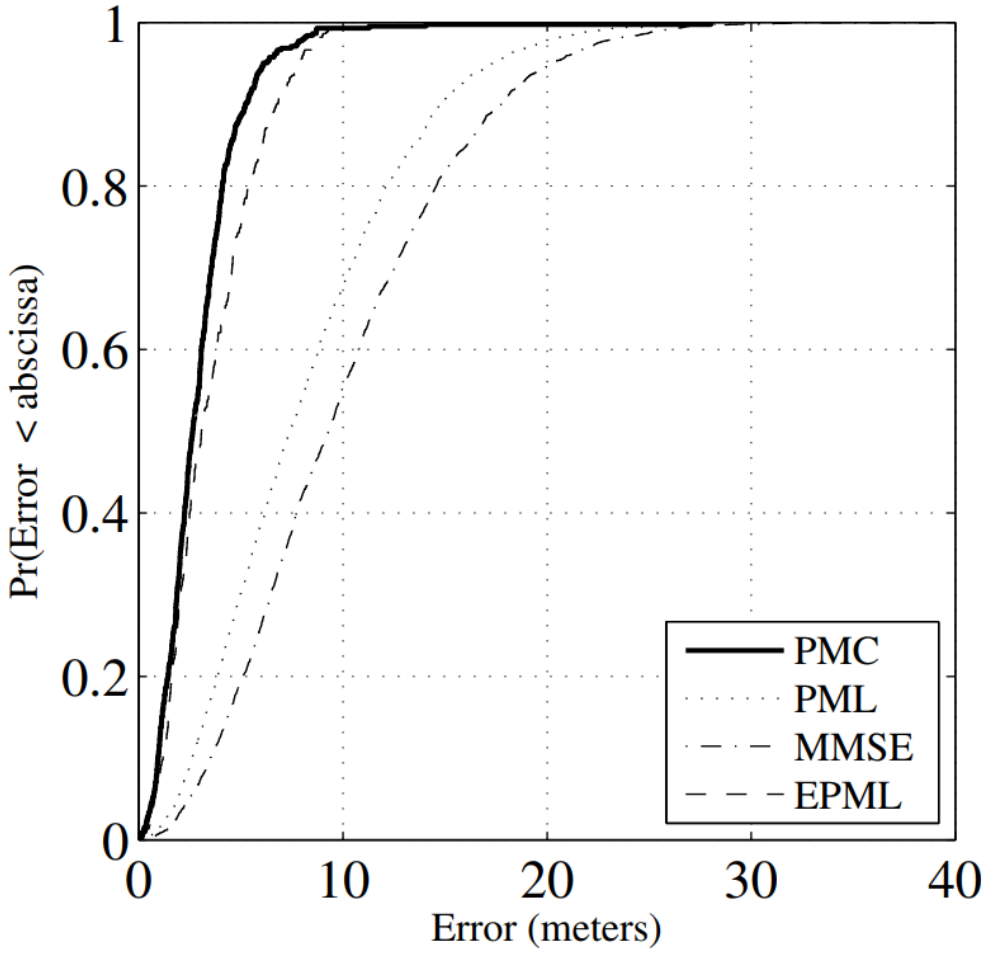


Figure 4.14: The estimation error curve using MMSE, PML, EPML, and PMC[35]

Finally, the reason for not selecting prior measurement comparison (PMC) as the localization method because this method collects fingerprint (an additional stage) for estimating the location. Additionally, for both PML and EPML method, a prior information is required to determine the most probable estimated position. As MMSE based localization method does not require a prior probability distribution information of the unknown nodes to decrease the error, for this reason, the performance analysis is done using MMSE based indoor localization.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This thesis project aimed to address the problem of indoor occupancy detection and developed an occupancy detection technique using wireless sensor networks to detect the RF energy from cellular devices, used universally by occupants, to locate the position of individuals in a building. Indoor occupancy detection is challenging because of the randomness of propagation caused by the walls and objects in buildings.

Using MMSE, the performance of our proposed framework was evaluated in real-world measurement scenario. The results showed that the performance analysis of the MMSE based indoor localization is comparable to the simulation results, for this reason, the proposed model is acceptable. It is also observed that more sensors or choosing the nearest sensors are not necessarily the best technique of occupancy detection.

Last observation is that it is not possible to perform occupancy detection when the cell phone is in idle mode. The reason behind that is: the MS does not transmit a signal continuously during idle mode. In idle mode, the MS transmits a signal only for location update i.e. while transferring from one BTS to another. After camping in a suitable cell (BTS), MS performs the necessary registration. After then the cell phone regularly checks if there is a better cell available compared with the selected cell. If one of the stored criteria changes such as the current serving cell becomes barred, or there is a downlink signaling failure only then a new cell is selected. MS continuously scan the downlink frequencies (RF channels) to check for changes in the stored parameters e.g.

receive system information and receive paging messages from the Public land mobile network (PLMN). The change in parameters will only occur when the MS is moved to another cell (BTS). When the MS finds that the received parameter is not matched with the stored parameters then MS transmits the signal to BTS for updating the current location. The selection of uplink frequency is based on the received downlink frequency.

In practice, BTS are grouped in areas of high population density, with the most potential users. The maximum range of a BTS (where it is not limited by interference with other BTSs nearby) depends on the ability of a low-powered personal cell phone to transmit back to the BTS. Based on a tall BTS and flat terrain, it may be possible to get between 50 and 70 km (30–45 miles). When the terrain is hilly, the maximum distance can vary from as little as 5 km (3.1 miles) to 8 km (5.0 miles) due to the encroachment of intermediate objects into the wide center Fresnel zone of the signal[75]. Depending on terrain and other circumstances, a GSM Tower can replace between 2 and 50 miles (80 km) of cabling for fixed wireless networks[75]. So inside the coverage radii of one BTS, the MS will not transmit any signal for location update. Hence, in idle mode, occupancy detection is not possible using the control signals transmitted from a cell phone.

Future work will investigate improving the accuracy of the wireless sensor networks by utilizing an improved path loss model for the building, and also implementing improved indoor localization technique to estimate the unknown node locations. Finally, performing the signal processing using the FPGA of the USRP B200 instead of the host computer, which will reduce cost.

BIBLIOGRAPHY

[1] T. Kindberg and A. Fox, “System Software for Ubiquitous Computing,” IEEE Pervasive Computing, vol. 1, no. 1, pp. 70–81, Jan. 2002.

[2] A. Hopper, A. Harter, and T. Blackie, “The Active Badge System (Abstract),” in Proceedings of the INTERACT ’93 and CHI ’93 Conference on Human Factors in Computing Systems, New York, NY, USA, 1993, pp. 533–534.

[3] P. Bahl and V. N. Padmanabhan, “RADAR: an in-building RF-based user location and tracking system,” in Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), 2000, vol. 2, pp. 775–784 vol.2.

[4] E. Elnahrawy, X. Li, and R. P. Martin, “The limits of localization using signal strength: a comparative study,” in 2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004., 2004, pp. 406–414.

[5] A. M. Ladd, K. E. Bekris, and A. Rudys, “Robotics-Based Location Sensing using Wireless Ethernet,” p. 12.

[6] L. Aalto, N. Göthlin, J. Korhonen, and T. Ojala, “Bluetooth and WAP Push Based Location-aware Mobile Advertising System,” in Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services, New York, NY, USA, 2004, pp. 49–58.

[7] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, “The Cricket location-support system,” in Proceedings of the 6th annual international conference on Mobile computing and networking - MobiCom ’00, Boston, Massachusetts, United States, 2000, pp. 32–43.

[8] A. Ward, A. Jones, and A. Hopper, “A new location technique for the active office,” IEEE Personal Communications, vol. 4, no. 5, pp. 42–47, Oct. 1997.

[9] I. Alyafawi, D. C. Dimitrova, and T. Braun, “Real-time passive capturing of the GSM radio,” in 2014 IEEE International Conference on Communications (ICC), 2014, pp. 4401–4406.

[10] V. Otsason, A. Varshavsky, A. LaMarca, and E. de Lara, “Accurate GSM Indoor Localization,” in UbiComp 2005: Ubiquitous Computing, 2005, pp. 141–158.

[11] “Enhancing smartphone indoor localization via opportunistic sensing - IEEE Conference Publication.” [Online]. Available: https://ieeexplore.ieee.org/document/7732988. [Accessed: 01-Oct-2018].

[12] N. E. Klepeis et al., “The National Human Activity Pattern Survey (NHAPS): a resource for assessing exposure to environmental pollutants,” Journal of Exposure Science and Environmental Epidemiology, vol. 11, no. 3, pp. 231–252, Jul. 2001.

[13] H. Scribner, “‘Indoor generation’: Here’s how much time we spend indoors,” DeseretNews.com, 16-May-2018. [Online]. Available: https://www.deseretnews.com/article/900018757/indoor-generation-heres-how-much-time-we-spend-indoors.html. [Accessed: 29-Nov-2018].

[14] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of Wireless Indoor Positioning Techniques and Systems,” Trans. Sys. Man Cyber Part C, vol. 37, no. 6, pp. 1067–1080, Nov. 2007.

[15] L. Pérez-Lombard, J. Ortiz, and C. Pout, “A review on buildings energy consumption information,” Energy and Buildings, vol. 40, no. 3, pp. 394–398, Jan. 2008.

[16] J. Yang, M. Santamouris, and S. E. Lee, “Review of occupancy sensing systems and occupancy modeling methodologies for the application in institutional buildings,” Energy and Buildings, vol. 121, pp. 344–349, Jun. 2016.

[17] F. Tahmasebi and A. Mahdavi, “The sensitivity of building performance simulation results to the choice of occupants’ presence models: a case study,” Journal of Building Performance Simulation, vol. 10, no. 5–6, pp. 625–635, Nov. 2017.

[18] A. Mahdavi and F. Tahmasebi, “The deployment-dependence of occupancy-related models in building performance simulation,” Energy and Buildings, vol. 117, pp. 313–320, Apr. 2016.

[19] C. Duarte, R. Budwig, and K. V. D. Wymelenberg, “Energy and demand implication of using recommended practice occupancy diversity factors compared to real occupancy data in whole building energy simulation,” Journal of Building Performance Simulation, vol. 8, no. 6, pp. 408–423, Nov. 2015.

[20] Y. Tachwali, H. Refai, and J. E. Fagan, “Minimizing HVAC Energy Consumption Using a Wireless Sensor Network,” in IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society, 2007, pp. 439–444.

[21] S. S. K. Kwok, R. K. K. Yuen, and E. W. M. Lee, “An intelligent approach to assessing the effect of building occupancy on building cooling load prediction,” Building and Environment, vol. 46, no. 8, pp. 1681–1690, Aug. 2011.

[22] Z. Yang and B. Becerik-Gerber, “Modeling personalized occupancy profiles for representing long term patterns by using ambient context,” Building and Environment, vol. 78, pp. 23–35, Aug. 2014.

[23] Z. Yang and B. Becerik-Gerber, “How Does Building Occupancy Influence Energy Efficiency of HVAC Systems?,” Energy Procedia, vol. 88, pp. 775–780, Jun. 2016.

[24] J. Brooks, S. Kumar, S. Goyal, R. Subramany, and P. Barooah, "Energy-efficient control of under-actuated HVAC zones in commercial buildings," Energy and Buildings, vol. 93, pp. 160–168, Apr. 2015.

[25] B. Dong and K. P. Lam, "Building energy and comfort management through occupant behaviour pattern detection based on a large-scale environmental sensor network," Journal of Building Performance Simulation, vol. 4, no. 4, pp. 359–369, Dec. 2011.

[26] V. Dhummi, D. Demetriou, H. J. Palanthandalam-Madapusi, H. E. Khalifa, and C. Isik, "Robust Occupancy-Based Distributed Demand Control Ventilation," International Journal of Ventilation, vol. 9, no. 4, pp. 359–369, Mar. 2011.

[27] Y. Agarwal, B. Balaji, S. Dutta, R. K. Gupta, and T. Weng, "Duty-cycling buildings aggressively: The next frontier in HVAC control," in Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks, 2011, pp. 246–257.

[28] "The impact of demand-controlled and economizer ventilation strategies on energy use in buildings - ProQuest." [Online]. Available: https://search.proquest.com/openview/d44332bb1d1ac384a1793c9e025790c5/1?pq-origsite=gscholar&cbl=34619. [Accessed: 02-Oct-2018].

[29] T. H. Pedersen, K. U. Nielsen, and S. Petersen, "Method for room occupancy detection based on trajectory of indoor climate sensor data," Building and Environment, vol. 115, pp. 147–156, Apr. 2017.

[30] T. Labeodan, C. De Bakker, A. Rosemann, and W. Zeiler, "On the application of wireless sensors and actuators network in existing buildings for occupancy detection and occupancy-driven lighting control," Energy and Buildings, vol. 127, pp. 75–83, Sep. 2016.

[31] A. Peruffo, A. Pandharipande, D. Caicedo, and L. Schenato, "Lighting control with distributed wireless sensing and actuation for daylight and occupancy adaptation," Energy and Buildings, vol. 97, pp. 13–20, Jun. 2015.

[32] P. Correia da Silva, V. Leal, and M. Andersen, "Occupants interaction with electric lighting and shading systems in real single-occupied offices: Results from a monitoring campaign," Building and Environment, vol. 64, pp. 152–168, Jun. 2013.

[33] T. Pedersen, M. Knudsen, R. Hedegaard, and S. Petersen, "Handling Stochastic Occupancy in an Economic Model Predictive Control Framework for Heating System Operation in Dwellings," 2016.

[34] "Enhancing indoor probabilistic localization methods using prior information - ProQuest." [Online]. Available: https://search-proquest-com.libproxy.unh.edu/docview/1508835985. [Accessed: 01-Oct-2018].

[35] R. Bera, N. J. Kirsch, and T. S. Fu, "An Indoor Probabilistic Localization Method Using Prior Information," in 2013 IEEE 78th Vehicular Technology Conference (VTC Fall), 2013, pp. 1–5.

[36] R. Bera, N. J. Kirsch, and T. S. Fu, "Using prior measurements to improve probabilistic-based indoor localization methods," in 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2013, vol. 01, pp. 478–482.

[37] "Performance Analysis of Localization Techniques with Generalized Prior Distributions." [Online]. Available: http://www.sensorsportal.com/HTML/DIGEST/P_2646.htm. [Accessed: 14-Feb-2018].

[38] J. R. Tefft, N. J. Kirsch, T. M. Adams, and T. S. Fu, "Wireless sensor system for detection of occupants to increase building energy efficiency," in 2016 IEEE International Smart Cities Conference (ISC2), 2016, pp. 1–6.

[39] "Number of mobile phone users worldwide 2015-2020," Statista. [Online]. Available: https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/. [Accessed: 15-Nov-2018].

[40] Y. Benezeth, H. Laurent, B. Emile, and C. Rosenberger, "Towards a sensor for detecting human presence and characterizing activity," Energy and Buildings, vol. 43, no. 2, pp. 305–314, Feb. 2011.

[41] H.-C. Shih, "A robust occupancy detection and tracking algorithm for the automatic monitoring and commissioning of a building," Energy and Buildings, vol. 77, pp. 270–280, Jul. 2014.

[42] S. Petersen, T. H. Pedersen, K. U. Nielsen, and M. D. Knudsen, "Establishing an image-based ground truth for validation of sensor data-based room occupancy detection," Energy and Buildings, vol. 130, pp. 787–793, Oct. 2016.

[43] X. Zhang, J. Yan, S. Feng, Z. Lei, D. Yi, and S. Z. Li, "Water Filling: Unsupervised People Counting via Vertical Kinect Sensor," in 2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance, 2012, pp. 215–220.

[44] Y. Zhao, W. Zeiler, G. Boxem, and T. Labeodan, "Virtual occupancy sensors for real-time occupancy information in buildings," Building and Environment, vol. 93, pp. 9–20, Nov. 2015.

[45] V. L. Erickson, M. Á. Carreira-Perpiñán, and A. E. Cerpa, "Occupancy Modeling and Prediction for Building Energy Management," ACM Transactions on Sensor Networks, vol. 10, no. 3, pp. 1–28, May 2014.

[46] J. R. Dobbs and B. M. Hencey, "Model predictive HVAC control with online occupancy model," Energy and Buildings, vol. 82, pp. 675–684, Oct. 2014.

[47] R. H. Dodier, G. P. Henze, D. K. Tiller, and X. Guo, "Building occupancy detection through sensor belief networks," Energy and Buildings, vol. 38, no. 9, pp. 1033–1043, Sep. 2006.

[48] C. Duarte, K. Van Den Wymelenberg, and C. Rieger, “Revealing occupancy patterns in an office building through the use of occupancy sensor data,” Energy and Buildings, vol. 67, pp. 587–595, Dec. 2013.

[49] B. Dong et al., “An information technology enabled sustainability test-bed (ITEST) for occupancy detection through an environmental sensing network,” Energy and Buildings, vol. 42, no. 7, pp. 1038–1046, Jul. 2010.

[50] G. Ansanay-Alex, “Estimating Occupancy Using Indoor Carbon Dioxide Concentrations Only in an Office Building: a Method and Qualitative Assessment,” 2013.

[51] C. Jiang, M. K. Masood, Y. C. Soh, and H. Li, “Indoor occupancy estimation from carbon dioxide concentration,” Energy and Buildings, vol. 131, pp. 132–141, Nov. 2016.

[52] L. M. Candanedo and V. Feldheim, “Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models,” Energy and Buildings, vol. 112, pp. 28–39, Jan. 2016.

[53] S. H. Ryu and H. J. Moon, “Development of an occupancy prediction model using indoor environmental data based on machine learning techniques,” Building and Environment, vol. 107, pp. 1–9, Oct. 2016.

[54] X. Liang, T. Hong, and G. Q. Shen, “Occupancy data analytics and prediction: A case study,” Building and Environment, vol. 102, pp. 179–192, Jun. 2016.

[55] K. P. Lam et al., “INFORMATION-THEORETIC ENVIRONMENTAL FEATURES SELECTION FOR OCCUPANCY DETECTION IN OPEN OFFICES,” p. 8.

[56] “Wireless Communications: Principles and Practice, 2nd Edition.” [Online]. Available: http://www.mypearsonstore.com/bookstore/wireless-communications-principles-and-practice-0130422320. [Accessed: 03-Oct-2018].

[57] N. A. M. Razali, M. H. Habaebi, N. F. B. Zulkurnain, R. Islam, and A.-H. Zyoud, “The Distribution of Path Loss Exponent in 3D Indoor Environment,” vol. 12, no. 18, p. 8, 2017.

[58] K. Curran, E. Furey, T. Lunney, J. Santos, D. Woods, and A. McCaughey, “An evaluation of indoor location determination technologies,” Journal of Location Based Services, vol. 5, no. 2, pp. 61–78, Jun. 2011.

[59] S. He and S.-G. Chan, “Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons,” IEEE Communications Surveys Tutorials, vol. 18, no. 1, pp. 466–490, Firstquarter 2016.

[60] C. Gentile, N. Alsindi, R. Raulefs, and C. Teolis, Geolocation Techniques: Principles and Applications. New York: Springer-Verlag, 2013.

[61] L. W. C. Wong, “Indoor Localization Methods,” p. 33.

[62] D. Niculescu and B. Nath, “Ad hoc positioning system (APS) using AOA,” in IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428), 2003, vol. 3, pp. 1734–1743 vol.3.

[63] Dan Li, K. D. Wong, Yu Hen Hu, and A. M. Sayeed, “Detection, classification, and tracking of targets,” IEEE Signal Processing Magazine, vol. 19, no. 2, pp. 17–29, Mar. 2002.

[64] A. Savvides, C.-C. Han, and M. B. Strivastava, “Dynamic fine-grained localization in Ad-Hoc networks of sensors,” in Proceedings of the 7th annual international conference on Mobile computing and networking - MobiCom ’01, Rome, Italy, 2001, pp. 166–179.

[65] “Localization using Cooperative AOA Approach - IEEE Conference Publication.” [Online]. Available: https://ieeexplore.ieee.org/document/4340377. [Accessed: 05-Oct-2018].

[66] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal, “Locating the nodes: cooperative localization in wireless sensor networks,” IEEE Signal Processing Magazine, vol. 22, no. 4, pp. 54–69, Jul. 2005.

[67] Y.-F. Huang, Y.-T. Jheng, and H.-C. Chen, “Performance of an MMSE based indoor localization with wireless sensor networks,” in The 6th International Conference on Networked Computing and Advanced Information Management, 2010, pp. 671–675.

[68] W. B. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” IEEE Trans. Wireless Communications, vol. 1, pp. 660–670, 2002.

[69] J. Arnold, N. Bean, M. Kraetzl, and M. Roughan, “Node Localisation in Wireless Ad Hoc Networks,” in 2007 15th IEEE International Conference on Networks, 2007, pp. 1–6.

[70] “b200-b210_spec_sheet.pdf.” .

[71] “BuildInstallRun - OpenBTS.” [Online]. Available: http://openbts.org/w/index.php?title=BuildInstallRun. [Accessed: 11-Sep-2018].

[72] Narang, 2G Mobile Networks. Tata McGraw-Hill Education, 2006.

[73] “UmTRX » OpenBTS.” .

[74] “Importance of Time Synchronization for Your Network.” [Online]. Available: https://www.endruntechnologies.com/network-time-synchronization.htm. [Accessed: 06-Oct-2018].

[75] “Cell site,” Wikipedia. 30-Sep-2018.

[76] “Building and Installing the USRP Open-Source Toolchain (UHD and GNU Radio) on Linux - Ettus Knowledge Base.” [Online]. Available: https://kb.ettus.com/Building_and_Installing_the_USRP_Open-Source_Toolchain_(UHD_and_GNU_Radio)_on_Linux. [Accessed: 11-Sep-2018].

[77] "Verifying the Operation of the USRP Using UHD and GNU Radio - Ettus Knowledge Base." [Online]. Available: https://kb.ettus.com/Verifying_the_Operation_of_the_USRP_Using_UHD_and_GNU_Radio . [Accessed: 11-Sep-2018].

[78] Start Here! Development Environment Tools and System Releases: RangeNetworks/dev. Range Networks, 2018.

[79] nadiia-kotelnikova, Contribute to nadiia-kotelnikova/openbts_systemd_scripts development by creating an account on GitHub. 2018.

[80] M. Iedema, Getting started with OpenBTS: build open source mobile networks, First edition. Sebastopol, CA: O'Reilly, 2015.

[81] "ProgrammingSIMcards - OpenBTS." [Online]. Available: http://openbts.org/w/index.php?title=ProgrammingSIMcards. [Accessed: 21-Sep-2018].

[82] "Wiki - pySim - Open Source Mobile Communications." [Online]. Available: http://osmocom.org/projects/pysim/wiki. [Accessed: 21-Sep-2018].

[83] "Install Communications Toolbox Support Package for USRP Radio - MATLAB & Simulink." [Online]. Available: https://www.mathworks.com/help/supportpkg/usrpradio/ug/install-support-package-for-usrp-radio.html. [Accessed: 15-Oct-2018].

[84] "time - ntpdate: no server suitable for synchronization found," Ask Ubuntu. [Online]. Available: https://askubuntu.com/questions/429306/ntpdate-no-server-suitable-for-synchronization-found. [Accessed: 06-Oct-2018].

Appendix A Prerequisite Installation

This section explains the steps required to follow to install necessary applications and dependencies before installing OpenBTS.

Appendix A.1 Ubuntu 16.04.4 Installation

Ubuntu Desktop 64-bit (not 32-bit) 16.04.4 LTS is used as the OS. It is recommended to use USB 3.0 flash drive, (not USB 2.0) to set the Ubuntu desktop onto the host computer because USB 2.0 flash drive is taken a significantly longer time to install. Before starting the installation, it is required to ensure that the computer has at least 25GB of free storage space, or 5GB for a minimal installation and have a recent backup of the data. There are many tutorials available online to guide about Ubuntu Installation.

Appendix A.2 Updating and Installing Dependencies

Before building UHD it is required to make sure that all the dependencies are first installed. Before installing any dependencies, first, it needs to make sure that all the packages that are already installed on the system are up-to-date [76]. This can be done from a graphical user interface (GUI), or from the command-line. To update the packages on Ubuntu systems run the following command:

```
sudo apt-get update
```

Once the system has been updated, then install the required dependencies for UHD. On Ubuntu 16.04 systems run the following command:

```
sudo apt-get -y install git swig cmake doxygen build-essential libboost-all-dev libtool libusb-1.0-0 libusb-1.0-0-dev libudev-dev libncurses5-dev libfftw3-bin libfftw3-dev libfftw3-doc libcppunit-1.13-0v5 libcppunit-dev libcppunit-doc ncurses-bin cpufrequtils python-numpy python-numpy-doc python-numpy-dbg python-scipy python-docutils qt4-bin-dbg qt4-default qt4-doc libqt4-dev libqt4-dev-bin python-qt4 python-qt4-dbg python-qt4-dev python-qt4-doc python-qt4-doc libqwt6abi1 libfftw3-bin libfftw3-dev libfftw3-doc ncurses-bin libncurses5 libncurses5-dev libncurses5-dbg libfontconfig1-dev libxrender-dev libpulse-dev swig g++ automake autoconf libtool python-dev libfftw3-dev libcppunit-dev libboost-all-dev libusb-dev libusb-1.0-0-dev fort77 libsdl1.2-dev python-wxgtk3.0 git-core libqt4-dev python-numpy ccache python-opengl libgsl-dev python-cheetah python-mako python-lxml doxygen qt4-default qt4-dev-tools libusb-1.0-0-dev libqwt5-qt4-dev libqwtplot3d-qt4-dev pyqt4-dev-tools python-qwt5-qt4 cmake git-core wget libxi-dev gtk2-engines-pixbuf r-base-dev python-tk liborc-0.4-0 liborc-0.4-dev libasound2-dev python-gtk2 libzmq-dev libzmq1 python-requests python-sphinx libcomedi-dev python-zmq
```

After installing the dependencies, the system should be rebooted. If the installation of the dependencies completes without any errors, then proceed to the next section.

Appendix A.3 Building and installing UHD from source code

UHD is open-source and is hosted on GitHub. To build UHD from source code, clone the GitHub repository, check out a branch or tagged release of the repository, and build and install. It is required to make sure that no USRP device is connected to the system at this point.[76]. First, make a folder to hold the repository.

```
cd $HOME
mkdir workarea-uhd
cd workarea-uhd
```

Next, clone the repository and change into the cloned directory.

```
git clone https://github.com/EttusResearch/uhd
cd uhd
```

Next, check out the compatible UHD version (**UHD 3.10.3.0** is used in this thesis project) by running the command:

```
git checkout release_003_010_003_000
```

Next, create a *build* folder within the repository.

```
cd host
mkdir build
cd build
```

Next, invoke `cmake` to create the *Makefiles*.

```
cmake ../
```

Next, run `make` to build UHD.

```
make
```

Next, some basic tests can be run optionally to verify that the build process completed properly.

```
make test
```

Next, install UHD, using the default install prefix, which will install UHD under the `/usr/local/lib` folder. This is needed to run as root due to the permissions on that folder.

```
sudo make install
```

Next, update the system's shared library cache.

```
sudo ldconfig
```

Finally, it is required to make sure that the *LD_LIBRARY_PATH* environment variable is defined and includes the folder under which UHD was installed. To do this, the below line should be added to the end of the *$HOME/.bashrc* file:

```
export LD_LIBRARY_PATH=/usr/local/lib
```

If the *LD_LIBRARY_PATH* environment variable is already defined with other folders in the *$HOME/.bashrc* file, then add the line below to the end of the *$HOME/.bashrc* file to preserve the current settings.

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

For this change to take effect, the current terminal window will be needed to close and then open a new terminal. At this point, UHD should be installed and ready to use. It can be tested by running *sudo uhd_find_devices* command. As no USRP device is attached, something similar to the following should be seen.

```
linux; GNU C++ version 4.8.4; Boost_105400; UHD_003.010.000.HEAD-0-g6e1ac3fc

No UHD Devices Found
```

Appendix A.4 Building and installing GNU Radio from source code

GNU Radio is installed to verify the installation of UHD. GNU Radio is also open-source and is hosted on GitHub. Similar to the process for UHD, to build GNU Radio from source code, clone the GitHub repository, then check out a branch or tagged release of the repository, and finally build and install. It is required to make sure that no USRP device is connected to the system at this point [76]. First, make a folder to hold the repository.

```
 cd $HOME
mkdir workarea-gnuradio
cd workarea-gnuradio
```

Next, clone the repository.

```
git clone --recursive https://github.com/gnuradio/gnuradio
```

Next, go into the repository and check out the compatible GNU Radio version.

```
cd gnuradio
git checkout v3.7.11
git submodule update --init --recursive
```

Next, create a *build* folder within the repository.

```
mkdir build
cd build
```

Next, invoke `cmake` to create the Makefiles.

```
cmake ../
```

Next, run `make` to build GNU Radio.

```
make
```

Next, some basic tests can be run optionally to verify that the build process completed properly.

```
make test
```

Next, install GNU Radio, using the default install prefix, which will install GNU Radio under the `/usr/local/lib` folder. This is needed to run as root due to the permissions on that folder.

```
sudo make install
```

Finally, update the system's shared library cache.

```
sudo ldconfig
```

At this point, GNU Radio should be installed and ready to use. This can be tested by running the following commands. USRP device does not require to be attached for this test.

```
gnuradio-config-info --version
gnuradio-config-info –prefix
gnuradio-config-info --enabled-components
```

Following is a simple flow graph that also does not require any USRP hardware. It is called the dial-tone test and produces a PSTN (Public Switched Telephone Network) dial tone on the computer's speakers. Running it verifies that all the libraries can be found and that the GNU Radio run-time is working.

```
python $HOME/workarea-gnuradio/gnuradio/gr-audio/examples/python/dial_tone.py
```

To launch the GNU Radio Companion (GRC) tool, a visual tool for building and running GNU Radio flow graphs, run the following command:

```
gnuradio-companion
```

If `gnuradio-companion` does not start and complains about the `PYTHONPATH` environment variable, then add the line below to the end of the `$HOME/.bashrc` file.

```
export PYTHONPATH=/usr/local/lib/python2.7/dist-packages
```

Appendix A.5 Configuring USB

On Linux, *udev* handles USB plug and unplug events. The following commands install an *udev* rule so that non-root users may access the device. This step is only necessary for devices that use USB to connect to the host computer, such as the B200, B210, and B200mini. This setting should take effect immediately and does not require a reboot or logout/login. It is required to make sure that no USRP device is connected via USB when running these commands.

```
cd $HOME/workarea-uhd/uhd/host/utils
sudo cp uhd-usrp.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules
sudo udevadm trigger
```

Appendix A.6 Connecting the USRP

After completing the installation of UHD and GNU Radio, connect the USRP to the host computer. As the interface is USB, so open a new terminal window, and run *lsusb*. It should display the USRP listed on the USB bus with a VID of *2500* and PID of *0020* for B200. Now *sudo uhd_find_devices* command should find UHD-supported software radio peripherals attached by USB and display something similar to the following.

```
$ sudo uhd_find_devices
linux; GNU C++ version 4.8.4; Boost_105400; UHD_003.010.000.HEAD-0-g6e1ac3fc

--------------------------------------------------
-- UHD Device 0
--------------------------------------------------
Device Address:
   type: usrp2
   addr: 192.168.10.2
   name:
   serial: xxxxxx
```

Furthermore, *sudo uhd_usrp_probe* command should report detailed information on UHD-supported software radio peripherals attached by USB and display something similar to the following. Details include unit names, revision numbers, and available sensors on all attached USRP motherboards and daughterboards.

```
$ uhd_usrp_probe
linux; GNU C++ version 4.8.4; Boost_105400; UHD_003.010.000.HEAD-0-g6e1ac3fc

-- Opening a USRP2/N-Series device...
-- Current recv frame size: 1472 bytes
-- Current send frame size: 1472 bytes
 _____________________________________________________
/
|       Device: USRP2 / N-Series Device
|     _____________________________________________________
/
|   |       Mboard: N210r4
|   |   hardware: 2577
|   |   mac-addr: 00:00:00:00:00:00
|   |   ip-addr: 192.168.10.2
|   |   subnet: 255.255.255.255
|   |   gateway: 255.255.255.255
|   |   gpsdo: none
|   |   serial: xxxxxx
|   |   FW Version: 12.4
|   |   FPGA Version: 11.1
|   |
|   |   Time sources:  none, external, _external_, mimo
|   |   Clock sources: internal, external, mimo
|   |   Sensors: mimo_locked, ref_locked
|   |     _____________________________________________________
|   |    /
|   |   |       RX DSP: 0
|   |   |
|   |   |   Freq range: -50.000 to 50.000 MHz
|   |     _____________________________________________________
|   |    /
|   |   |       RX DSP: 1
|   |   |
|   |   |   Freq range: -50.000 to 50.000 MHz
|   |     _____________________________________________________
|   |    /
|   |   |       RX Dboard: A
|   |   |   ID: WBX, WBX + Simple GDB (0x0053)
|   |   |   Serial: xxxxxx
```

```
|   |   |   ____________________________________________________
|   |   |  /
|   |   |  |      RX Frontend: 0
|   |   |  |  Name: WBXv2 RX+GDB
|   |   |  |  Antennas: TX/RX, RX2, CAL
|   |   |  |  Sensors: lo_locked
|   |   |  |  Freq range: 68.750 to 2200.000 MHz
|   |   |  |  Gain range PGA0: 0.0 to 31.5 step 0.5 dB
|   |   |  |  Bandwidth range: 40000000.0 to 40000000.0 step 0.0 Hz
|   |   |  |  Connection Type: IQ
|   |   |  |  Uses LO offset: No
|   |   |   ____________________________________________________
|   |   |  /
|   |   |  |      RX Codec: A
|   |   |  |  Name: ads62p44
|   |   |  |  Gain range digital: 0.0 to 6.0 step 0.5 dB
|   |   |  |  Gain range fine: 0.0 to 0.5 step 0.1 dB
|   |    ____________________________________________________
|   |  /
|   |  |      TX DSP: 0
|   |  |
|   |  |  Freq range: -50.000 to 50.000 MHz
|   |    ____________________________________________________
|   |  /
|   |  |      TX Dboard: A
|   |  |  ID: WBX (0x0052)
|   |  |  Serial: xxxxxx
|   |  |   ____________________________________________________
|   |  |  /
|   |  |  |      TX Frontend: 0
|   |  |  |  Name: WBXv2 TX+GDB
|   |  |  |  Antennas: TX/RX, CAL
|   |  |  |  Sensors: lo_locked
|   |  |  |  Freq range: 68.750 to 2200.000 MHz
|   |  |  |  Gain range PGA0: 0.0 to 25.0 step 0.1 dB
|   |  |  |  Bandwidth range: 40000000.0 to 40000000.0 step 0.0 Hz
|   |  |  |  Connection Type: IQ
|   |  |  |  Uses LO offset: No
|   |  |   ____________________________________________________
|   |  |  /
```

```
|   |   |   |      TX Codec: A
|   |   |   |   Name: ad9777
|   |   |   |   Gain Elements: None
```

The following command displays the firewall settings.

```
sudo iptables -L
```

Appendix A.7 Additional UHD Utilities

The `sudo uhd_images_downloader` command downloads FPGA images for current UHD version.

```
$ sudo uhd_images_downloader
Images destination:      /usr/local/share/uhd/images
Downloading images from: http://files.ettus.com/binaries/images/uhd-
images_003.010.000.000-release.zip
Downloading images to:   /tmp/tmpPYd_5J/uhd-images_003.010.000.000-release.zip
58959 kB / 58959 kB (100%)

Images successfully installed to: /usr/local/share/uhd/images
```

Appendix A.8 Thread priority scheduling

When UHD spawns a new thread, it may try to boost the thread's scheduling priority. If setting the new priority fails, the UHD software prints a warning to the console, as shown below. This warning is harmless; it simply means that the thread will retain a normal or default scheduling priority.

```
UHD Warning:
   Unable to set the thread priority. Performance may be negatively affected.
   Please see the general application notes in the manual for instructions.
   EnvironmentError: OSError: error in pthread_setschedparam
```

To address this issue, non-privileged (non-root) users need to be given special permission to change the scheduling priority. To enable this, add the line below to the file `/etc/security/limits.conf`.

```
@GROUP   - rtprio   99
```

Here, replace `GROUP` with a group in which the user is a member. Sometimes it is required to log out and log back into the account for the settings to take effect. In most Linux distributions, a list of groups and group members can be found in the `/etc/group` file.

Appendix A.9 Verifying the Operation of the USRP Using UHD and GNU Radio

This section explains the use of UHD and GNU Radio to verify the correct operation of the USRP[77]. On Linux, the default installation prefix location is `/usr/local`. The example programs are located in the `/usr/local/lib/uhd/examples` folder.

Appendix A.9.1 Benchmarking the system

To check the benchmarks interface with device, run `./benchmark_rate --rx_rate 10e6 --tx_rate 10e6`. Below shows the example output from `benchmark_rate`:

```
$ ./benchmark_rate --rx_rate 10e6 --tx_rate 10e6
linux; GNU C++ version 4.8.4; Boost_105400; UHD_003.010.000.HEAD-0-g6e1ac3fc


Creating the usrp device with: ...
-- Opening a USRP2/N-Series device...
-- Current recv frame size: 1472 bytes
-- Current send frame size: 1472 bytes
Using Device: Single USRP:
  Device: USRP2 / N-Series Device
  Mboard 0: N210r4
  RX Channel: 0
    RX DSP: 0
    RX Dboard: A
    RX Subdev: WBXv2 RX+GDB
  TX Channel: 0
    TX DSP: 0
    TX Dboard: A
    TX Subdev: WBXv2 TX+GDB

Setting device timestamp to 0...
Testing receive rate 10.000000 Msps on 1 channels
Testing transmit rate 10.000000 Msps on 1 channels

Benchmark rate summary:
  Num received samples:    100104043
  Num dropped samples:     0
  Num overflows detected:  0
  Num transmitted samples: 100229019
  Num sequence errors:     0
  Num underflows detected: 0
Num late commands:       0
  Num timeouts:          0


Done!
```

Appendix A.9.2 Receiving Samples

Run `./rx_samples_to_file --freq 98e6 --rate 5e6 --gain 20 --duration 10 usrp_samples.dat` to save samples to a file. Following is the example output from `rx_samples_to_file`:

```
$ /usr/local/lib/uhd/examples/rx_samples_to_file --freq 98e6 --rate 5e6 --gain 20 --duration 10
usrp_samples.dat
linux; GNU C++ version 4.8.4; Boost_105400; UHD_003.010.000.HEAD-0-g6e1ac3fc


Creating the usrp device with: ...
-- Opening a USRP2/N-Series device...
-- Current recv frame size: 1472 bytes
-- Current send frame size: 1472 bytes
Using Device: Single USRP:
  Device: USRP2 / N-Series Device
  Mboard 0: N210r4
  RX Channel: 0
    RX DSP: 0
    RX Dboard: A
    RX Subdev: WBXv2 RX+GDB
  TX Channel: 0
    TX DSP: 0
    TX Dboard: A
    TX Subdev: WBXv2 TX+GDB

Setting RX Rate: 5.000000 Msps...
Actual RX Rate: 5.000000 Msps...

Setting RX Freq: 98.000000 MHz...
Actual RX Freq: 98.000000 MHz...

Setting RX Gain: 20.000000 dB...
Actual RX Gain: 20.000000 dB...

Waiting for "lo_locked": ++++++++++ locked.

Press Ctrl + C to stop streaming...

Done!
```

Following is the example of file size output from *rx_samples_to_file*:

```
$ ls -al usrp_samples.dat
-rw-rw-r-- 1 user user 200040000 Sep  1 14:43 usrp_samples.dat
```

Appendix A.9.3 Transmitting Samples

Run *./tx_samples_from_file --freq 915e6 --rate 5e6 --gain 10 usrp_samples.dat* to transmit samples from file. Below shows the example of output from *tx_samples_from_file*:

```
$ /usr/local/lib/uhd/examples/tx_samples_from_file --freq 915e6 --rate 5e6 --gain 10
usrp_samples.dat
linux; GNU C++ version 4.8.4; Boost_105400; UHD_003.010.000.HEAD-0-g6e1ac3fc

Creating the usrp device with: ...
-- Opening a USRP2/N-Series device...
-- Current recv frame size: 1472 bytes
-- Current send frame size: 1472 bytes
Using Device: Single USRP:
  Device: USRP2 / N-Series Device
  Mboard 0: N210r4
  RX Channel: 0
    RX DSP: 0
    RX Dboard: A
    RX Subdev: WBXv2 RX+GDB
  TX Channel: 0
    TX DSP: 0
    TX Dboard: A
    TX Subdev: WBXv2 TX+GDB

Setting TX Rate: 5.000000 Msps...
Actual TX Rate: 5.000000 Msps...

Setting TX Freq: 915.000000 MHz...
Actual TX Freq: 915.000000 MHz...
```

```
Setting TX Gain: 10.000000 dB...
Actual TX Gain: 10.000000 dB...

Checking TX: LO: locked ...

Done!
```

Appendix A.9.4 Terminal DFT/FFT

To create ASCII/Ncurses FFT, run `./rx_ascii_art_dft --freq 98e6 --rate 5e6 --gain 20 --bw 5e6 --ref-lvl -30`. Following is the example output from `rx_ascii_art_dft`:

```
$ ./rx_ascii_art_dft --freq 98e6 --rate 5e6 --gain 20 --bw 5e6 --ref-lvl -30
linux; GNU C++ version 4.8.4; Boost_105400; UHD_003.010.000.HEAD-0-g6e1ac3fc


Creating the usrp device with: ...
-- Opening a USRP2/N-Series device...
-- Current recv frame size: 1472 bytes
-- Current send frame size: 1472 bytes
Using Device: Single USRP:
  Device: USRP2 / N-Series Device
  Mboard 0: N210r4
  RX Channel: 0
    RX DSP: 0
    RX Dboard: A
    RX Subdev: WBXv2 RX+GDB
  TX Channel: 0
    TX DSP: 0
    TX Dboard: A
    TX Subdev: WBXv2 TX+GDB

Setting RX Rate: 5.000000 Msps...
Actual RX Rate: 5.000000 Msps...

Setting RX Freq: 98.000000 MHz...
Actual RX Freq: 98.000000 MHz...

Setting RX Gain: 20.000000 dB...
Actual RX Gain: 20.000000 dB...

Setting RX Bandwidth: 5.000000 MHz...
Actual RX Bandwidth: 5.000000 MHz...

Checking RX: LO: locked ...
```

Figure A.1 shows the example screenshot of *rx_ascii_art_dft* running:

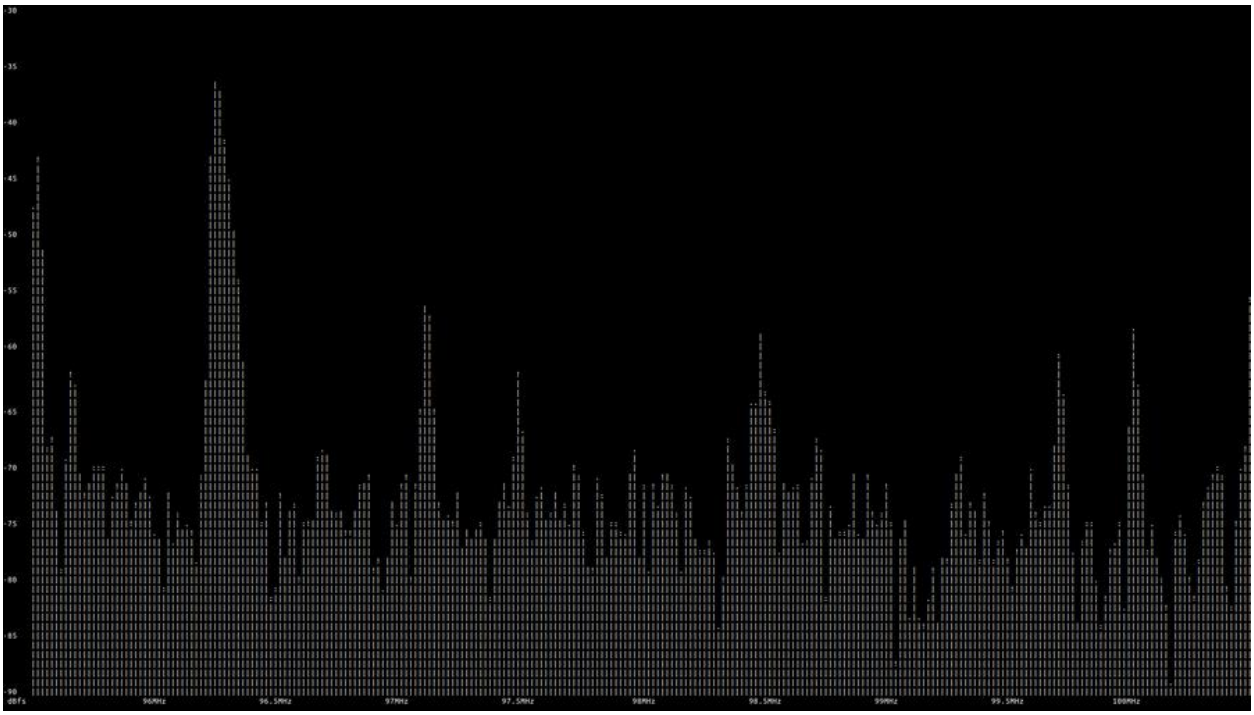


Figure A.1: Screenshot of running `rx_ascii_art_dft`

Appendix A.9.5 Transmiting test signal

To transmit specific waveform, run `./tx_waveforms --freq 915e6 --rate 5e6 --gain 0`. Below shows the example output from `tx_waveforms`:

```
$ ./tx_waveforms --freq 915e6 --rate 5e6 --gain 0
linux; GNU C++ version 4.8.4; Boost_105400; UHD_003.010.000.HEAD-0-g6e1ac3fc


Creating the usrp device with: ...
-- Opening a USRP2/N-Series device...
-- Current recv frame size: 1472 bytes
-- Current send frame size: 1472 bytes
Using Device: Single USRP:
  Device: USRP2 / N-Series Device
Mboard 0: N210r4
```

```
RX Channel: 0
    RX DSP: 0
    RX Dboard: A
    RX Subdev: WBXv2 RX+GDB
  TX Channel: 0
    TX DSP: 0
    TX Dboard: A
    TX Subdev: WBXv2 TX+GDB

Setting TX Rate: 5.000000 Msps...
Actual TX Rate: 5.000000 Msps...

Setting TX Freq: 915.000000 MHz...
Actual TX Freq: 915.000000 MHz...

Setting TX Gain: 0.000000 dB...
Actual TX Gain: 0.000000 dB...

Setting device timestamp to 0...
Checking TX: LO: locked ...
Press Ctrl + C to stop streaming...
^C
Done!
```

Appendix B OpenBTS Installation

This section explains the required steps to install OpenBTS.

Appendix B.1 Updating the System and Git Installation

The OpenBTS project utilizes several new features in Git. It is required to make sure the client is compatible (e.g. newer than 1.8.2). To do this, perform the following [78].

```
 $ sudo apt-get install software-properties-common python-software-properties
 $ sudo add-apt-repository ppa:git-core/ppa
 (press enter to continue)
 $ sudo apt-get update
 $ sudo apt-get install git ntp ntpdate bind9 resolvconf
Or, $ sudo apt install git ntp ntpdate bind9 resolvconf
```

After installing the Git, most of the remaining installation process is automated via scripts.

Appendix B.2 Getting the OpenBTS source code

From the command line in the fresh development environment, execute the following to download the most recent set of tools:

```
$ git clone https://github.com/RangeNetworks/dev.git
```

Before proceeding to the next stage, it is required to check the Git version as these tools require a modern version of Git (>1.8.2). If the instructions above in Git setup prerequisites are followed perfectly then it should have been good to go. To check the Git version, run the following command:

```
 $ git –version
 git version 2.18.0
```

Now, to download all of the components simply run the *clone.sh* script.

```
$ cd dev
$ ./clone.sh
```

Appendix B.3 Selecting a Branch or Tag

Before building, the desired branch or tag for compilation should be chosen using *switchto.sh*.

```
$ ./switchto.sh master
```

Appendix B.4 Installing required Libraries

To compile the various parts of OpenBTS (without the PBX) the following additional libraries, packages, and utilities will be needed. These can be installed (on a Debian-flavored Unix distro) with the following command:

```
sudo apt-get install autoconf libtool libosip2-dev libortp-dev libusb-1.0-0-dev g++ sqlite3
libsqlite3-dev erlang libreadline6-dev libncurses5-dev
```

liba53 will be needed to install, which is included with the distribution, the following commands should install it correctly from the *OPENBTS_ROOT*.

```
cd liba53
sudo make install
```

Appendix B.5 Building the OpenBTS code

The *build.sh* script will automatically install any build dependencies (build them manually if required) using *./build.sh radio-type (component-name)* command. For this project, the used radio-type is B200. After dependencies are taken care of, each component is compiled into an installable package.

```
$ cd ..
$ ./build.sh B200
```

Compiled packages will be in a new directory named `BUILDS/sometimestamp`. The time stamp format will be like `20XX-XX-XX--XX-XX-XX`.

Appendix B.6 Installing Packages

`dpkg` is used to install the fresh packages (this will complain about dependencies).

```
$ cd BUILDS/20XX-XX-XX--XX-XX-XX/
$ sudo dpkg -i *.deb
```

To have aptitude resolve the dependencies, execute the following:

```
$ sudo apt-get -f install
```

Appendix B.7 Installing OpenBTS scripts for systemd

Each component has an Upstart service definition for Ubuntu version lesser than Ubuntu 16.04. For Ubuntu 16.04 Upstart service will not work, therefore, systemd is required to install. To run OpenBTS, Asterisk, SipAuthServe and Smqueue services on Ubuntu 16.04, execute the following commands [79]:

```
$ cd
$ cd etc/systemd/system
$ sudo git clone https://github.com/nadiia-kotelnikova/openbts_systemd_scripts.git
$ sudo cp /etc/systemd/system/openbts_systemd_scripts/systemd/asterisk.service .
$ sudo cp /etc/systemd/system/openbts_systemd_scripts/systemd/openbts.service .
$ sudo cp /etc/systemd/system/openbts_systemd_scripts/systemd/sipauthserve.service .
$ sudo cp /etc/systemd/system/openbts_systemd_scripts/systemd/smqueue.service .
```

Appendix B.8 Configuring OpenBTS

After building OpenBTS, it is required to configure for running correctly. There are two key files that must be created for this to happen. *OpenBTS.db* is the database store for all OpenBTS configuration. It must be installed at */etc/OpenBTS*, which likely does not exist. So, to create this file, run the following command:

```
$ cd /etc/OpenBTS
$ sudo sqlite3 -init ./apps/OpenBTS.example.sql /etc/OpenBTS/OpenBTS.db ".quit"
```

To test this, run the following command. If a lot of configuration variables is displayed that means the database has been installed correctly.

```
sqlite3 /etc/OpenBTS/OpenBTS.db .dump
```

Appendix B.9 Running OpenBTS

At this point, a basic sanity check of OpenBTS should have been able to perform. In a new terminal/window run the following commands:

```
$ cd /OpenBTS
$ sudo ./OpenBTS
```

From here, a few OpenBTS configuration variables can be checked. Connect to OpenBTS with the *OpenBTSCLI* command in a new terminal/window:

```
$ cd /OpenBTS
$ sudo ./OpenBTSCLI
```

The *OpenBTSCLI* has several commands that can be used to interact with the software. Execute the following command within the *OpenBTSCLI*.

```
OpenBTS> config
```

A lot of stock configuration options will display. However, a few are required for basic operation. These are:

- **GSM.Radio.Band:** Set this to the GSM band appropriate for the hardware.
- **GSM.Radio.C0:** This is the Absolute Radio Frequency Channel Number (ARFCN). Set an appropriate value for the band.
- **Control.LUR.OpenRegistration:** Set this to a regular expression matching the IMSIs of the test phones. This tells OpenBTS not to reject the handset just because the registration server is not responding. Useful for debugging and initializing the system.

The values can be modified from *OpenBTSCLI* with the *config* command. For example, the following command will allow registrations from any phone regardless of their provider. With this configuration, all phones have access without any restriction. Anybody can connect to the network by using an attached antenna.

```
OpenBTS> config Control.LUR.OpenRegistration .*
```

Appendix B.9.1 Changing the Band and ARFCN

The first things that need to check are the radio band and ARFCN. The radio band is one of four values: 850 MHz, 900 MHz, 1800 MHz, or 1900 MHz, corresponding to the four GSM bands available around the world. An ARFCN is simply a pair of frequencies within the selected band that will be used for the transmission and reception of radio signals. Each radio band has over 100 different ARFCNs that can be used. ARFCN may also be referred to as the carrier (e.g., systems using multiple ARFCNs are multiple carrier systems). Choosing the correct band and ARFCN is important for regulatory reasons and to avoid interference with or from local carriers. OpenBTS *config* command is used to inspect the current band and ARFCN settings [80].

These configuration keys are in the *GSM.Radio* category. To view all configuration keys with the word *GSM.Radio* in their name, enter the following command:

```
OpenBTS> config GSM.Radio
GSM.Radio.ARFCNs 1          [default]
GSM.Radio.Band 900          [default]
GSM.Radio.C0 51             [default]
GSM.Radio.MaxExpectedDelaySpread 4          [default]
GSM.Radio.PowerManager.MaxAttenDB 10      [default]
GSM.Radio.PowerManager.MinAttenDB 0       [default]
GSM.Radio.RSSITarget -50   [default]
GSM.Radio.SNRTarget 10     [default]
```

The `GSM.Radio.Band` key shows that the 900 MHz band is being used and the `GSM.Radio.C0` key indicates that ARFCN is 51 in that band is currently selected.

If the radio hardware does not have limitations on or optimizations for a particular frequency, then proceed with these settings. An easy optimization for eliminating interference is to choose a band that is not used by other carriers in the country. In America, the used systems are 850 MHz and 1900 MHz while the rest of the world uses 900 MHz and 1800 MHz. Furthermore, it is required to choose a lower frequency to improve coverage with lower power. For this thesis project, the used GSM band was 900 MHz to avoid interference with the local carriers.

Appendix B.9.2 Ettus Research Radio Calibration

The proper value for `GSM.Radio.RxGain` needs to be adjusted for the Ettus Research equipment to work correctly. Otherwise, the signal being received will overdrive the demodulator. For this thesis project, the used `GSM.Radio.RxGain` was 10.

```
OpenBTS> devconfig GSM.Radio.RxGain 10
GSM.Radio.RxGain changed from "52" to "10"
GSM.Radio.RxGain is static; change takes effect on restart
```

Appendix B.9.3 Programming SIM card

Subscriber Identity Modules (SIMs) are trimmed-down smartcards. To program the SIMs, Smartcard writers are used. In general, smartcard writers come in two varieties[81]:

- **PC/SC devices:** These devices are interfaced through a Personal Computer/Smart Card (PC/SC) driver and do not appear in the `/dev` directory. An example device of this type is "Bluedrive II" available from Range.
- **USB-Serial devices:** In UNIX systems, these appear in `/dev` as serial port devices, like `/dev/ttyUSB0`. In this thesis project, the USB-Serial device is used for SIM programming.

pySim-prog is a small command line utility written in Python, which is used for programming various programmable SIM/USIM (Universal Subscriber Identity Module) cards. Such SIM/USIM cards are special cards, unlike those issued by regular commercial operators, which come with the kind of keys that allow writing the files/fields that normally only an operator can program. This is useful particularly for running own cellular network and issuing the customized SIM/USIM cards for that network[82].

For SIM card programming in this thesis project, SuperSIM 16-in-1 card and Identiv SCR3310v2.0 USB Smart Card Reader are used.

To install dependencies, run the following command on a new terminal/window:

```
sudo apt-get install pcscd pcsc-tools libccid libpcsclite-dev python-pyscard
```

Now, connect the SIM card reader to the computer with programmable SIM card inserted on it. To check the status of the connection, enter the `pcsc_scan` command on the terminal. If SIM card reader is recognized, then it will display something similar to the below output:

```
$ pcsc_scan
 PC/SC device scanner
 V 1.4.25 (c) 2001-2011, Ludovic Rousseau ludovic.rousseau@free.fr
 Compiled with PC/SC lite version: 1.8.14
 Using reader plug'n play mechanism
 Scanning present readers...
 0: SCM Microsystems Inc. SCR 3310 [CCID Interface] 00 00
 Tue Oct 18 11:48:08 2018
 Reader 0: SCM Microsystems Inc. SCR 3310 [CCID Interface] 00 00
 Card state: Card inserted,
 ATR: 3B 99 18 00 11 88 22 33 44 55 66 77 60
 + TS = 3B --> Direct Convention
 + T0 = 99, Y(1): 1001, K: 9 (historical bytes)
  TA(1) = 18 --> Fi=372, Di=12, 31 cycles/ETU
  129032 bits/s at 4 MHz, fMax for Fi = 5 MHz => 161290 bits/s
  TD(1) = 00 --> Y(i+1) = 0000, Protocol T = 0
 -----
 + Historical bytes: 11 88 22 33 44 55 66 77 60
 Category indicator byte: 11 (proprietary format)
 Possibly identified card (using /usr/share/pcsc/smartcard_list.txt):
 3B  99  18  00  11  88  22  33  44  55  66  77  60
 sysmocom sysmoSIM-GR1
```

To exit from `pcsc_scan`, press `Ctrl+C`. To get the code of `pysim`, enter the below commands:

```
git clone git://git.osmocom.org/pysim pysim
cd pysim
```

To read the SIM card, run the any of the following commands:

```
./pySim-read.py -p0
or ./pySim-read.py -p1
```

If everything is done correctly, then the output should be something similar to the following:

```
$ ./pySim-read.py -p0
 Reading ...
 ICCID: 1791198229180000071
 IMSI: 001640000000071
 SMSP: ffffffffffffffffffffffffe1ffffffffffffffffffffffff0581005155f5ffffffffffff000000
 ACC: ffff
 MSISDN: Not available
 Done !
```

To program the SIM card, enter `./pySim-prog.py -help` to get overview of possible options. The similar result should appear:

```
$ ./pySim-prog.py -help
 Usage: pySim-prog.py [options]
 Options:
  -h, --help                    show this help message and exit
  -d DEV, --device=DEV          Serial Device for SIM access [default: /dev/ttyUSB0]
  -b BAUD, --baud=BAUD          Baudrate used for SIM access [default: 9600]
  -p PCSC, --pcsc-device=PCSC
                                Which PC/SC reader number for SIM access
  -t TYPE, --type=TYPE          Card type (user -t list to view) [default: auto]
  -a PIN_ADM, --pin-adm=PIN_ADM
                                ADM PIN used for provisioning (overwrites default)
  -e, --erase                   Erase beforehand [default: False]
  -S SOURCE, --source=SOURCE
                                Data Source[default: cmdline]
  -n NAME, --name=NAME          Operator name [default: Magic]
  -c CC, --country=CC           Country code [default: 1]
  -x MCC, --mcc=MCC             Mobile Country Code [default: 901]
  -y MNC, --mnc=MNC             Mobile Network Code [default: 55]
  -m SMSC, --smsc=SMSC          SMSP [default: '00 + country code + 5555']
  -M SMSP, --smsp=SMSP          Raw SMSP content in hex [default: auto from SMSC]
  -s ID, --iccid=ID             Integrated Circuit Card ID
  -i IMSI, --imsi=IMSI          International Mobile Subscriber Identity
  -k KI, --ki=KI                Ki (default is to randomize)
  -o OPC, --opc=OPC             OPC (default is to randomize)
  --op=OP                       Set OP to derive OPC from OP and KI
  --acc=ACC                     Set ACC bits (Access Control Code). not all card types
                                are supported
  -z STR, --secret=STR          Secret used for ICCID/IMSI autogen
  -j NUM, --num=NUM             Card # used for ICCID/IMSI autogen
  --batch                       Enable batch mode [default: False]
  --batch-state=FILE            Optional batch state file
  --read-csv=FILE               Read parameters from CSV file rather than command line
  --write-csv=FILE              Append generated parameters in CSV file
  --write-hlr=FILE              Append generated parameters to OpenBSC HLR sqlite3
  --dry-run                     Perform a 'dry run', don't actually program the card
```

The 16-in-1 SIM cards are intended for COMP128v1 based cloning and enable the user to aggregate up to 16 SIM card identities in a single card. Below example shows how to change the card's IMSI to 460003113237934 (option -i) and at the same time can specify a new set of -x MCC (Mobile Country Code), -y MNC (Mobile Network Code), -s ID (Integrated Circuit Card ID) , -o OPC and -k KI (Ki) values.

To program a SuperSIM 16-in-1 card, run `./pySim-prog.py -p 0 -x 470 -y 01 -i 460003113237934 -s 8988211000000110000 -o 398153093661279FB1FC74BE07059FEF -k 1D8B2562B992549F20D0F42113EAA6FA`. The similar output should appear:

```
$ ./pySim-prog.py -p 0 -x 470 -y 01 -i 460003113237934 -s 8988211000000110000 -o
398153093661279FB1FC74BE07059FEF -k 1D8B2562B992549F20D0F42113EAA6FA
Insert card now (or CTRL-C to cancel)
Autodetected card type fakemagicsim
Generated card parameters :
 > Name    : Magic
 > SMSP    : e1ffffffffffffffffffffffff0581005155f5ffffffffffff000000
 > ICCID   : 8988211000000110000
 > MCC/MNC : 470/1
 > IMSI    : 460003113237934
 > Ki      : 1D8B2562B992549F20D0F42113EAA6FA
 > OPC     : 398153093661279FB1FC74BE07059FEF
 > ACC     : None

Programming ...
Done !
```

Appendix B.9.4 Searching for the Network

Now it is time to use a handset to search for the newly created network. Each handset's menu is different but the item is usually similar to "Carrier Selection" or "Network Selection." The process for manually selecting a different carrier on Android is detailed below and also displayed in Figure B.1.

- Launch the "Settings" application from the Android menu system
- Select "More"
- Select "Mobile networks"
- Select "Network operators" This may or may not start a search. If it does not, select "Search networks"
- Once the search has finished, a list of available carrier networks will present

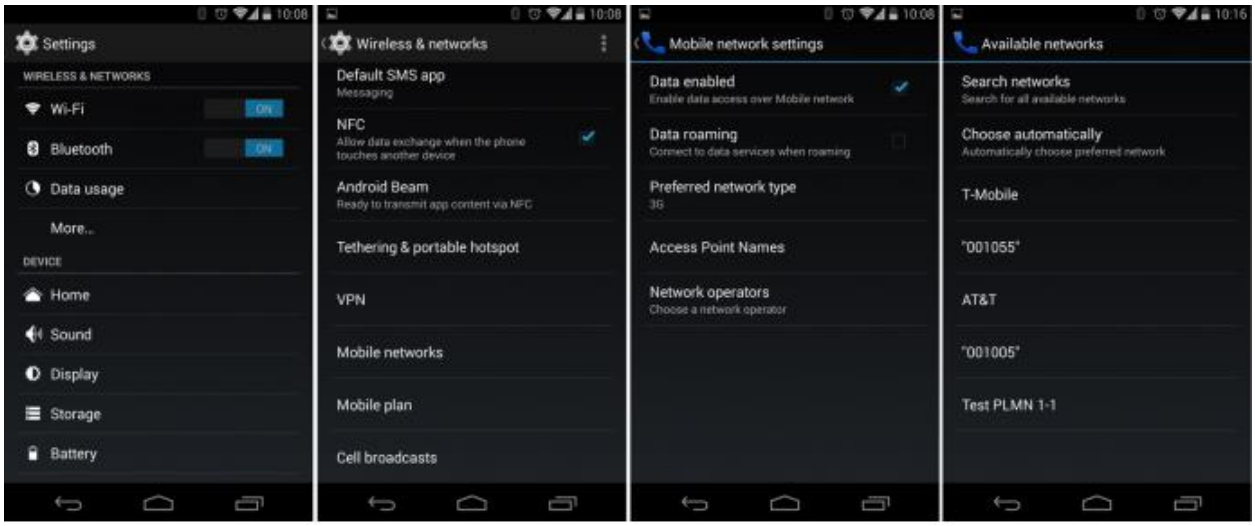


Figure B.1: Android Manual carrier selection[80]

Here, the test network is displayed in the list of selectable carriers. Depending on the handset model, firmware, and SIM used, the network ID will be displayed as “00101”, “001-01”, “Test PLMN 1-1”, or the GSM shortname of “OpenBTS”. Now, OpenBTS will not allow attaching with the network. This is because OpenBTS, by default, only allows registered handsets to connect. As the registration server (sipauthserve) is not running, no phones will camp. However, this verifies that the downlink is functional.

If the test network is not detected, force the search again by either reselecting the menu item, toggling airplane mode between on and off, or power cycling the handset. If that still does not work, confirm again that the handset supports the above-configured GSM band and that the baseband is unlocked (i.e. not restricted by contract to only using a specific carrier).

If these steps are complete that means BTS is working perfectly. A home location registrar (sipauthserve), short message service center (smqueue) and switch (PBX) is needed in order to register, send and receive SMS, and send and receive calls. The following sections are going to describe those steps.

Appendix B.10 Building and Installing the Subscriber Registry and Sipauthserve

OpenBTS depends on the installation of the SIP authorization server, *sipauthserver* for traffic registration. OpenBTS will not be a usable system without sipauthserver.

Appendix B.10.1 Subscriber Registry

To set up the subscriber registry database, the file path, where the db will reside in, must be created first. By default, this path is `/var/lib/asterisk/sqlite3dir`.

```
$ sudo mkdir -p /var/lib/asterisk/sqlite3dir
```

Appendix B.10.2 Sipauthserve

`sipauthserve` is an aptly-named daemon providing SIP authentication services. The `SIP.Proxy.Registration` config variable in `openbts` should point to its hostname and port. To build `sipauthserve`, run the following commands.

```
$ cd dev/subscriberRegistry
$ make
```

The above command will produce a `sipauthserve` executable. As with OpenBTS, `sipauthserve` will be needed to configure. `/etc/OpenBTS/` should be already exist.

```
$ sudo sqlite3 -init subscriberRegistry.example.sql /etc/OpenBTS/sipauthserve.db ".quit"
```

Appendix B.10.3 Running Sipauthserve

Running `sipauthserve` will provide a registration server. To do so, execute the following commands.

```
$ cd apps
$ sudo ./sipauthserve
```

`sipauthserve` does not have a Command-Line Interface (CLI), so a small output will be displayed only like the following:

```
ALERT 12935:12935 2018-09-13T00:06:44.8 sipauthserve.cpp:328:main: ./sipauthserve
(re)starting
```

If any of the configuration variables are changed, *sipauthserve* will be needed to restart for the changes to take effect.

Appendix B.11 Building and Installing Smqueue

Smqueue is the store-and-forward message service packaged with OpenBTS. Building and running are very similar to the process used for OpenBTS.

Appendix B.11.1 Building Smqueue

In the *dev/smqueue/* directory, run the following commands to build *smqueqe*.

```
$ cd dev/smqueue
$ autoreconf –i
$ ./configure
$ make
```

The above command will produce a *smqueue* executable in the *dev/smqueue/smqueue* directory.

Appendix B.11.2 Configuring Smqueue

Similar to OpenBTS, *smqueue* also depends on a configuration file located at */etc/OpenBTS/smqueue.db*. *smqueue* creates an empty, nonfunctional version of this database if the database is not available, but that will be useless. So to make a functional version of *smqueue*, need to do the same as did with OpenBTS and run the following command:

```
$ sudo sqlite3 -init smqueue/smqueue.example.sql /etc/OpenBTS/smqueue.db ".quit"
```

The above command will initialize */etc/OpenBTS/smqueue.db* with default values. These configuration variables should work without modification.

Appendix B.11.3 Running Smqueue

To run *smqueue* execute the following commands:

```
$ cd smqueue
$ sudo ./smqueue
```

smqueue does not have a CLI, instead just reading configuration values and processing messages. So a small output will be displayed only like the following:

```
ALERT 16478:16478 2018-09-13T00:29:33.7 smqueue.cpp:2798:main: smqueue (re)starting
smqueue logs to syslogd facility LOCAL7, so there's not much to see here
```

If any of the configuration variables are changed, *smqueue* will be needed to restart for the changes to take effect.

Appendix B.12 Building and Configuring Asterisk

There are three primary open-source PBX/Soft switches available. These are Asterisk, FreeSwitch, and Yate. The differences, tradeoffs, and advantages to using one system over the other are too numerous. However, the key point is that all are actively supported and used by the OpenBTS community. However, Asterisk is the standard OpenBTS PBX. It is the easiest to set up and most documented option though it is probably the most difficult.

Appendix B.12.1 Installing Standard Asterisk

For the simplest installation, the suggestion is to just install Asterisk from the main software repository. On Ubuntu, run the following command:

```
$ sudo apt-get install asterisk
```

Appendix B.12.2 Configuring Asterisk

Standard asterisk does not need any configuration to handle calls; they all come into an external context. For higher level-functions (e.g., routing), hooks may be needed to install into the subscriber registry.

Appendix B.12.3 Installing Asterisk Real-Time

The basic asterisk system is sufficient for simple testing of the OpenBTS system. However, it does not support secure connections, as all communications go through the external context. Similarly, Asterisk is unable to dynamically route calls without the subscriber registry. If any of these features are needed, then Asterisk Real-Time will be needed to install which pulls all of the needed information from a set of connected databases.

Appendix B.13 Running the whole system

Each component has a systemd service definition for Ubuntu 16.04. To start all the required services, execute the following commands in a separate terminal/window.

```
$ sudo systemctl start sipauthserve
$ sudo systemctl start smqueue
$ sudo systemctl start openbts
$ sudo systemctl start asterisk
```

Conversely, to stop them, run the following commands:\

```
$ sudo systemctl stop sipauthserve
$ sudo systemctl stop smqueue
$ sudo systemctl stop openbts
$ sudo systemctl stop asterisk
```

Appendix B.13.1 Exploring

To explore the OpenBTS, run the following commands in a new terminal:

```
$ cd /OpenBTS
$ ./OpenBTSCLI
OpenBTS Command Line Interface (CLI) utility
Copyright 2012, 2013, 2014 Range Networks, Inc.
Licensed under GPLv2.
Includes libreadline, GPLv2.
Connecting to 127.0.0.1:49300...
Remote Interface Ready.
Type:
"help" to see commands,
"version" for version information,
"notices" for licensing information,
"quit" to exit console interface.
OpenBTS> help            (list all commands available)
OpenBTS> audit           (check if your configuration is correct)
OpenBTS> config          (list all parameters)
OpenBTS> config XYZ      (list all parameters that contain XYZ)
OpenBTS> devconfig       (change developer and factory parameters)
OpenBTS> trxconfig       (view the factory radio calibration) [so far only on Range Networks
                         equipment]
OpenBTS> chans           (view the currently active channels)
OpenBTS> tmsis           (view all IMSIs that have interacted with the system)
OpenBTS> trans           (view all completed transactions like calls and sms)
OpenBTS> quit
```

Appendix B.13.2 Subscriber Registry Database

For subscriber registry database, run the following commands in a new terminal:

```
$ sudo sqlite3 /var/lib/asterisk/sqlite3dir/sqlite3.db
SQLite version 3.11.0 2016-02-15 17:29:24
Enter ".help" for usage hints.
sqlite> .tables
DIALDATA_TABLE RRLP SIP_BUDDIES rates
sqlite> select * from sip_buddies;
sqlite> select * from dialdata_table;
sqlite> .quit
```

Appendix C Testing the System

Appendix C.1 Test SMS

Now the handset has access to the network, it is ready to perform some tests. The first is a quick test of SMS capability of the network.

The component responsible for receiving, routing, and scheduling the delivery of SMS messages is SMQueue. It must be started before testing out these features; execute the following command to do so:

```
$ sudo systemctl start smqueue
```

Appendix C.1.1 Echo SMS (411)

On the handset, compose an SMS to the number 411. This is a *shortcode* handler in SMQueue that will simply echo back whatever it receives along with some additional information about the network and subscriber account that was used. The body of the message to 411 can be anything, although it can be useful to use unique content for each message or sequential numbers or letters. This helps to pinpoint which message is being responded to in case an error occurs.

Compose the message to 411. After sending the SMS, a reply should appear within a few seconds. Following is an example reply:

```
"1 queued, cell 0.1, IMSI901990000000018, phonenum 0000001, at Sep 8 02:30:59, Hi"
```

This indicates the following:

- There is one message queued for delivery.
- The base station has a load factor of 0.1.
- The message was received from IMSI 901990000000018, MSISDN 0000001.
- The message was sent on September 8 at 02:30:59.
- The message body was "Hi".

Appendix C.1.2 Direct SMS

SMS messages can also be tested directly from OpenBTS by using the *sendsms* command. From the OpenBTS CLI, run the *help sendsms* command to see how it is invoked.

```
OpenBTS> help sendsms
sendsms IMSI src# message... -- send direct SMS to IMSI on this BTS, addressed from source
number src#.
```

Messages are sent by specifying a target IMSI, the source number the message should appear to have originated from, and the message body itself. Substitute the information for the subscriber account to compose a message and then press Enter:

```
OpenBTS> sendsms 901990000000018 8675309 direct SMS test
message submitted for delivery
```

After a few seconds, the handset should display a new incoming message from the imaginary number 8675309 with a body of "*direct SMS test*".

SMS messages created in this way do not route through SMQueue at all; they are sent directly out through the GSM air interface to the handset and, as such, cannot be rescheduled. If the handset is offline or unreachable, these messages are simply lost. This is why SMQueue is needed to attempt and reschedule deliveries in an inherently unpredictable wireless environment.

Appendix C.1.3 Two-Party SMS

If more than one handset is configured for use in the network, send a few messages back and forth between them. It is required to verify that the source numbers are correct when receiving messages and that replies to these messages are routed back to the original sender.

Appendix C.2 Test Calls

The other service to test is the voice. As with SMS, OpenBTS does not directly handle voice and requires an additional service (Asterisk) to be run. First, start Asterisk:

```
$ sudo systemctl start asterisk
```

Using the same handset that used in the SMS tests, now verify a few aspects of the voice service. This is accomplished by utilizing a few test extensions that the `range-asterisk-configs` package defines. An extension is an internal phone number, unreachable from the outside.

Appendix C.2.1 Test Tone Call (2602)

The first used test extension is playing back a constant tone. This might not sound too exciting but does confirm many things about the network. Those are:

- Asterisk is running and reachable
- Call routing is working as expected
- Downlink audio is functional

By calling to *2602* with the handset, listen to the tone and listen for changes in pitch. These changes in pitch are due to missing information in the downlink voice stream path, similar to packet loss. In the field, this is the primary use for the test tone extension: testing downlink quality. A downlink loss of *3*% is normal in production networks, with losses of *5*%–*7*% still providing an understandable conversation.

Appendix C.2.2 Echo Call (2600)

The next test extension creates an *echo call*. Basically, all audio that Asterisk receives will be immediately echoed back to the sender. In addition to confirming the items listed for the test tone call, the echo call will reveal any delay or uplink quality issues present in the network.

By calling to *2600* with the handset, speak into the microphone. One should hear themselves very shortly afterward in the earpiece. A little delay is normal, but longer delays lead to an experience more like using a walkie-talkie. The human brain can deal with delays up to about 200ms without trouble. Beyond that, the conversation starts to break down and becomes uncomfortable.

Appendix C.2.3 Two-Party Call

If more than one handset is configured for use in the network, place some calls between them. It is required to verify that the source numbers are correct when receiving a call.

Appendix D Installation of Communications Toolbox Support Package for USRP Radio in MATLAB for each sensor

Following steps are required to install the Communications Toolbox Support Package for USRP Radio in MATLAB[83].

- On the MATLAB *Home* tab, in the *Environment* section, click *Add-Ons* > *Get Hardware Support Packages*
- In Add-On Explorer, browse or search for the Communications Toolbox Support Package for USRP Radio.
- Select the support package, and then click *Install*
- The support package installer prompts while it installs drivers needed for the USRP Radio software.

Appendix E Synchronize Time on the Network

The Network Time Protocol (NTP) has long been the king of time-setting software. The problem of synchronizing the networks is solved by using NTP to go out on the internet to get time from a public internet time server. But, this approach is prone to problems because: sometimes NTP transmission is blocked in the network. Therefore, *htpdate* is used to syncs time over http protocol. Here accuracy will be within 0.5 secs[84], according to the main page. To install *htpdate* run the following commands in a new terminal.

```
sudo apt-get install htpdate
sudo htpdate -a google.com
```

htpdate service will start after installing the package and time will be updated immediately if there is an internet connection.