Emergency Stations in the Grand Mosque of Mecca Using Wireless Sensor Network (WSN)

A Thesis Submitted in fulfilment of the requirement for the Degree of

Master of Engineering

By

MOHAMMED AMER AL NIZARI

MASSEY UNIVERSITY

School of Engineering and Advanced Technology (SEAT)

Massey University
Palmerston North,
September 2011

Emergency Stations in the Grand Mosque of Mecca Using Wireless Sensor Network (WSN)

MOHAMMED AMER AL NIZARI

2011

Abstract

The Hajj is one of the five pillars of Islam. Every year Muslims from all over the world gather in the two Holy Mosques, Mecca and Medina, in the Kingdom of Saudi Arabia to make the pilgrimage. The kingdom of Saudi Arabia has therefore invested heavily over the years in the security and emergency services for the comfort of pilgrims. While it is a great spiritual experience for all the pilgrims, at the same time it poses a range of series challenges to the authorities responsible for facilitating the Hajj. Security and emergency issues cause most difficulties and challenges.

Today, there are more than 2.5 million pilgrims with different languages, different ages and level of education gathering in a particular place at a specific time. A significant number of pilgrims die due to both accidents and natural causes and a large number get lost in this extremely crowded gathering. In fact, it is very common for some pilgrims to lose contact with their groups or friends during the rituals and this situation may be critical for certain pilgrims such as women children, the elderly, the sick, etc.

The use of recent technologies in the Hajj season could help in monitoring and tracking the pilgrims. Specifically these technologies could help in congestion, tracking missing pilgrims, discovering who is walking in reverse direction at peak times, re-detection etc. technologies may be found in Wireless Sensor Networks (WSNs). However; Wireless Sensor Networks (WSNs) are not easy to implement and some of them are costly.

This project to make the mission much easier, we made Wireless Sensors Network Stations as emergency fixed stations. These stations will be spread around the holy mosque to support local rescuers and aid the retrieval of missing pilgrims. Each emergency station has a button switch to press if the pilgrims get lost or if they need to request services. The range of the sensor, power consumption and the price are important in choosing the sensor. To meet these criteria we have used RF Engines from Synapse there have a range of up to 5 Km, and the lowest power consumption (IEEE 802.15.4 module running 2.4GHz Frequency, Up to 250 Kbps Data Rate).

Acknowledgments

Firstly I would like to thank, Professor Dr. Subhas Mukhopadhyay, for giving me an opportunity to do my Masters under his supervision. His wisdom, knowledge and continued support has always inspired and motivated me. Without his help, advice and expert guidance, this work would not have been possible, and, above all, I thank him for his technical and emotional support.

I extend my sincere thanks to the Custodian of the Two Holy Mosques King Abdullah bin Abdulaziz, for the support he gives to the scholarship program. King Abdullah provided an opportunity for me and my brothers to get the best level of education.

I will never forget the efforts of the Ministry of Higher Education in the Kingdom of Saudi Arabia, represented in the Saudi Cultural Mission in New Zealand in facilitating the process of scholarship and supervision on this project.

Finally, words will not describe all of the sacrifices and emotional supports made by my parents, my wife and my children as well as by my brothers and sisters. Without them my success would have no meaning. Thank you from the depths of my heart.

Table of Contents

CHAPTER 5 Experiment Set- up and results

List of Figures

List of Tables

CHAPTER 1

Introduction

CHAPTER 1 Introduction

1.1 Mecca and Hajj

Mecca is the holiest city for all Muslims since it is the Islamic spiritual centre and the sacred city of Islam. It is called "Mother of cities" in the Qur'an. Mecca is located in the western part of the Kingdom of Saudi Arabia, 75 km east Red Sea, at an altitude of 300 meters, and in a valley surrounded by low mountains [1]. Figure 1.1

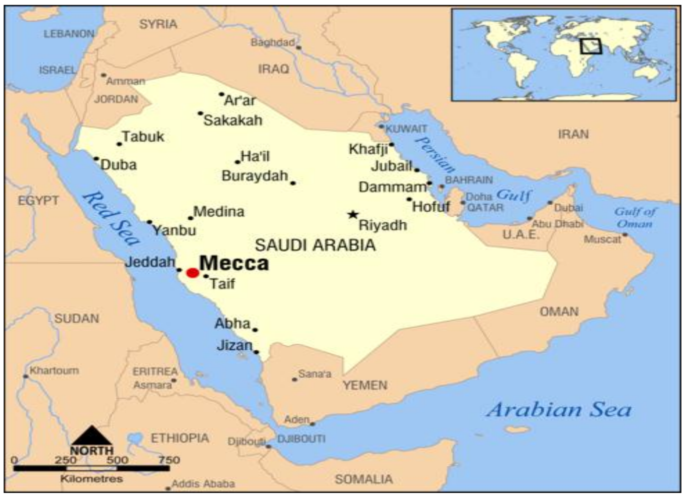


Figure1.1: Mecca location

Mecca has the largest mosque in the world, the Grand Mosque (Figure 1.2) which accommodates 1.2 million worshippers at a time [2]. The mosque covers an area of 356,800 square meters, with 32 entry and exit doors, including 4 main doors, as shown in Figure 1.3.

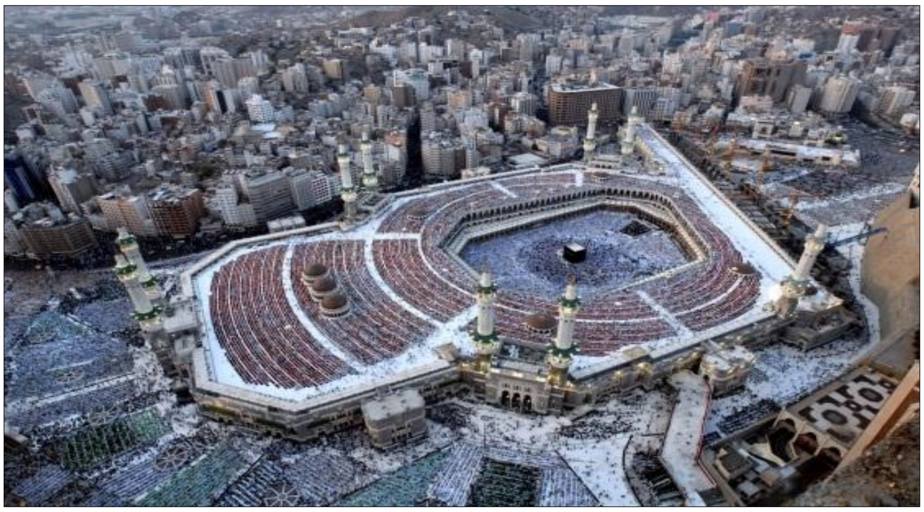


Figure 1.2: The Grand mosque

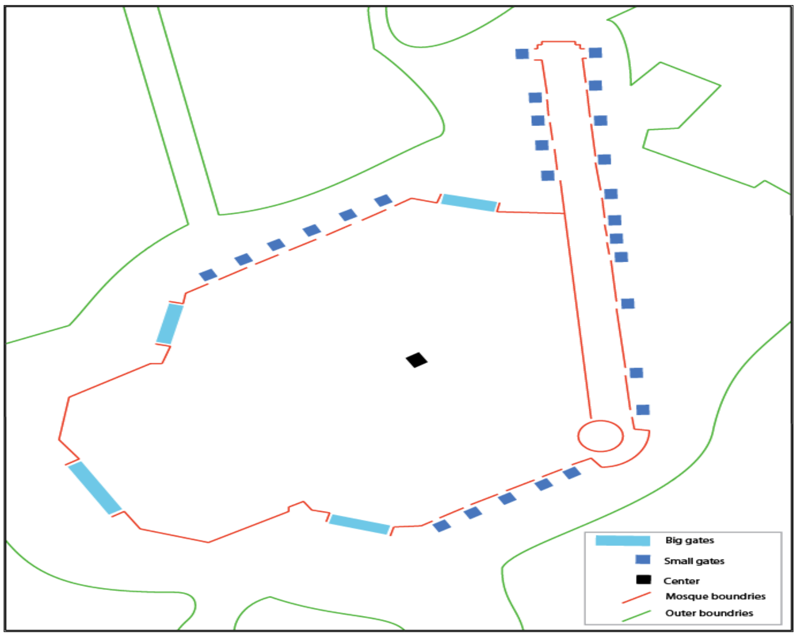


Figure 1.3: The Grand mosque doors

Two million people from over 100 countries assemble annually in Mecca to perform the Hajj ritual [3]. The Hajj is one of the five fundamental pillars of Islam. It is a set of acts of worship to be performed at least once in a lifetime. The Hajj is obligatory for every Muslim who satisfies certain conditions imposed by Allah, namely being healthy and financially able to do the Hajj.

The number of pilgrims is increasing dramatically every year, as shown in Figure 1.4. In the last ten years the numbers have increased from one million to about 2 million pilgrims. Hajj is considered as one of the largest and most long-standing annual mass gathering events on earth. This continued increase in the number of people has made the Hajj more difficult for both the pilgrims and the authorities in terms of safety and security.

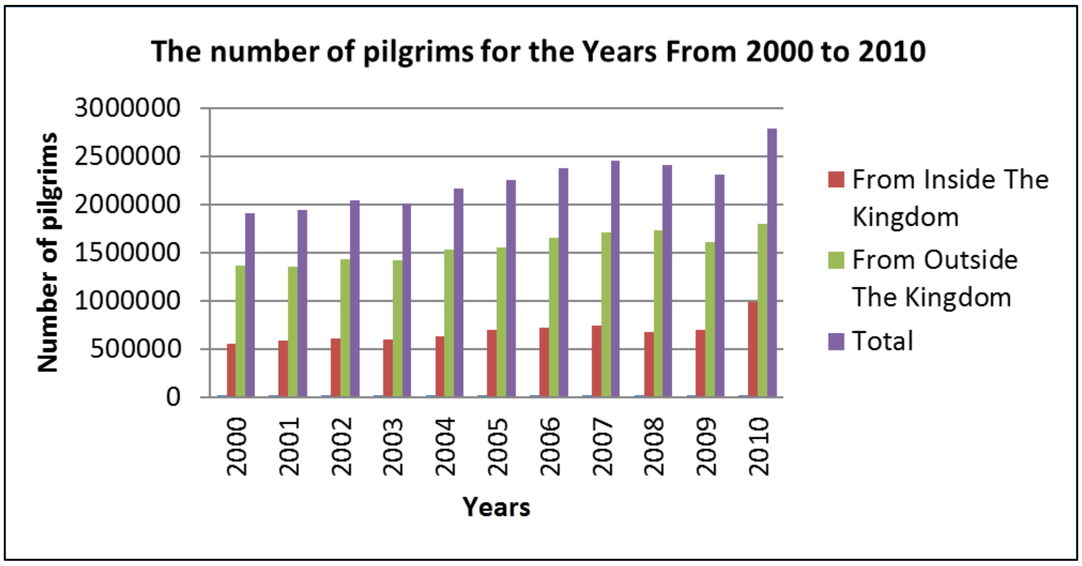


Figure 1.4: Number of pilgrims [4]

1.2 Problem definition

The exponential rise in the numbers of pilgrims attending the Hajj presents enormous challenges to the Saudi Arabian authorities. Their role as Custodians of the Holy Sites in Mecca is to provide extensive, multi-faceted services to these ‘Guests of God’. The authorities arrange free health care services, security services, crowd control, transportation and accommodation to ensure that all aspects of the pilgrimage rituals are conducted safely and without major incident throughout the days of Hajj [1]. However, with all the developments and expansions of service to the pilgrims at the Sacred Mosque congestion and communication between pilgrims and services providers remain the biggest challenges facing pilgrims and the authorities [5].

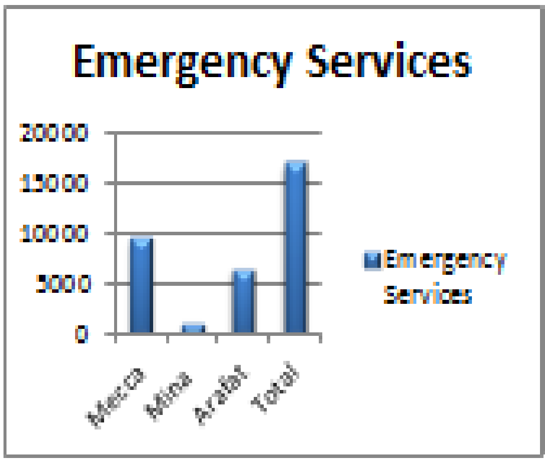
1.3 Problem statement

People get lost in Mecca and around the Grand Mosque every year [5]. The vast majority of people who get lost are usually children, women, old men, and especially those from overseas. These groups of pilgrims get lost when they are separated from their families while walking among the crowd. Different nationalities and languages make it difficult for some pilgrims to ask for help [26]. In most cases, people are generally lost for a few hours but sometimes may be lost for an entire day.

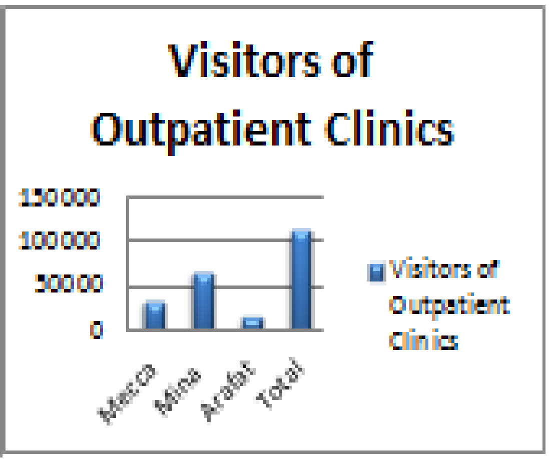
In addition, according to the Ministry of Health in 2010, nearly half a million pilgrims over the 15 day during Hajj period are provided with medical care. Figures 1.6 A, B, C, D, E, and F clearly present the numbers of pilgrims who are in need of medical and emergency help during the Hajj. Moreover, the total number of deaths among pilgrims registered in the same period in Mecca was 646 cases [8].

The current system is ineffective in saving the lives of the pilgrims who are lost and in need of medical help due to the size of the Mosque, and it takes a very long time to provide emergency services. So there is a need for an accurate system to communicate with the emergency services providers to address these problems.

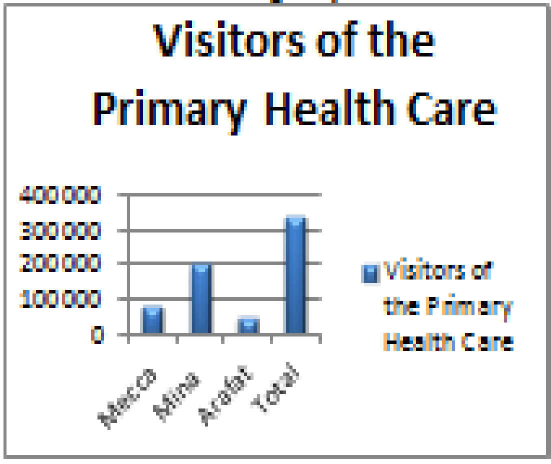
Figures 1.6: Use of Medical Services during 15 Day Period of Hajj 2010



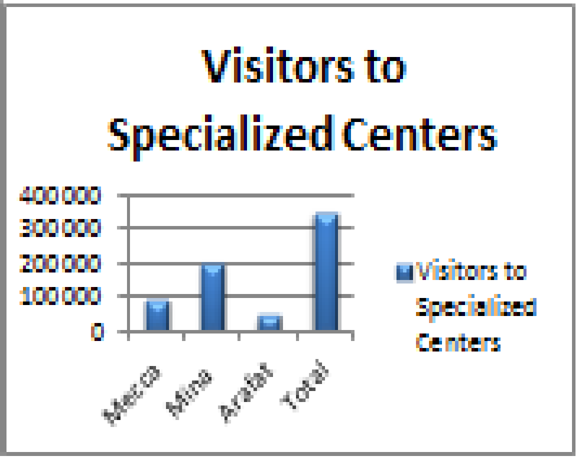
A: Emergency services.



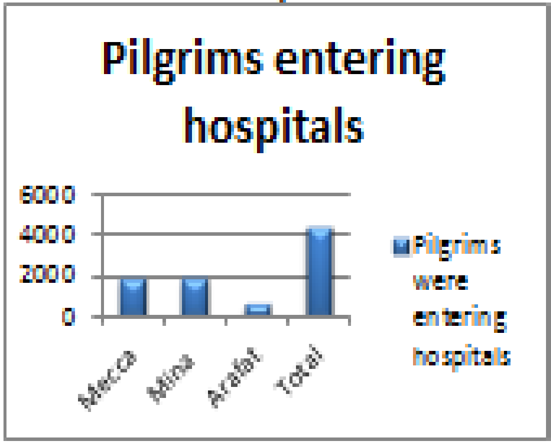
B: Visitors of Outpatient Clinics.



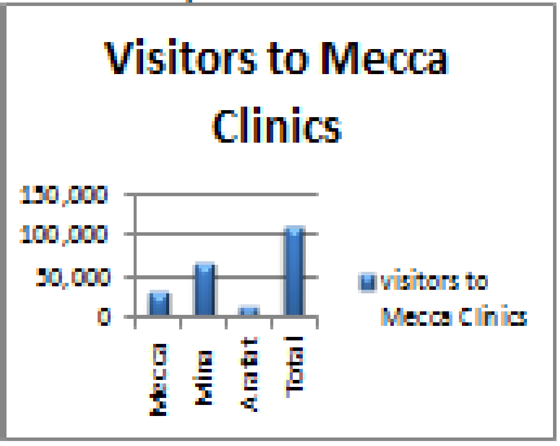
C: Primary Health Care.



D: Specialized Centers.



E: Pilgrims entering hospitals.



F: Visitors to Mecca Clinics.

1.4 The Objective

The objective of this project is to design a Wireless Sensor Network using sensor nodes based on the standard of IEEE 802.15.4 and RF technology, to respond to any emergency alert inside the Grand Mosque of Mecca and the areas surrounding the mosque. The design includes fixed stations containing two switches: one for medical help, and the other for the people who get lost. It is designed to help the pilgrims in need of medical assistance or security, for example, missing children, or elderly people during the Hajj time, and inform appropriate authorities in a timely and cost effective manner. The project further aims to enable faster and easier communication with the providers of the security services or the emergency medical services.

1.5 Thesis organization

The thesis is organized into eight chapters. After the introduction and the problem background, chapter II is a literature review of Wireless Sensor Network. Chapter III focuses on the design and specification of the system, showing the devices that have been used in this system. Then, Chapter IV will present a network simulation and results followed by experiments set-up and results in chapter V. Next, Chapter VI is a summary of finding and discussion. Chapter VII is a conclusion and recommendations.

CHAPTER 2

Literature Review of Wireless sensor network (WSN)

CHAPTER 2 Literature Review of Wireless sensor network (WSN)

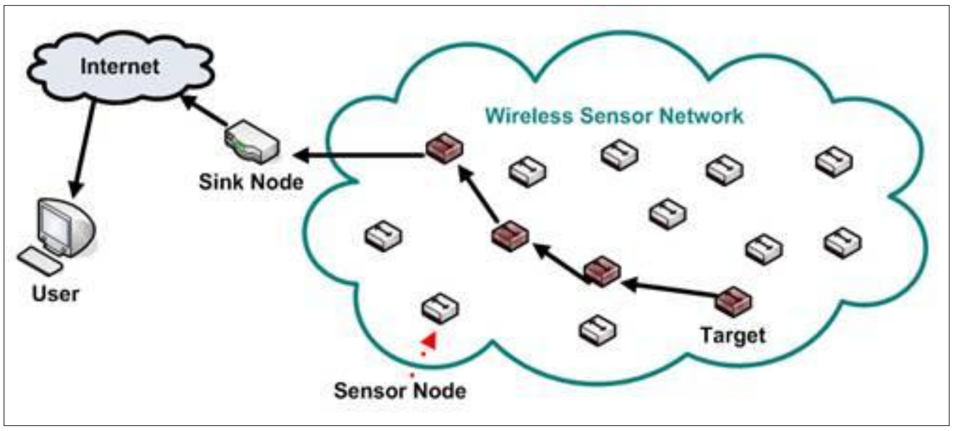


Figure: 2.1: Wireless sensor network

The wireless sensor network (WSN) is a collection of nodes organized into a cooperative network. The concept of WSN is that each node consists of the processing capability for one or more microcontrollers, CPUs or DSP chips. There may contain multiple types of memory program, data and flash memories. They have a power source e.g., batteries and solar cells; and accommodate various sensors and actuators [7]. WSN generally consists of a base station or gateway that can communicate with a number of wireless sensors via a radio link. Data is collected at the wireless sensor node, compressed, and transmitted to the gateway directly or, if required, uses other wireless sensor nodes to forward data to the gateway. The transmitted data is then presented to the system by the gateway connection. There are no limitations regarding the number of nodes anticipated, so there can be systems consisting of 1000 or even 10,000 nodes. The system can be used across numerous application areas [8].

A wide range of applications for WSN has been conducted in different areas such as health, military, factories, oil industries and home automation, entertainment, crisis management, and homeland defense.

2.1 Existing WSN technologies and applications

There are a number of existing technologies and protocols that have been used recently on the WSN. Bluetooth, UWB, Wi-Fi, and ZigBee protocols. There correspond to the IEEE 802.15.1, 802.15.3, 802.11a/b/g, and 802.15.4 standards, respectively. The IEEE defines only the PHY and MAC layers in its standards. [27]

2.1.1 Bluetooth or IEEE 802.15.1

Bluetooth, also known as IEEE 802.15.1 standard, is based on a system for wireless devices designed for short ranges [27]. It is inexpensive to replace the cables for computer peripherals, for example mice, keyboards, joysticks and printers. It defines a suite of applications, and wireless personal area network (WPAN) [9]

2.1.2 UWB or IEEE 802.15.3

UWB has attracted considerable attention in recent times as an indoor short-range high-speed wireless communications system [28]. It also acts as a replacement for cable wireless high speed serial bus such as USB 2.0 and IEEE 1394 [10]

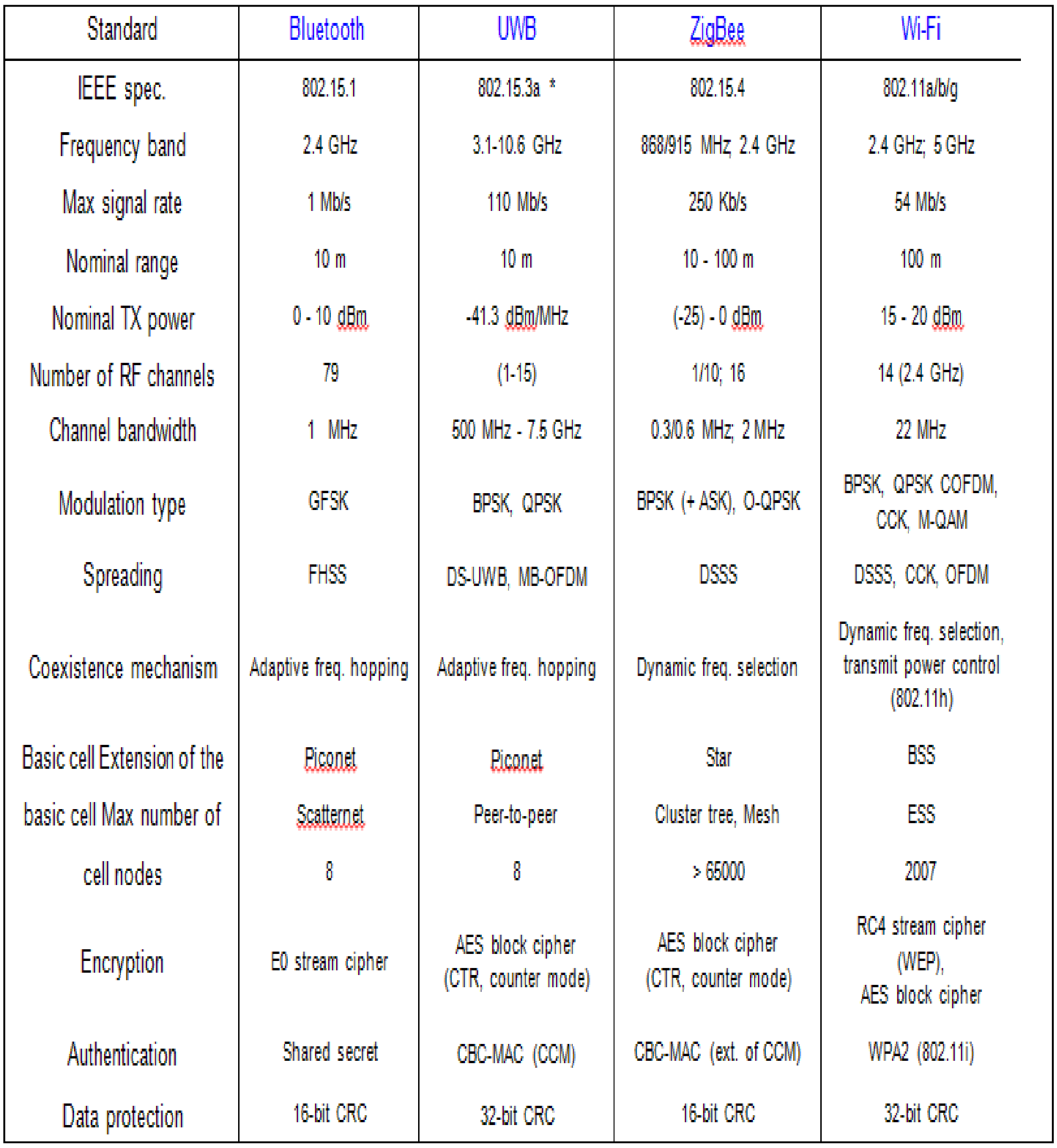
2.1.3 Wi-Fi or IEEE 802.11a/b/g

Wireless fidelity (Wi-Fi) and IEEE standards include as standard 802.11a/b/g for wireless local area networks (WLAN) [27]. Wi-Fi enables users to browse the Internet at broadband speeds when plugged to an access point (AFP) or in the custom mode [30] [10]

2.1.4 ZigBee or IEEE 802.15.4

ZigBee or IEEE 802.15.4 defines specifications for low-rate WPAN (LR-WPAN) for supporting simple devices that consume minimal power and typically operate in personal operating spaces (POS) of 10m. ZigBee can also reach 100m in some applications [11]. ZigBee provides self-organized, multi-hop, and reliable mesh networking with a long battery lifetime [31]. The Table below shows the major differences between the four protocols.

Table 2:1: COMPARISON OF THE BLUETOOTH, UWB, ZIGBEE, AND WI-FI PROTOCOLS [10]



Standard	Bluetooth	UWB	ZigBee	Wi-Fi
IEEE spec.	802.15.1	802.15.3a *	802.15.4	802.11a/b/g
Frequency band	2.4 GHz	3.1-10.6 GHz	868/915 MHz; 2.4 GHz	2.4 GHz; 5 GHz
Max signal rate	1 Mb/s	110 Mb/s	250 Kb/s	54 Mb/s
Nominal range	10 m	10 m	10 - 100 m	100 m
Nominal TX power	0 - 10 dBm	-41.3 dBm/MHz	(-25) - 0 dBm	15 - 20 dBm
Number of RF channels	79	(1-15)	1/10; 16	14 (2.4 GHz)
Channel bandwidth	1 MHz	500 MHz - 7.5 GHz	0.3/0.6 MHz; 2 MHz	22 MHz
Modulation type	GFSK	BPSK, QPSK	BPSK (+ ASK), O-QPSK	BPSK, QPSK COFDM, CCK, M-QAM
Spreading	FHSS	DS-UWB, MB-OFDM	DSSS	DSSS, CCK, OFDM
Coexistence mechanism	Adaptive freq. hopping	Adaptive freq. hopping	Dynamic freq. selection	Dynamic freq. selection, transmit power control (802.11h)
Basic cell Extension of the	Piconet	Piconet	Star	BSS
basic cell Max number of	Scatternet	Peer-to-peer	Cluster tree, Mesh	ESS
cell nodes	8	8	> 65000	2007
Encryption	E0 stream cipher	AES block cipher (CTR, counter mode)	AES block cipher (CTR, counter mode)	RC4 stream cipher (WEP), AES block cipher
Authentication	Shared secret	CBC-MAC (CCM)	CBC-MAC (ext. of CCM)	WPA2 (802.11i)
Data protection	16-bit CRC	32-bit CRC	16-bit CRC	32-bit CRC

2.2 Existing WSN Applications

This section provides a number of examples of WSN applications. First, examples of WSN applications in healthcare are provided to understand the utilization of WSN technology. Second, examples of WSN applications that have been conducted in Mecca are given.

2.2.1 Recent WSN applications in health care

The impact of wireless technology in healthcare has proved to be enormous and its usage is rapidly spreading [12]. Health care is probably the most demanding area in terms of WSN application and further use. The variety of opportunities provided by WSN in the area of remote patient care and monitoring systems cannot be overestimated [29]. The following examples of WSN applications in healthcare show how WSN technology may be used.

2.2.1.1 Monitoring System: Early Detection Of Alzheimer’s Disease

One of the recent WSN applications in the health care is found in “Monitoring System: Early Detection of Alzheimer’s disease”. In 2010, Ho Ting Cheng and Weihua Zhuang, from the University of Waterloo, conducted a study for facilitating the early detection of Alzheimer’s disease [13]. They proposed that e-healthcare solutions are expected to facilitate medical treatments, improve the quality of life of senior people, and reduce healthcare costs. By using a Bluetooth-enabled in-home patient monitoring system, they take advantage of short range Bluetooth communications for in-home patient location tracking. The location of information can then be recorded in a local database. With knowledge of the movement pattern of a patient a medical practitioner is more likely to be able to determine whether a target patient is developing Alzheimer’s disease or not. Early detection of Alzheimer's disease can promote the best treatment for patients, and preserve time and money to find a cure for Alzheimer's disease. The researchers also conducted a feasibility study, and their study shows that the proposed in-home patient monitoring system is feasible and can be applied in practice.

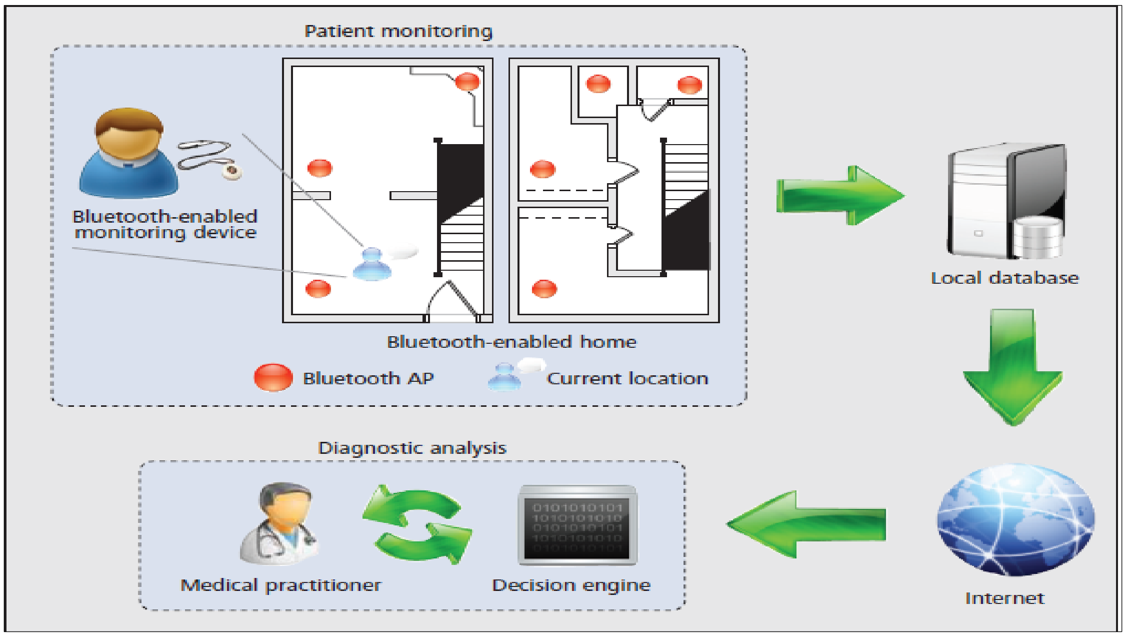


Figure 2.2: An illustration of the proposed Bluetooth-enabled in-home patient monitoring system

2.2.1.2 Wireless Patient Monitoring System

Recently in Turkey Radosveta Sokullu, Mustafa Alper Akkaú from Ege University, and Hüseyin Ertürk Çetin from Aselsan A.S. provided another WSN system example which has been successfully implemented in the health sector. This is called the "Wireless Patient Monitoring System" [12]. The project showed that WSN supported the medical staff and doctors in monitoring the patient's status continuously and remotely by reading oxygen saturation (%SPO2), sphygmo (pulse) and plethysmogram levels. The project also sets out the three main subsystems involved: wireless network structure, data measurement subsystem and the base station with its graphical interface, as shown in Figure 2.3.

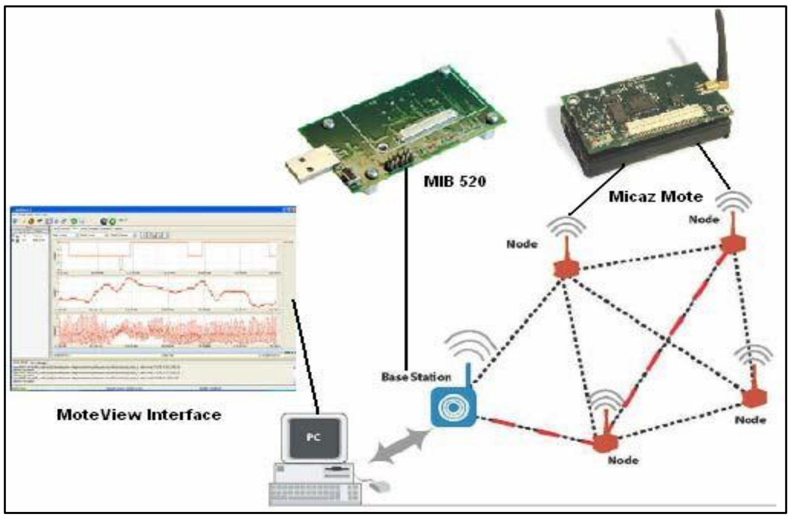


Figure 2.3: Remote Monitoring System

The patient's status readings are transmitted wirelessly from the patient through routing nodes to the base station. Then the base station is connected to a host computer running Mote View to explain, store and view the gathered data.

2.2.2 Existing WSN Project in Mecca and Hajj Season

There are also examples of WSN applications and projects which are conducted in Mecca and in the Hajj season. Recently, the numbers of WSN applications have been increasing gradually. Interesting examples of applications are shown below to develop the services that are provided in the Hajj season in Mecca include Group of Pilgrims Monitoring (GPM), and an RFID-Based Pilgrim Identification System.

2.2.2.1 Group of Pilgrims Monitoring (GPM)

In 2010, Moaad Al-Salman from Imam Muhammad Bin Saud Islamic University [18] aimed to develop the quality of services provided to the pilgrims. Al-Salman conducted this project, by using WSN technology to manage a group of pilgrims in the Hajj season. He had earlier studied the situation of the hajj extensively. His system provided a reasonable solution for tracking pilgrims in the Hajj using fixed wireless sensor nodes. The fixed nodes were placed in a consistent manner around the Grand Mosque and an algorithm was used for predicting the nearest anchor to the pilgrim. The system required a Pilgrim Node, Anchor node, Leader node, and Personal computer (PC).

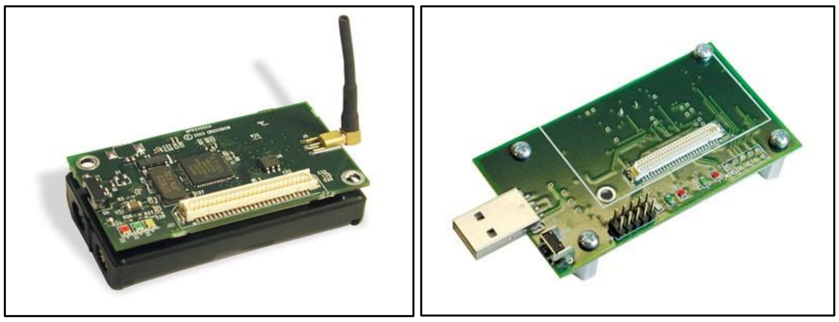


Figure 2.4: A base station and gateway board [18]

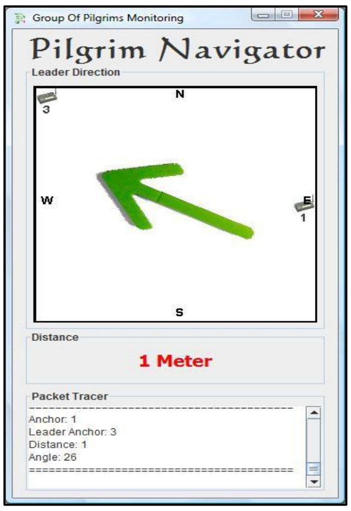
Al-Salman used an MICAz a 2.4 GHz, IEEE/ZigBee 802.15.4, as a base station and an MIB520 as a gateway board as shown in Figure 2.4. Both were programmed via Mote Works Software Platform which is fully compatible with the MICAz Module. The actual system is works through three key processes involving the following tools:

1. **Base Station Package:** BaseStation is a basic TinyOS utility application with some modification to read RSSI value. It acts as a bridge between the serial port and the radio network. When it receives a packet from the serial port, it transmits it on the radio; when it receives a packet over the radio, it transmits it to the serial port. Because TinyOS has a tool chain for generating and sending packets to a Mote over a serial port, using a BaseStation allows PC tools to communicate directly with Mote networks.

2. **Anchor Package**: This application listens to the radio channel. When it receives a Locate Message, the event received is signaled and the blue or yellow LED flashes. In Receive Event the anchor package creates a new packet with a payload equal to the Received Packet payload. Then the anchor field is set to anchor ID using TOS NODE ID and to store the RSSI value using CC2420Packet.get Rssi() method in the Rssi field. Finally, the packet is sent to the leader Mote (BaseStation) and the green LED flashes [18].

3. **Pilgrim Package:** This application has a timer which fires periodically (by default every second). When the timer fires it creates a packet (Locate Message) and does the following:

- Set pilgrim Id field to pilgrim ID using TOS NODE ID.
- Set counter field to current counter value.
- Set txp field to an appropriate TXP value (depends on current distance between the group leader and the pilgrim).



```
------------------------------------------------
Anchor: 1
Leader Anchor: 3
Distance: 1
Angle: 26
================================================
```

Figure 2.5: Pilgrim Navigator indicates [18]

Finally it sends the packet to the nearest anchor, and from there it will be forwarded to Java GUI (Pilgrim Navigator) over the serial port and the blue or yellow LED flashes, as shown in Figure 2.5. The screen of the Pilgrim Navigator indicates where the nearest anchor to the pilgrim is (anchor 1) and nearest anchor to his leader is (anchor 3). The arrow is pointing to anchor 3 and the distance between them is one meter. A special sound occurs when the distance between them increases.

2.2.2.2 An RFID-Based Pilgrim Identification System

King Fahd University of Petroleum Minerals in Saudi Arabia supported the project of Mohammed Mohandes, Maan Kousa and Ahmed A Hussain under the title of “An RFID-Based Pilgrim Identification System” [15]. This project provides a solution to help the Hajj authorities manage the overcrowding problem and the pilgrim identification, using RFID technology.

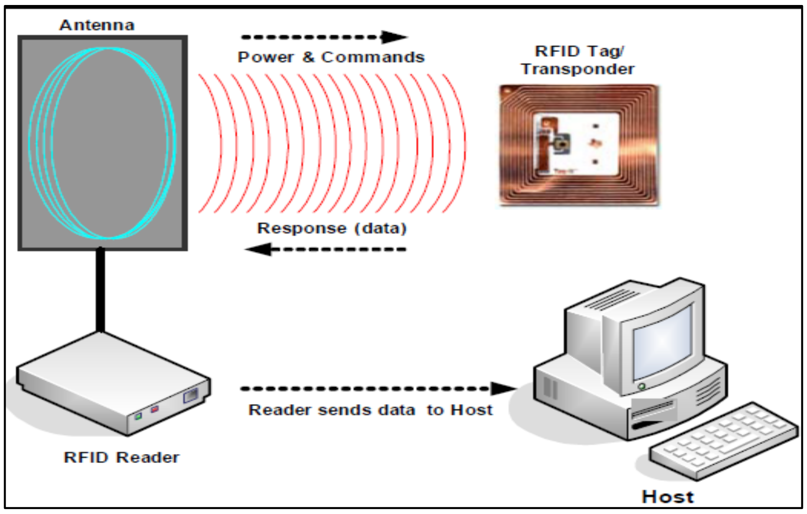


Figure 2.6: RFID-Based Pilgrim Identification System [15]

The system that became a prototype Pilgrim Identification System uses a wristband RFID tag, an RFID reader, and a Graphical User Interface application running on a PC. The Graphical User Interface communicates with an RFID system that consists of an RFID reader and a set of RFID wristband tags. The reader is used to read a unique ID number (UID) stored in the wristband tag which is then sent to the PC as shown in Figure 1.12.

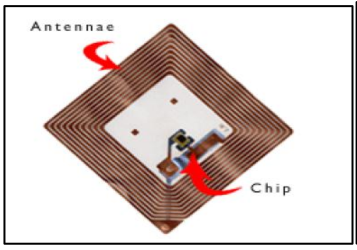


Figure 2.7: RFID tag [15 **Figure 2.8**: Wristband RFID tag [15]

As shown in Figure 1.14 a waistband RFID is carried by the pilgrims all the time during the Hajj, and includes the following data:

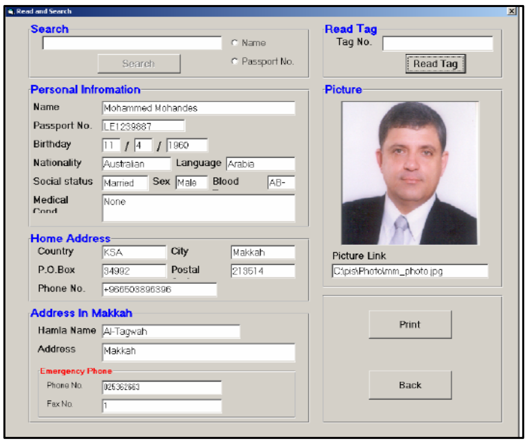
• Personal details like name, address, blood type, nationality, etc.

• Any Medical conditions.

• Contact information of the pilgrim's Hajj group.

• An E-purse that can be loaded with an optional amount of money.

If the pilgrims pass a Wristband RFID tag in front the RFID reader the information stored in the tag can be regained and presented on the GUI on the PC unit as shown in Figure 2.9.



Tag No.

Personal Infromation

Name	Mohammed Mohandes
Passport No.	LE1239887
Birthday	11 / 4 / 1960
Nationality	Australian
Language	Arabia
Social status	Married
Sex	Male
Blood	AB-
Medical Cond	None

Home Address

Country	KSA
City	Makkah
P.O.Box	34992
Postal	213514
Phone No.	+966503896396

Picture Link: C:\pis\Photo\mm_photo.jpg

Address In Makkah

Hamla Name	Al-Tagwah
Address	Makkah

Emergency Phone

Phone No.	025362663
Fax No.	1

Figure 2.9: GUI on the PC unit [19]

CHAPTER 3

Specification of the System and Design

CHAPTER 3 Specification of the System and Design

3.1 Introduction

This chapter will explore the fundamental properties of the project, including the main components that have been used. This involves the System structure, the Synapse RF Engine module, and the evaluation kit for the system. It also presents how the system is built and how it works, and the system programs and data that have been used. Various programs have been used in the project, including python programming language, C# language in correlation with the XML\RPC function library, Synapse Portal software, and SNAPconnect software, each with specific objectives for system operation.

3.2 System structure

The system contains three main components: Station nodes, a Bridge node and Graphical User Interface (GUI) as shown in Figure 3.1

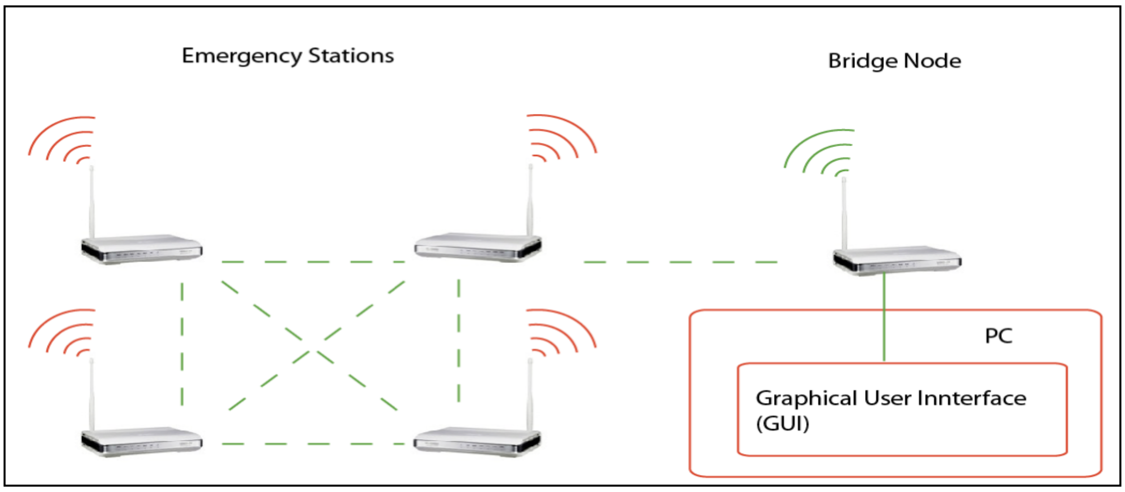


Figure 3.1: System structure

3.2.1 Station nodes

The station nodes are connected together wirelessly. These station nodes have two switches. When the switch is pressed, the bridge node will be informed wirelessly.

3.2.2 Bridge node

The bridge node is connected to the PC and it works as a receiver. When it receives a signal from the stations, it forwards it directly to the PC.

3.2.3 Graphical user interface (GUI)

The GUI takes the data received from the bridge node and displays the results on a friendly interface program.

3.3 RF Engine

The Synapse RF Engine module is the backbone of the system and it is an IEEE 802.15.4, low power, highly-reliable solution to embedded wireless control. The monitoring network and wireless mesh network operating system has an integrated transceiver radio data rate up to 2 Mbits/sec. The Synapse RF Engine module is a low-cost module and can have a range of up to 5 Km with a power consumption as low as 1.6 µA to enable a new generation of battery-driven systems [16]. The physical dimensions of the RF Engine are 33.86mm in width and 46.66mm in height as shown Figure 3.2.

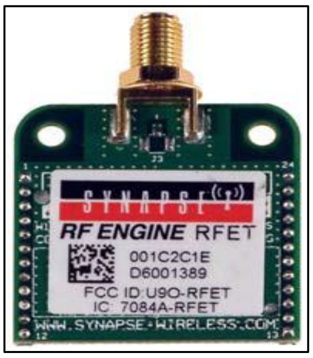


Figure 3.2: RF Engine

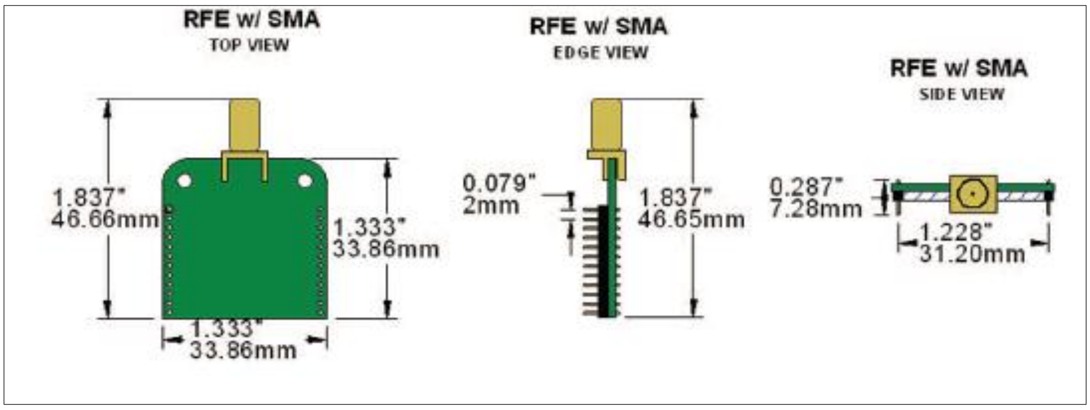


Figure 3.3: Physical Dimensions of RF Engine

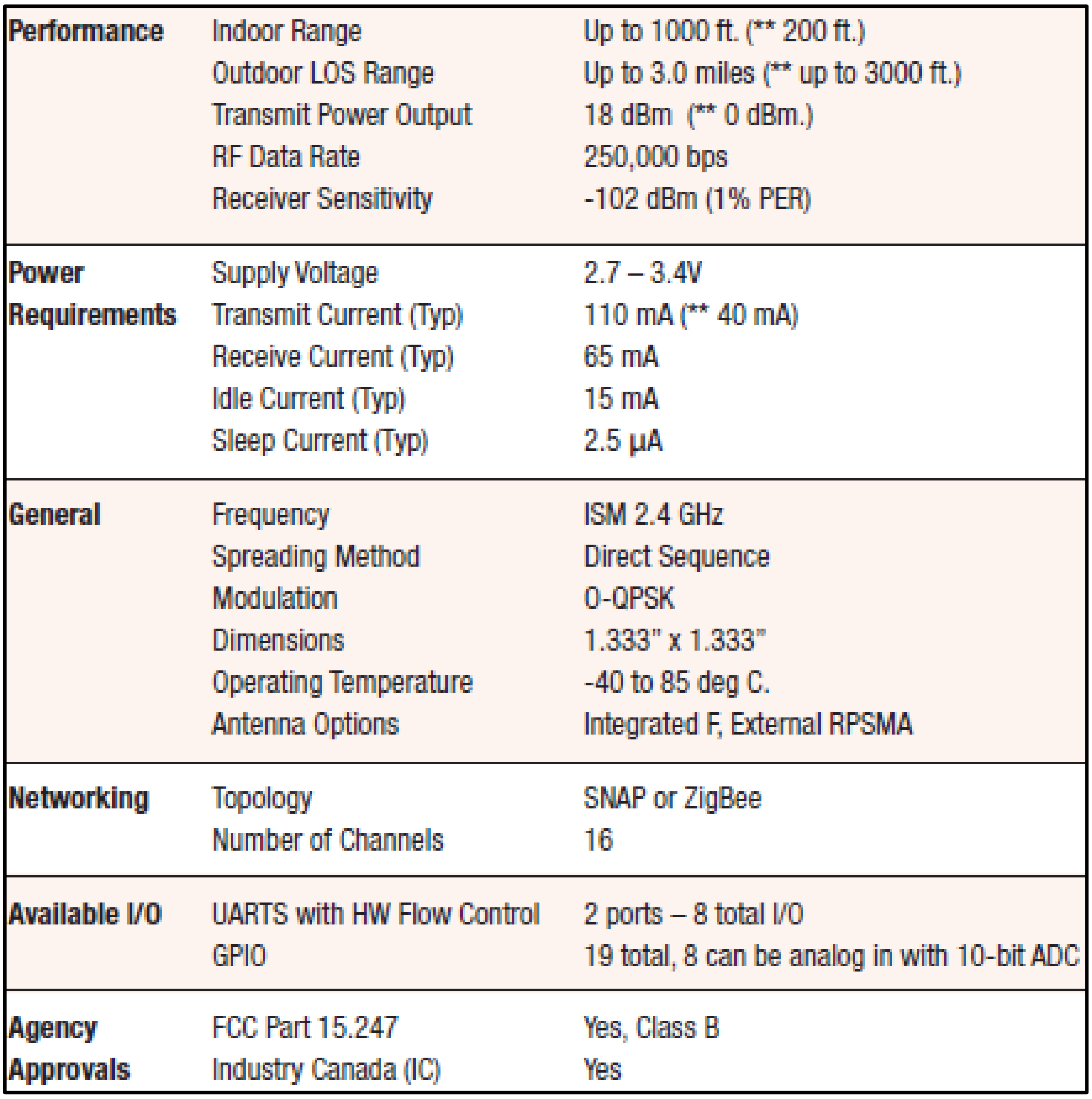
Each RF Engine combines a microcontroller, an 802.15.4 radio, and an antenna. It has an on-board microcontroller with its own internal RAM and ROM. No external components are required for operation. It is includes 19 General Purpose I/O (GPIO) pins, which can be configured as digital inputs or outputs. Many of these same 19 GPIO pins can also be switched to alternate functionality:

- 8 can be analog inputs
- 4 can be serial data lines (2 TX pins, 2 RX pins)
- 4 can be serial handshake lines (2 RTS pins, 2 CTS pins)

There are 19 Input/Output pins available to hook up the exact functionality required by the application. The minimal hookup to an RF Engine consists of two wires, one wire for VCC (2.7-3.4 volts DC) and the other wire for GND.

The RF Engines contain core code (written in C) that implements basic wireless networking functionality. This core code also implements a virtual machine which executes a subset of the Python programming language. Synapse has named this subset Python SNAPpy.

Table 3.1: RF Engine Specifications [16]



Performance	Indoor Range	Up to 1000 ft. (** 200 ft.)
	Outdoor LOS Range	Up to 3.0 miles (** up to 3000 ft.)
	Transmit Power Output	18 dBm (** 0 dBm.)
	RF Data Rate	250,000 bps
	Receiver Sensitivity	-102 dBm (1% PER)
Power Requirements	Supply Voltage	2.7 – 3.4V
	Transmit Current (Typ)	110 mA (** 40 mA)
	Receive Current (Typ)	65 mA
	Idle Current (Typ)	15 mA
	Sleep Current (Typ)	2.5 µA
General	Frequency	ISM 2.4 GHz
	Spreading Method	Direct Sequence
	Modulation	O-QPSK
	Dimensions	1.333" x 1.333"
	Operating Temperature	-40 to 85 deg C.
	Antenna Options	Integrated F, External RPSMA
Networking	Topology	SNAP or ZigBee
	Number of Channels	16
Available I/O	UARTS with HW Flow Control	2 ports – 8 total I/O
	GPIO	19 total, 8 can be analog in with 10-bit ADC
Agency Approvals	FCC Part 15.247	Yes, Class B
	Industry Canada (IC)	Yes

Table 3.1 illustrates the specifications of the modules. These provide up to 16 channels of operation in the ISM 2.4 GHz frequency band. In addition it contains both a power amplifier for transmission and a low noise amplifier in the receive path for extended range. Where the networking is based on SNAP or ZigBee topology this is the same as mesh networking topology.

3.4 Evaluation kit

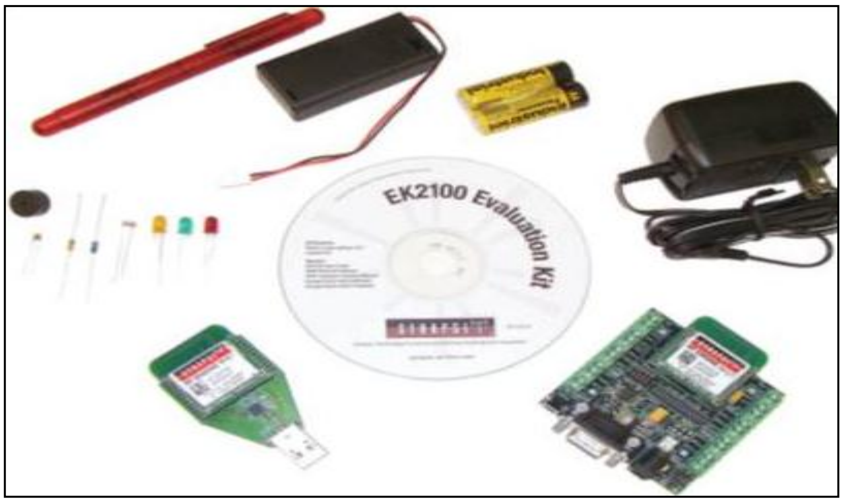


Figure 3.5: Evaluation kit [17]

The evaluation kit has four main components [17]: SN132 SNAPstick, Synapse Portal software and SN171 SNAP Node Proto Board as shown in Figure 3.5

3.4.1 SN132 SNAPstick

The Synapse SNAPstick as shown in Figure 2. 1 is a USB connector that connects the RF engines to the PC. Once the RF engines are connected to the PC, the RF engine can be easily programmed via Synapse Portal software [17].

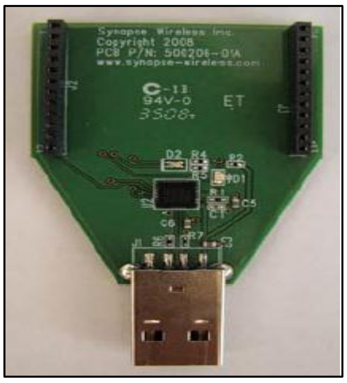


Figure 3.6: SN132 SNAPstick [17]

3.4.2 Synapse Portal software

Synapse Portal software is a graphical user interface that enables users to access the RF engines. Users can upload Python codes and monitor the activities of RF engines via Synapse Portal.

3.4.3 SN171 SNAP Node ProtoBoard

The ProtoBoard exposes the RF Engine’s GPIO pins. In this system, ProtoBoard is very important because it allows the connection of the switches to the RF engines.

3.5 Station design

The station components are: RF Module, ProtoBoard switches, 3V power adapter and 2x1.5 AA batteries. Figure 3.7 shows the circuit structure of the station.

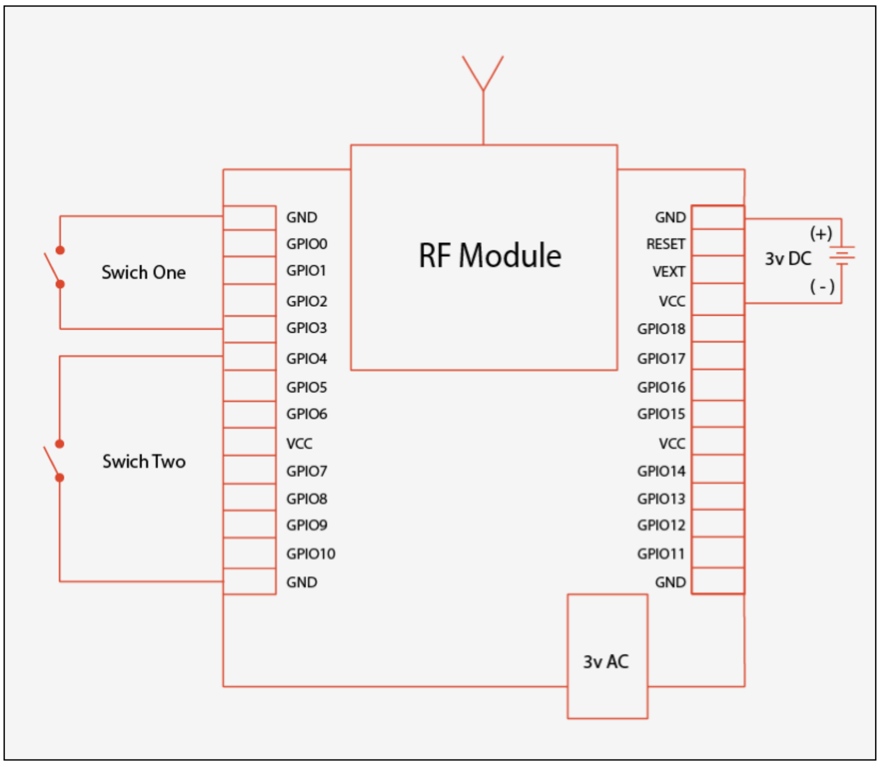


Figure 3.7: The circuit structure of the station

A 3V power adapter will power the station. However, the station is provided with disposable 2x1.5 AA batteries in case of power failure. The RF module is programmed to send pings to the bridge node when the switches are pressed. The first switch will be for the medical support and it will be connected to GPIO3 on the board. The second switch is for the security support and it will be connected to GPIO4 on the board. The GPIOs are identified in the RF engine's python code.

3.6 Station nodes' Python scrip

In the station node, python programing language has been used to set up the codes of the two switches for the medical and security switches. Switch one is defined as input pin 3 and switch two is defined as input pin 4 by using the function called def onStartup(): and def init(pin): as shown in Figure 3.8 below.

```
def onStartup():

   # Set pin 5 as a watched input

   init(3);

   init(4);

def init(pin):

   setPinDir(pin, False)

   setPinPullup(pin, True)

   monitorPin(pin, True)
```

Figure 3.8: start-up function code

Pin 1 was set as an output in the start-up function by using the function called # Set pin 1 as an output as shown in Figure 3.9.

```
 # Set pin 1 as an output

 setPinDir(1, True)

 writePin(1, False)
```

Figure 3.9: (Set pin) Function

When switches one or two are pressed the station node calls (onPin) function to sends RPC signals to the bridge node as shown in Figure 3.10.

```
def onPin(pin, isSet):

    if isSet:

        if pin == 3:

              rpc('\x00\x00\x01', "security", 1)

        elif pin == 4:

            rpc('\x00\x00\x01', "medical", 1)
```

Figure 3.10: (onPin) function call

In addition, the station node calls the function *check()* to confirm whether the sensor node is working or not. When the function is triggered, it will send RPC to the server to run *checkstations* function on the client (the node), with the network address of the node *localAddr()* as a parameter.

```
def check():

 rpc('\x00\x00\x01', 'checkstations', localAddr())
```

Figure 3.11: *check(),(checkstations)* and *(localAddr())* function call

3.7 SNAPconnect

SNAPconnect works as a middleware between the bridge and the GUI software [18]. It is used to allow a third party, who is the client, to easily access a SNAP wireless network through GUI applications. It provides a terminal for the client programs to interact with remote station nodes. SNAPconnect exposes the functions contained in SNAPpy Scripts including the standard SNAP built-in functions. The following diagrams outline the message exchange sequence between the SNAPconnect server and the client application, as well as between the server and a remote SNAP node.

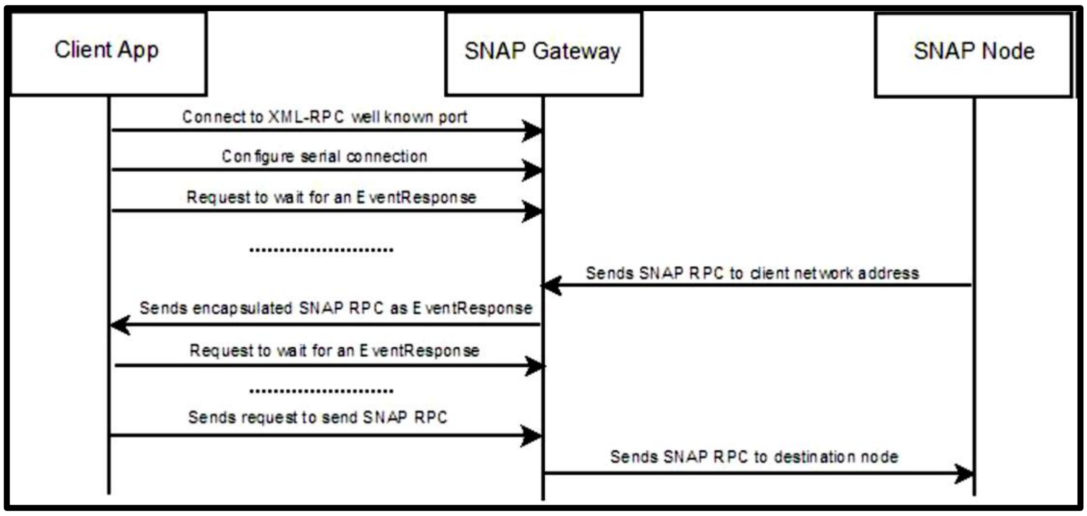

Client App
SNAP Gateway
SNAP Node
Connect to XML-RPC well known port
Configure serial connection
Request to wait for an EventResponse
....
Sends SNAP RPC to client network address
Sends encapsulated SNAP RPC as EventResponse
Request to wait for an EventResponse
....
Sends request to send SNAP RPC
Sends SNAP RPC to destination node

Figure 3.12: Message Exchange (SNAP node initiated) [18]

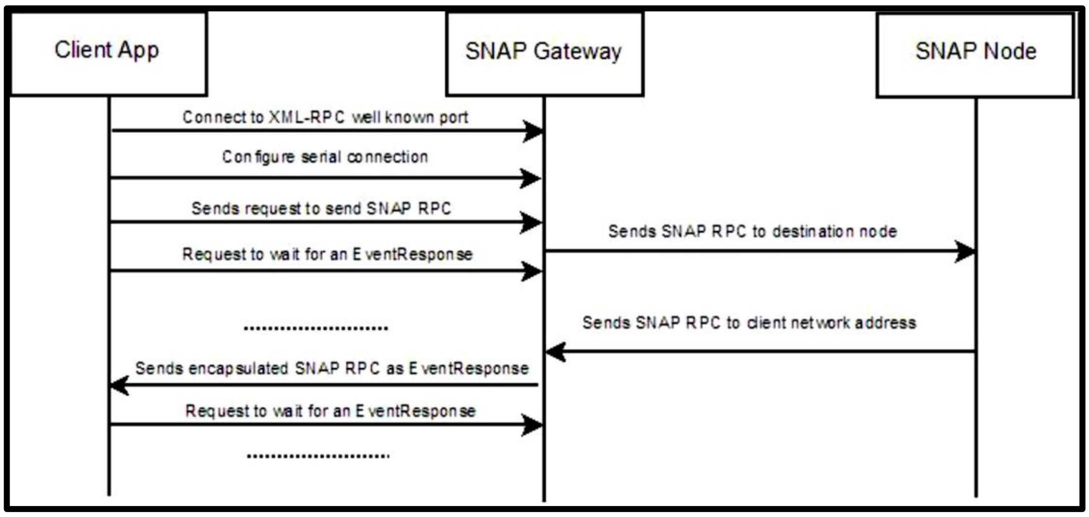

Client App
SNAP Gateway
SNAP Node
Connect to XML-RPC well known port
Configure serial connection
Sends request to send SNAP RPC
Sends SNAP RPC to destination node
Request to wait for an EventResponse
....
Sends SNAP RPC to client network address
Sends encapsulated SNAP RPC as EventResponse
Request to wait for an EventResponse
....

Figure 3.13: Message Exchange (client application initiated) [18]

Figure 3.12 shows a client connecting to SNAPconnect software, and then an exchange initiated by a remote SNAP node while the client application waits. Figure 3.13 shows a similar setup and exchange initiated by the client application [18].

3.8 Graphical user interface (GUI)

The GUI was programmed with C# language in correlation with the XML\RPC function library. The GUI is divided into four sections as shown in Figure 3.14

- Events.
- Message box.
- Statistics frame.
- Map.

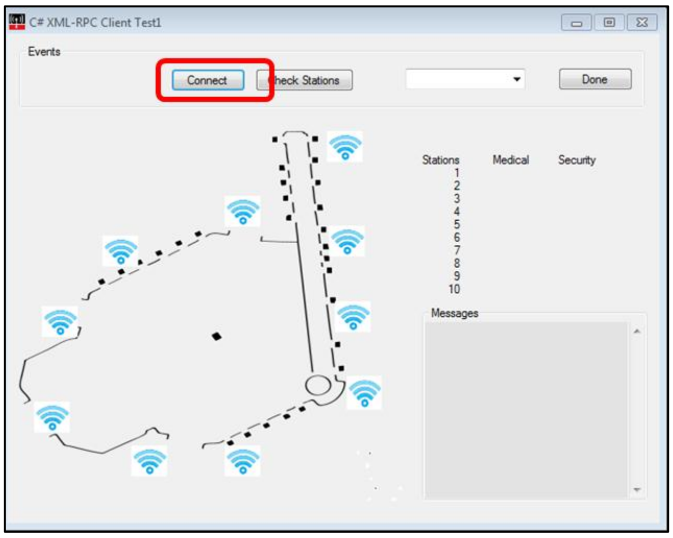


Figure 3.14: Graphical user interface (GUI) of the system

3.8.1 Events

In the event box there are three buttons which are the Connect button, Check station button, and Pull up menu button.

3.8.1.1 Connect button

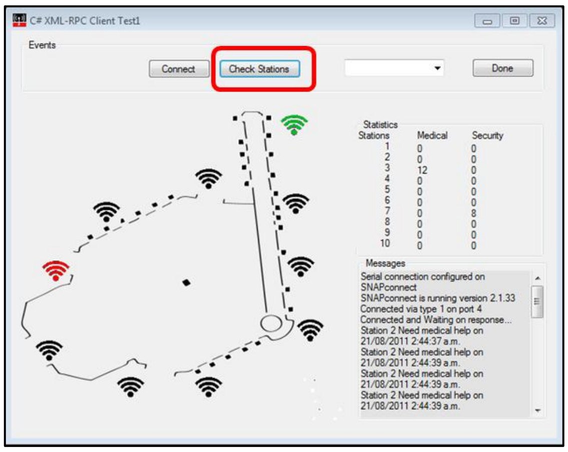
The connect button is a button to connect the PC with the bridge node by SNAPconnect server. The connect button calls the *BeginConnectSerial* function, which is defined in *XmlRpcInterface.cs* file included in the XML\RPC function library. Figure3.15 shows the connect button code.

```
proxy.BeginConnectSerial((int)SerialType.RS232,(int)4,false,this.SerialConnectCallbak
```

Figure 3.15: Connect button code

3.8.1.2 Check station button

The check station button was created to check all station nodes statues. For example, figure 3.16 shows that only two stations are working and the rest of the stations are disabled.



Statistics

Stations	Medical	Security
1	0	0
2	0	0
3	12	0
4	0	0
5	0	0
6	0	0
7	0	8
8	0	0
9	0	0
10	0	0

Messages

```
Serial connection configured on
SNAPconnect
SNAPconnect is running version 2.1.33
Connected via type 1 on port 4
Connected and Waiting on response...
Station 2 Need medical help on
21/08/2011 2:44:37 a.m.
Station 2 Need medical help on
21/08/2011 2:44:39 a.m.
Station 2 Need medical help on
21/08/2011 2:44:39 a.m.
Station 2 Need medical help on
21/08/2011 2:44:39 a.m.
```

Figure 3.16: check station button example

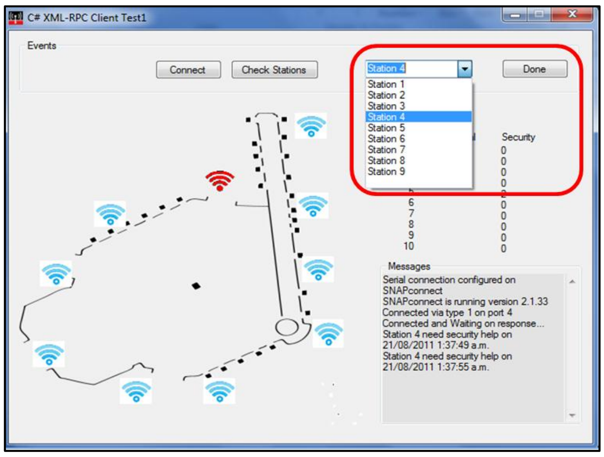
Figure 2.16. Below shows the function: *checkstations(byte[] netAddr)*, which is used to call the stations for checking the statues of each node.

```
public void checkstations(byte[] netAddr)

   {

       string statinNetAdress = HexStr(netAddr);

       string stationNum = whichstation(statinNetAdress);

       int StationInt = Convert.ToInt32(stationNum);

 checkStations[StationInt- 1] = 1;

       mapsignal(StationInt, "idle.gif");
```

Figure 3.17: Check station button code

3.8.1.3 Pull up menu

The pull up menu presents a list of the station nodes. When the station is activated, and a appropriate action is taken by the authorities, the pull-up menu can be used to reset the station to inactive status, as shown in Figure 3.18

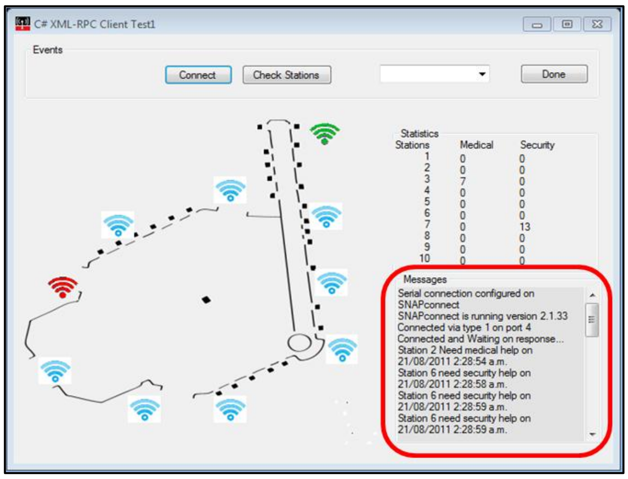


```
6     0
7     0
8     0
9     0
10    0
```

```
Serial connection configured on
SNAPconnect
SNAPconnect is running version 2.1.33
Connected via type 1 on port 4
Connected and Waiting on response...
Station 4 need security help on
21/08/2011 1:37:49 a.m.
Station 4 need security help on
21/08/2011 1:37:55 a.m.
```

Figure 3.18: Pull up menu

3.8.2 Messages Box



Stations	Medical	Security
1	0	0
2	0	0
3	7	0
4	0	0
5	0	0
6	0	0
7	0	13
8	0	0
9	0	0
10	0	0

```
Serial connection configured on
SNAPconnect
SNAPconnect is running version 2.1.33
Connected via type 1 on port 4
Connected and Waiting on response...
Station 2 Need medical help on
21/08/2011 2:28:54 a.m.
Station 6 need security help on
21/08/2011 2:28:58 a.m.
Station 6 need security help on
21/08/2011 2:28:59 a.m.
Station 6 need security help on
21/08/2011 2:28:59 a.m.
```

Figure 3.19: message box Frame

Figure 3.19 show the message box shows the station node that has been pressed, the type of request, date and time by using functions below in Figure 3.20

```
txtInfo.Text += "\r\n"+"Station " + stationNum +

               " Need medical help" + " on " + localZone.ToLocalTime(time).ToString();
```

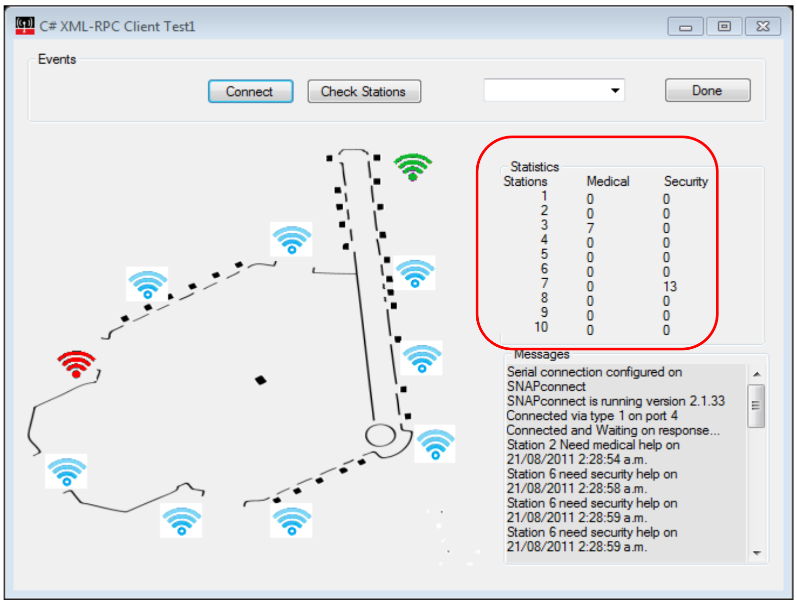
Or

```
txtInfo.Text += "\r\n"+ "Station " + stationNum +

               " need security help" + " on " + localZone.ToLocalTime(time).ToString();
```

Figure 3.20: message box code

3.8.3 Statistics Frame

The Statistics frame shows the station number and the accumulative request number of each station and whether a medical or security request, as shown in Figure 3.21.



Stations	Medical	Security
1	0	0
2	0	0
3	7	0
4	0	0
5	0	0
6	0	0
7	0	13
8	0	0
9	0	0
10	0	0

```
Serial connection configured on
SNAPconnect
SNAPconnect is running version 2.1.33
Connected via type 1 on port 4
Connected and Waiting on response...
Station 2 Need medical help on
21/08/2011 2:28:54 a.m.
Station 6 need security help on
21/08/2011 2:28:58 a.m.
Station 6 need security help on
21/08/2011 2:28:59 a.m.
Station 6 need security help on
21/08/2011 2:28:59 a.m.
```

Figure 3.21: Statistics Frame

The message box presents all the data from the station node by using the function below as shown in Figure 3.22.

```
for (int m = 0; m < 10; m++)
        {
           textBox1.AppendText("\r\n" + MedicalStations[m].ToString());
        }
```

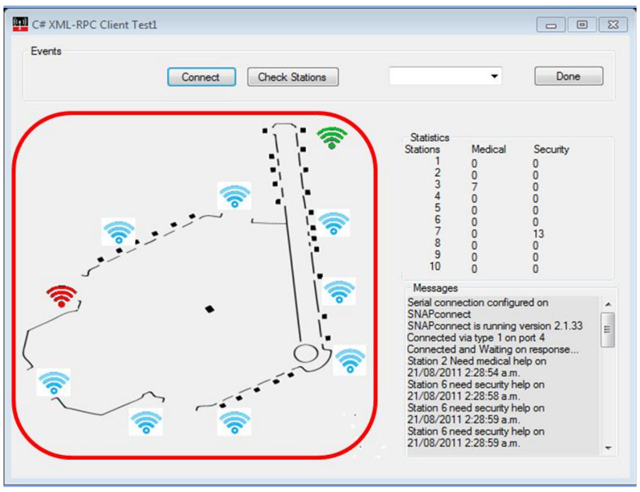
Or

```
for (int m = 0; m < 10; m++)
        {
           textBox2.AppendText("\r\n" + SecurityStations[m].ToString());
        }
```

Figure 3.22: statistics frame code

3.8.4 Map

The map shown in figure 3.23 was drawn to represent the layout of the mosque, and it shows the main entry doors of the mosque and where the station nodes will be positioned in the real implementation as shown in Figure 3.23.



Statistics

Stations	Medical	Security
1	0	0
2	0	0
3	7	0
4	0	0
5	0	0
6	0	0
7	0	13
8	0	0
9	0	0
10	0	0

Messages

```
Serial connection configured on
SNAPconnect
SNAPconnect is running version 2.1.33
Connected via type 1 on port 4
Connected and Waiting on response...
Station 2 Need medical help on
21/08/2011 2:28:54 a.m.
Station 6 need security help on
21/08/2011 2:28:58 a.m.
Station 6 need security help on
21/08/2011 2:28:59 a.m.
Station 6 need security help on
21/08/2011 2:28:59 a.m.
```

Figure 3.23: The map layout

When the function is executed it will translate the byte from python binary code to Hexadecimal through (HexStr(byte[] p)) function. The station number will be retrieved from (whichstation()) function. Then it will use (mapsignal()) function to show the right status icon on the map (alert , idle, down). The criterion of the icon is explained in table 3.1 below.

Table 3.1: icons criteria

Icon Image	Description
🟢	When the MIDICAL button is pressed.
🔵	When the station is idle or not pressed yet.
🔴	When the SCURITY button is pressed.
⚫	When the node is disabled.

When a buttons on the node is pressed it will send RPC calling (MEDICAL() or SECURITY ()) functions to the client. As shown in Figures2.22, 2.23.

```
{
  textBox1.AppendText("\r\n" + MedicalStations[m].ToString());
}
```

Figure: 3.24 Medical Stations code

```
{
  textBox2.AppendText("\r\n" + SecurityStations[m].ToString());
}
```

Figure: 3.25 Security Stations code

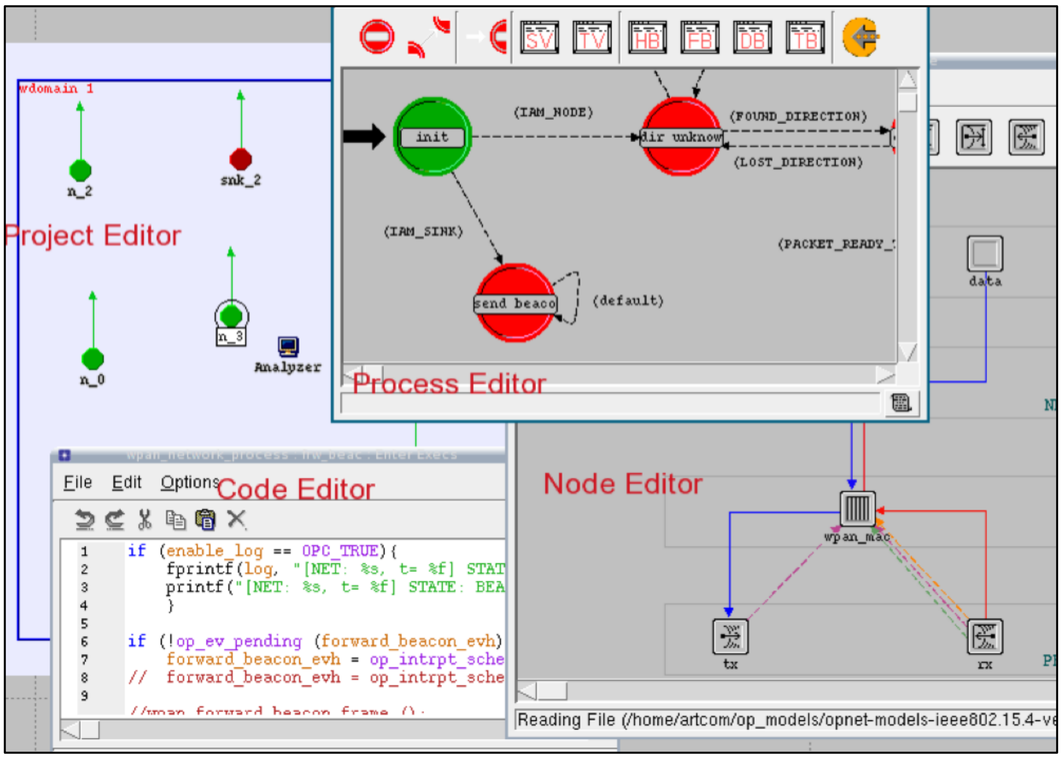
CHAPTER 4

SIMULATION AND RESULTS

CHAPTER 4 SIMULATION AND RESULTS

4.1 Introduction

The simulation section in this thesis is very important for the development and evaluation of the network. The simulation shows the expected behaviour of the network under different conditions [20]. The OPNET Modeller simulator has been used due to its accuracy and ease of use. Figure 4.1 shows the OPNET simulation environment.



```
if (enable_log == OPC_TRUE){
    fprintf(log, "[NET: %s, t= %f] STAT
    printf("[NET: %s, t= %f] STATE: BEA
    }

if (!op_ev_pending (forward_beacon_evh)
    forward_beacon_evh = op_intrpt_sche
//  forward_beacon_evh = op_intrpt_sche

//wpan_forward_beacon_frame ();
```

Reading File (/home/artcom/op_models/opnet-models-ieee802.15.4-v

Figure 4.1: OPNET simulation environment

The OPNET Modeller also supports IEEE 802.15.4 Standard and all three types of devices used (Coordinator, Router and End device) [21]. The results obtained from the simulation are throughput, packet dropped, data traffic received and data traffic sent for the project scenarios.

4.2 IEEE 802.15.4 Standard OPNET simulator

The OPNET Modeller simulator for IEEE 802.15.4 gives full access to change the structure of the node to suit the system requirements [21]. Figure 4.2 shows the model structure for a one node model including physical, MAC, network and application layer beside the battery model. The PHY has a transmitter and a receiver working at 2.4 GHz frequency. The MAC layer contains slotted CSMA/CA, generates beacon frame and synchronizes nodes with a Coordinator. The APP layer generates data using unacknowledged frames and a MAC command frame generator creating acknowledged frames. The battery module is used for measuring the consumed and remaining energy levels [22].

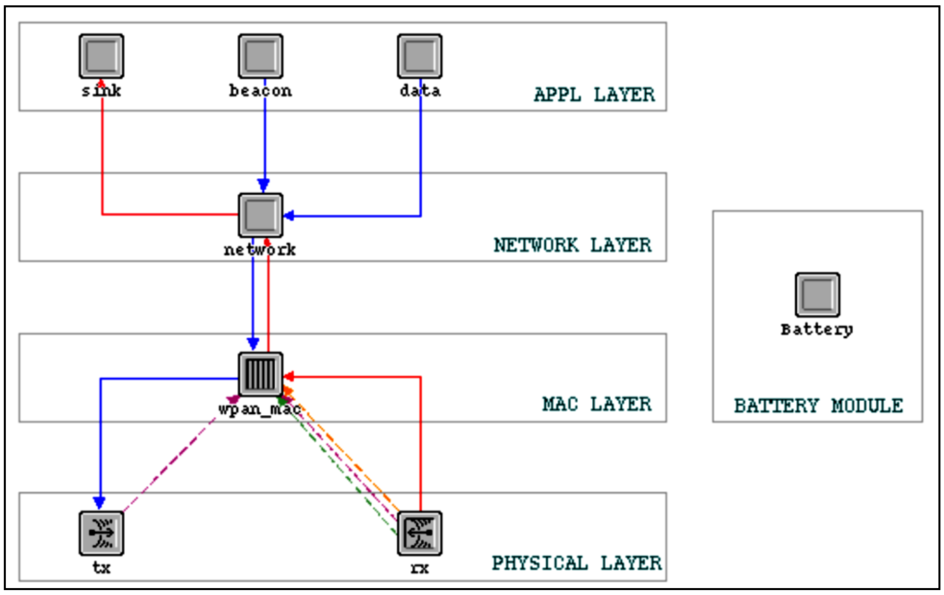


Figure 4.2: The structure of the IEEE 802.15.4 simulation model

4.3 Types of Wireless Sensor Network Topology

This section explains the topology model used in the OPNET Modeller simulator. In particular the mesh networking topology which has been used in this project. The IEEE 802.15.4 model in OPNET may operate in one of three topologies which are Star topology, Cluster tree topology and Mesh networking topology as shown in Figure 4.3.

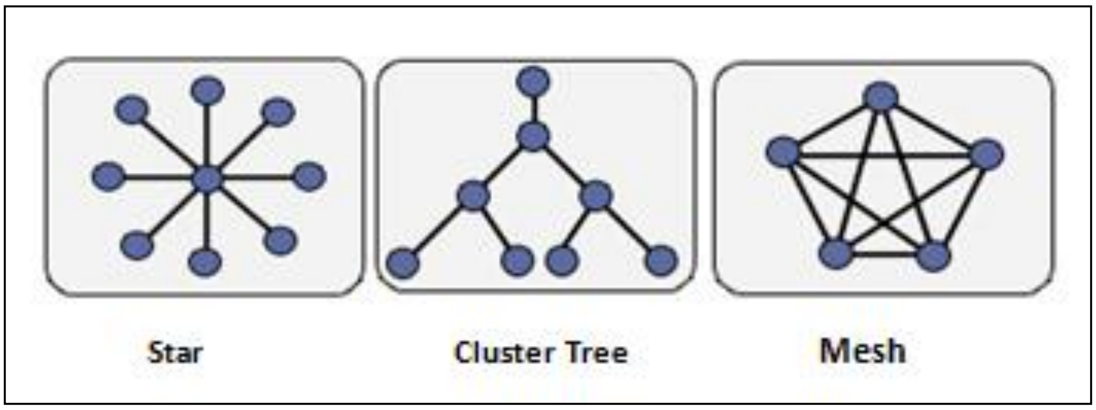


Figure 4.3: Types of Wireless Sensor Network Topology

Wireless sensor networks are typically organized in one of three types of network topologies: star, cluster tree, or mesh as shown in Figure 3.3. Network topologies are similar regardless of the areas of their application. WSN is still a network, so the common network topologies work for it as well.

4.3.1.1 Star Network Topology

Star network topology is based on a single base-station that sends and/or receives a message to or from a number of remote nodes. Therefore, the remote nodes are able to send/receive messages to a base-station only, not to each other. The simplicity of this type of network is the main advantage. Another advantage is the ability to keep the power consumption of nodes at a minimum [8] as shown in Figure 4.3. Almost all Wi-Fi networks use a star network, where each client (sensor node) is tied to a wireless access point (the gateway) [23].

4.3.1.2 Cluster Tree Network Topology

In a cluster tree network, each node connects to a node in the top of the tree until the data reaches the gateway. These can be used by employing a simple algorithm with any wireless technology [24]

4.3.1.3 Mesh Network Topology

The mesh network allows any node in the network to transmit to any other node in the network that is within its radio transmission range. If one link is affected, and cannot be passed then the data will reach the nearest available node, then the data can go through another route to reach the gateway [25].

4.4 Node types

As mentioned before the node types in the IEEE 802.15.4 are Coordinator, Router and End device. In this project we make use of the coordinators and routers, where the coordinator acts as the main device. The main purpose of the coordinator is initiating and synchronizing the network. It also acts as a router. The routers are the reader nodes in this project and they can relay messages from one node to another.

4.5 Simulation setup

This simulation contains of ten routers and one coordinator and the simulation scenario is 700m X 200m matching the ground mosque area as seen in Figure 4.3. The simulation runtime is 5000 seconds. Some modifications were introduced to the nodes parameters to meet the system requirements. The modified parameters are shown in Table 1.1.

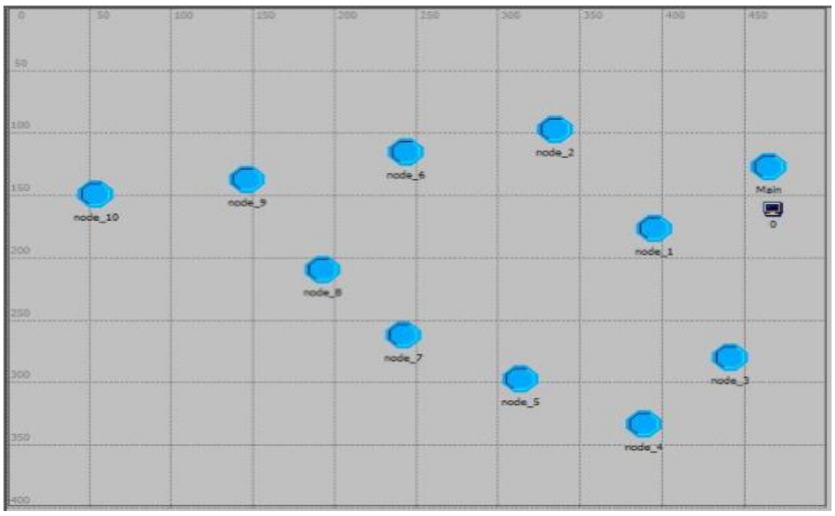


Figure4.4: Simulation scenario layout

Table 4.1: Nodes parameters

Physical layer	
Data rate	**250 kbps**
Receiver sensitivity	**-102 dBm**
Transmission band	**2.4 GHz**
Transmission Power	**18 dBm**
MAC parameters	
ZCK Wait Duration	**0.05**
Number of Retransmission	**5**
CSMA Parameters	
Minimum Back off Exponent	**3**
Maximum Back off Exponent	**4**
Channel Sense Duration	**0.1**
Topology type	**Mesh**
Synchronization mode	**Beacon-enabled**
Battery Module Parameters	
Radio power in receiving	**66 mA**
Radio power in sending	**110 mA**
Radio power in idle	**15 mA**
Radio power in sleeping	**2.5 µA**

4.6 Simulation results

The obtained results for this simulation are average throughout, average data dropped, average delay, network collision ratio and total number of received and sent packets.

4.6.1 Throughput:

Throughput is related to the network capacity and the number of nodes has a direct impact on the network capacity. Figure 4.5 shows the average throughput for this scenario with the 10 nodes.

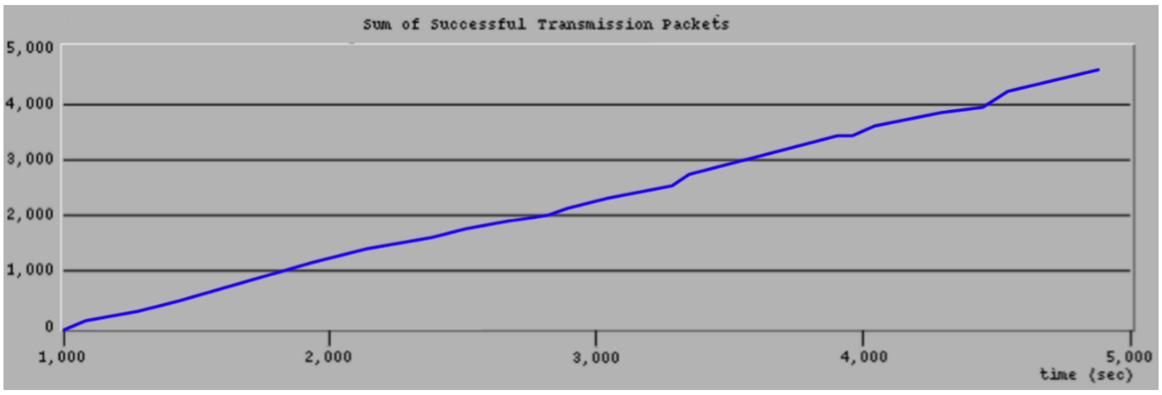


Figure 4.5: Average throughput of the network

The network capacity as seen in this figure is relatively high which will result in high network performance.

4.6.2 The Average Delay and Packet loss:

The delay of a network specifies how long it takes for the packets to travel across the network to reach their destination. Packet loss happens when the packets fail to reach their destination. The Average delay and packet loss for this scenario are shown in Figure 4.6 and 4.7.

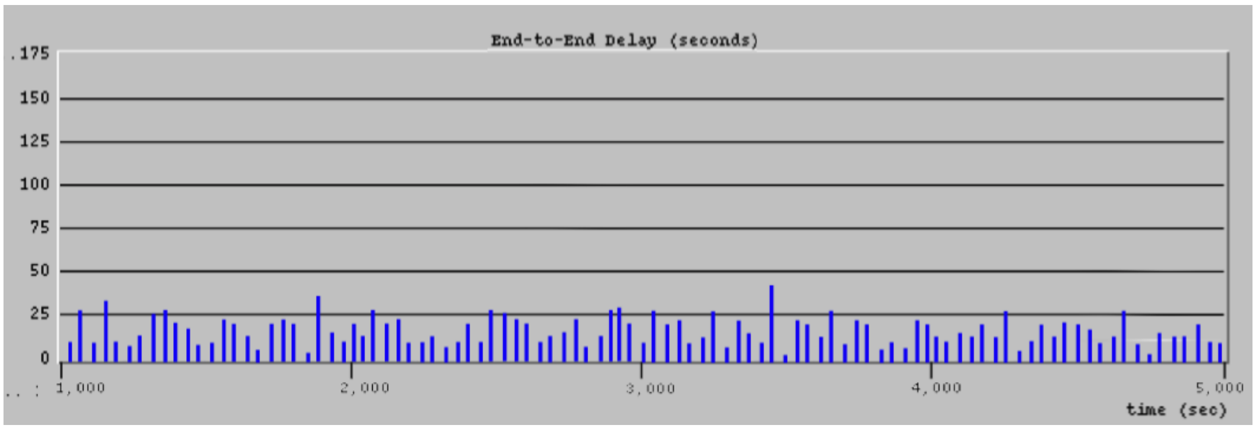


Figure 4.6: Network average delay

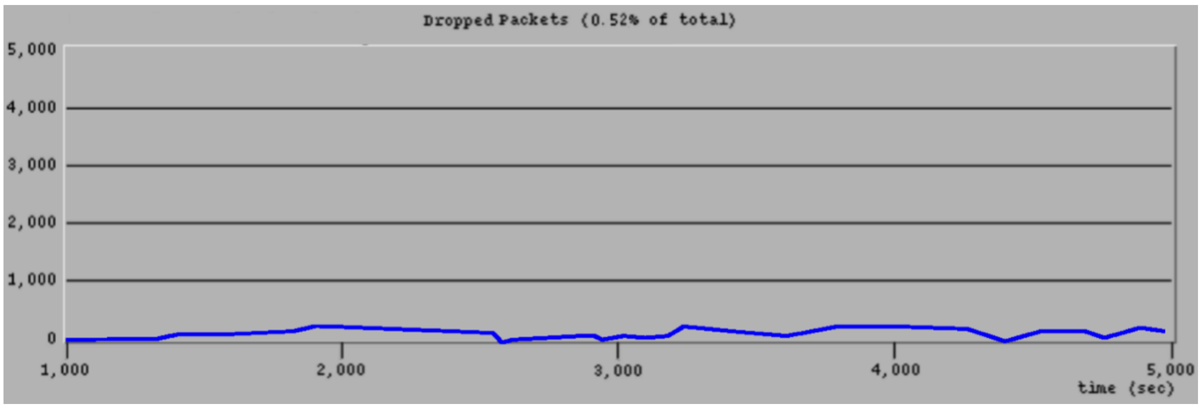


Figure 4.7: Data loss rate

The results show low delay and packet loss rates. High network congestion usually causes high delay and packet loss rates. Figure 4.8 shows the collision ratio in this simulation which is 0.012%.

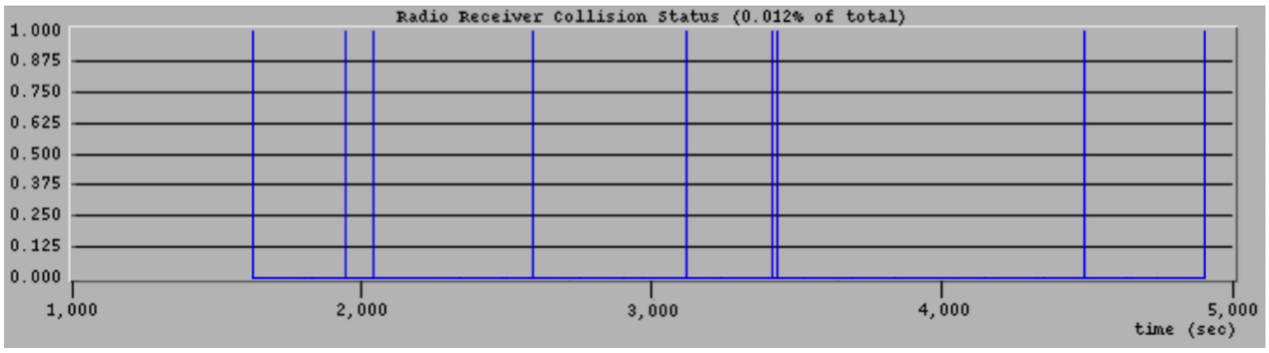


Figure 4.8: Network collision ratio

4.6.3 Total number of received and sent packets

The difference between the number of sent packets by the ten readers and received packets by the coordinator can be clearly seen in Figure 5.9.

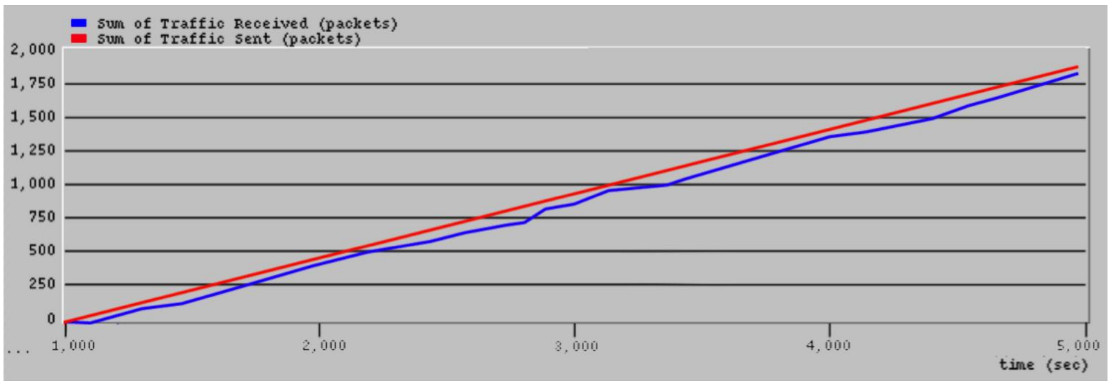


Figure 4.9: Total number of received and sent packets

4.7 Conclusion

This chapter has presented a simulation of the proposed system. The OPNET modeller has been used to create a scenario of 10 nodes and one coordinator. The obtained results have shown stability in sending and resaving packets with low rates of delay and packet loss. According to these results and due to the small size of the network, the system will be expected to achieve high performance.

CHAPTER 5

EXPERIMENT SET- UP AND RESULTS

CHAPTER 5 EXPERIMENT SET- UP AND RESULTS

5.1 Introduction

This chapter presents the actual field experiment conducted including the selecting of a suitable area and the tools used to do the experiment. Several tests and results have been gathered from each test. And each experiment has tested test of different aspects. The major consideration in this test is the signal strength and the quality of the signal when sending or receiving.

5.2 Field Experiment

The field of experiment was conducted at the **Victoria Esplanade Garden located** on the Park Road in Palmerston North, New Zealand. The site was selected because it has a similar area size to the Grand Mosque in Mecca, which has a length of approximately 440 meters by 270 meters, as shown in Figures 5.1 and 5.2

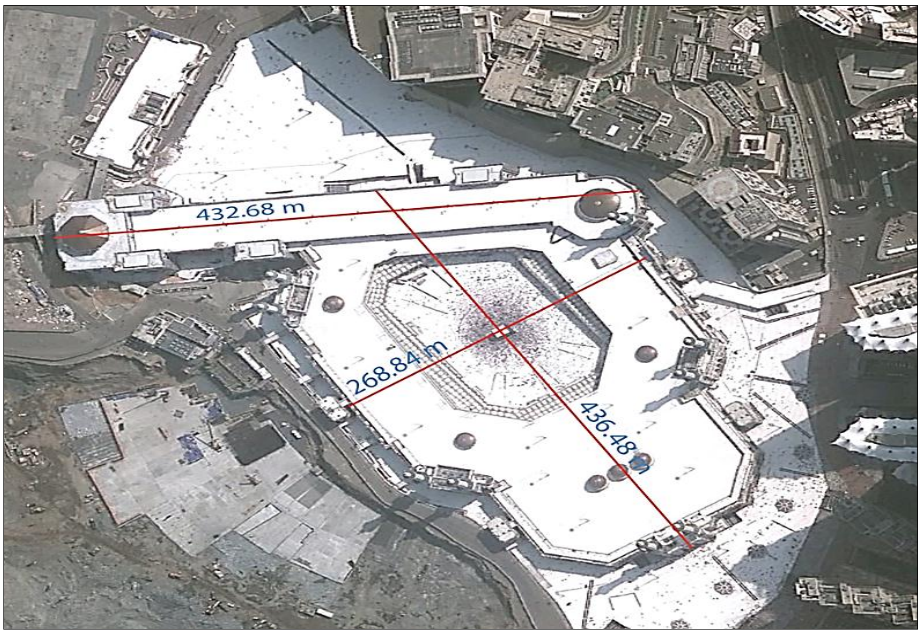


Figure 5.1: Dimensions of the Grand Mosque in Mecca

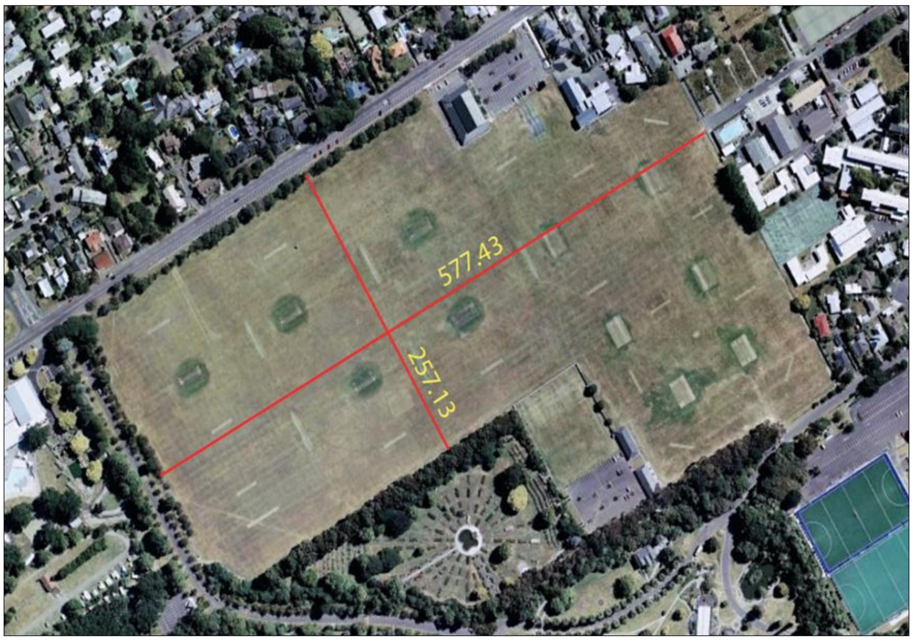


Figure 5.2: Dimensions of the Victoria Esplanade Garden

5.3 Positioning of the sensors (draw the map network)

One of the primary stages of the experiment is nodes positioning including the main station. The distance between each sensor node must be considered .Ten sensor nodes have been placed on the map of the Mosque of Mecca based on the main gates of the Mosque as shown in Figure 5.4. The location of the station nodes has been carefully determined to insure a fully connected circle around the mosque. The map of the Mosque was matched with the map of the Victoria Esplanade Garden, showing the sensor node yellow, and the main station in red a receiver as shown in Figure 5.3, 5.4. Each sensor node has a number from 1 to 10 to make them easy to recognise and monitor.

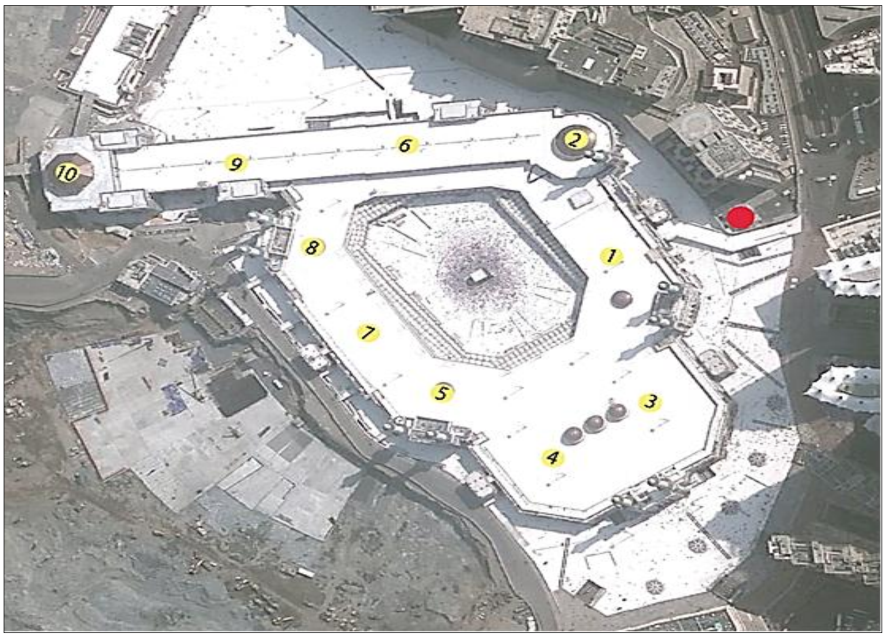


Figure 5.3: Positioning of the sensor node in Mecca

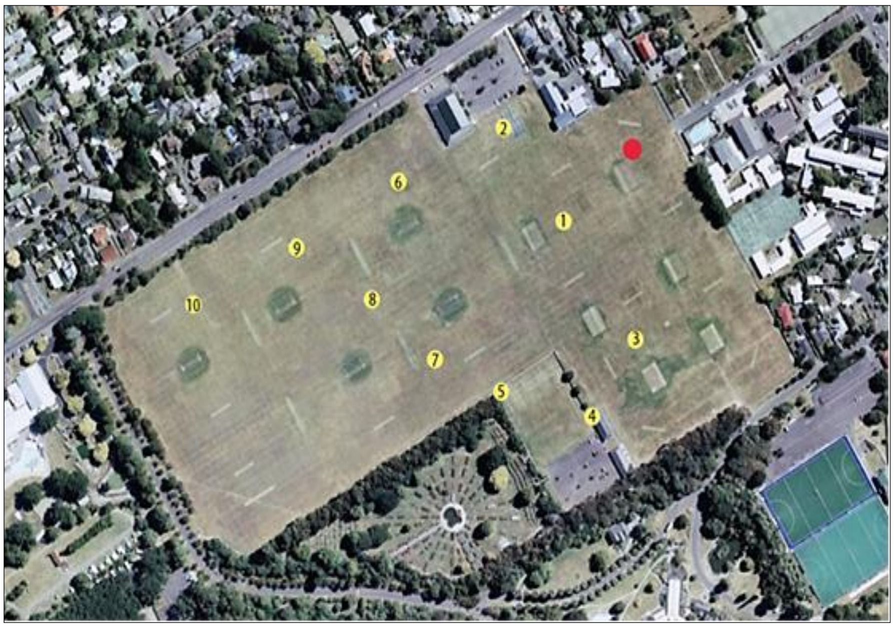


Figure 5.4: Positioning of the sensor node in Victoria Esplanade Garden

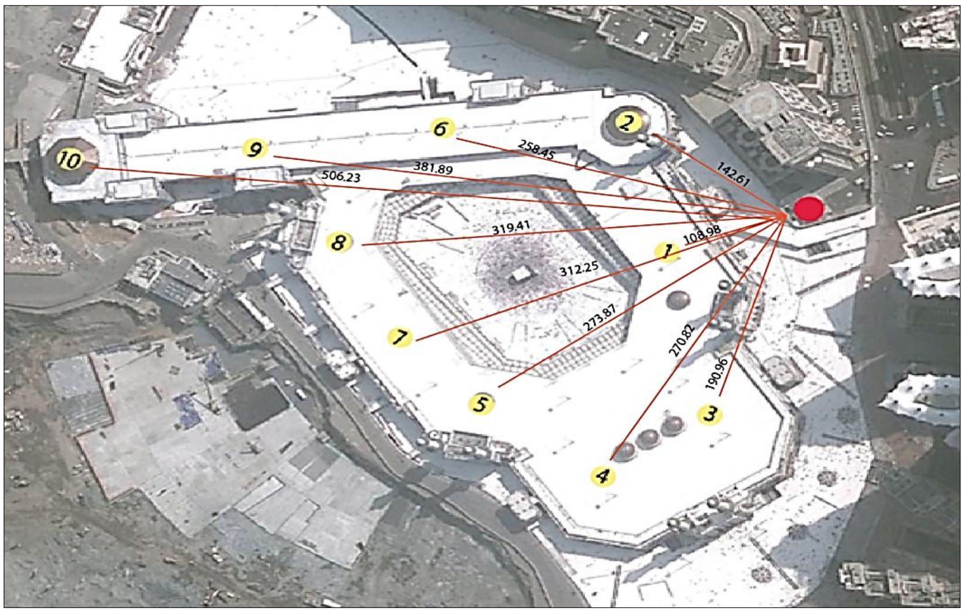


Figure 5.5: Distance from the main node in Mecca

Table 5.1: Distance from the main node

Node	Distance from the main station
Sensor node 1	108.89 M
Sensor node 2	142.61 M
Sensor node 3	190.96 M
Sensor node 4	270.82 M
Sensor node 5	273.78 M
Sensor node 6	258.45 M
Sensor node 7	312.34 M
Sensor node 8	319.00 M
Sensor node 9	381.89 M
Sensor node 10	506.23 M

Distances are assisted with a high consideration of the type of network topology Figure 5.5 and the table 5.1 show clearly the distance from each sensor node to the main sensor, measured in metres.

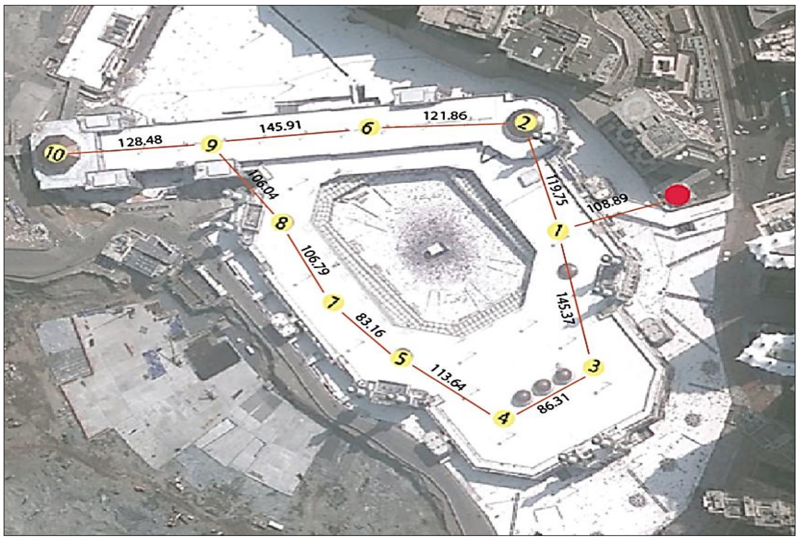


Figure 5.6: Distance between nodes

Table 5.2: Distance between nodes

Sensor node number	Distance
From node 1 to 2	119.75 M
From node 2 to 6	121.86 M
From node 6 to 9	145.91 M
From node 9 to 10	128.48 M
From node 9 to 8	106.06 M
From node 8 to 7	106.79 M
From node 7 to 5	83.16 M
From node 5 to 4	113.64 M
From node 4 to 3	86.31 M
From node 3 to 1	145.37 M

The distances between the sensor nodes were selected to ensure a good quality of the signal strength, as shown in Figure 5.6 and Table 5.2.

5.4 Tools of the Experiment

The sensor and the system which had been used in the project is RF ENGINE from SYNAPSE Wireless, Inc as mentioned previously in chapter three.

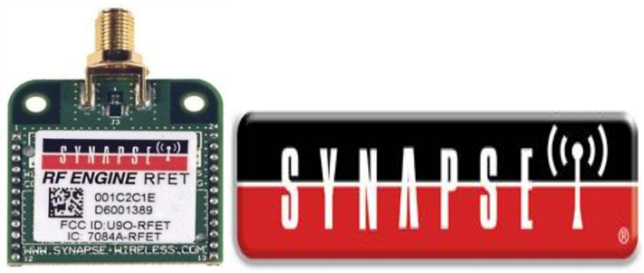
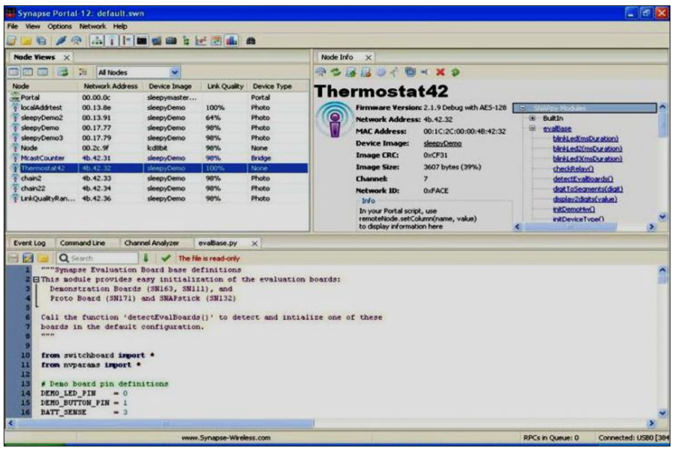


Figure 5.7: SYNAPSE Wireless

Synapse Portal software, from SYNAPSE Wireless, has been used in this project to become a graphical user interface (GUI) for the entire network [19]. Portal is used to monitor the sensor nodes RF ENGINE which is a SNAP-based network application. Once connected to the USB or RS232 interface, it connects to any node in the SNAP Wireless Network, as shown in Figure 5.8



Node	Network Address	Device Image	Link Quality	Device Type
Portal	00.00.0c	sleepymaster...		Portal
localAddrtest	00.13.8e	sleepyDemo	100%	Photo
sleepyDemo2	00.13.91	sleepyDemo	64%	Photo
sleepyDemo	00.17.77	sleepyDemo	98%	Photo
sleepyDemo3	00.17.79	sleepyDemo	98%	Photo
Node	00.2c.9f	lcd8bit	98%	None
McastCounter	4b.42.31	sleepyDemo	98%	Bridge
Thermostat42	4b.42.32	sleepyDemo	100%	None
chain2	4b.42.33	sleepyDemo	98%	Photo
chain22	4b.42.34	sleepyDemo	98%	Photo
LinkQualityRan...	4b.42.36	sleepyDemo	98%	Photo

Thermostat42

Firmware Version:	2.1.9 Debug with AES-128
Network Address:	4b.42.32
MAC Address:	00:1C:2C:00:00:4B:42:32
Device Image:	sleepyDemo
Image CRC:	0xCF31
Image Size:	3607 bytes (39%)
Channel:	7
Network ID:	0xFACE

Info

In your Portal script, use remoteNode.setColumn(name, value) to display information here

```
"""Synapse Evaluation Board base definitions
This module provides easy initialization of the evaluation boards:
  Demonstration Boards (SN163, SN111), and
  Proto Board (SN171) and SNAPstick (SN132)

Call the function 'detectEvalBoards()' to detect and intialize one of these
boards in the default configuration.
"""

from switchboard import *
from nvparams import *

# Demo board pin definitions
DEMO_LED_PIN    = 0
DEMO_BUTTON_PIN = 1
BATT_SENSE      = 3
```

Figure 5.8: A snapshot of Portal

Each sensor node is programmed according to its network address and it's position. Table 5.3 below shows the number of the sensor node with the network address of the sensor, which appears on the back of each node, as shown in Figure 5.9

Table5.3: The network address of each node

Node Number	Network address
Node 1	03.EA.03
Node 2	03.EA.13
Node 3	03.EA.4C
Node 4	03.EE.C4
Node 5	03.EE.CC
Node 6	03.EB.B6
Node 7	03.EA.17
Node 8	03.EA.0F
Node 9	03.EA.15
Node 10	E6.00.A6.07

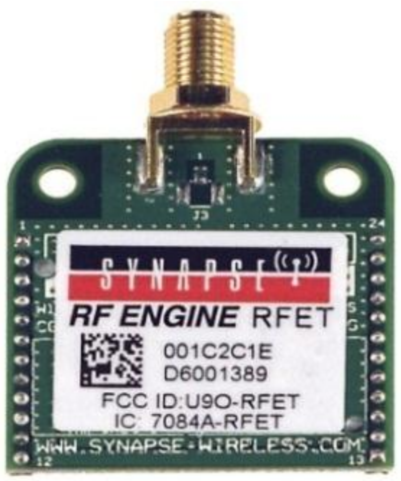
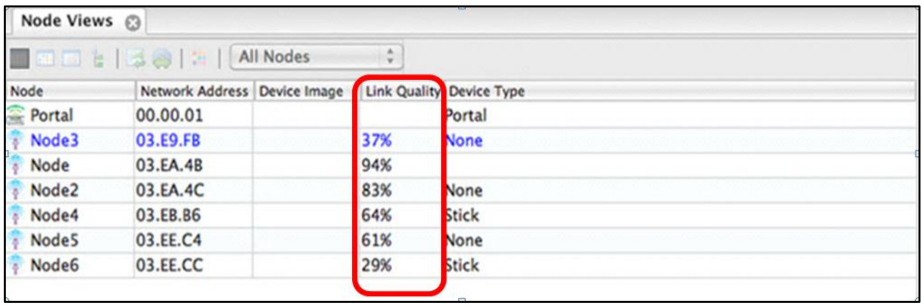


Figure 5.9: RF Engine network address

5.5 Distance VS. Signal Strength RSSI

The primary test of the experiment was designed to illustrate the signal strength of each sensor node. This would help to know the criteria of the sensor, and the test would determine the best distance to locate the sensors in the accurate position.

The first step is monitoring the column of the Link Quality on the Portal software to record the signal strength. The Link Quality column shows a snapshot of the radio receive level by default. It is expressed as a percentage, with 0% (-95 dBm) representing the weakest possible signal and 100% (-18 dBm) representing a maximum strength signal. Figure 5.10 shows the first reading of the link quality column.



Node	Network Address	Device Image	Link Quality	Device Type
Portal	00.00.01			Portal
Node3	03.E9.FB		37%	None
Node	03.EA.4B		94%	
Node2	03.EA.4C		83%	None
Node4	03.EB.B6		64%	Stick
Node5	03.EE.C4		61%	None
Node6	03.EE.CC		29%	Stick

Figure 5.10: A snapshot of Link Quality

It is important to understand three things about the displayed Link Quality [19]:

1) Normally this field is not continuously refreshed – Portal does not "poll" nodes unless told to. The Broadcast PING button is used to update the Link quality fields of all active units. There is also a Refresh button that can be clicked to force a refresh of a single node's Link Quality. This button is on the Node Info toolbar. Finally, there is a WatchNodes button that essentially turns on automatic broadcast pings.

2) The value shown is based on the received signal strength of the most recent message from any other wireless node. It does not represent the signal strength between Portal and the node.

3) It is possible that at the time the Link Quality field was read from the unit, it had not yet received any radio messages from any other node. In this case, a value of 0 will be reported. This does not mean the unit has a faulty radio; it simply has not done any radio communications yet. This is most often seen with the node that is acting as a “bridge” for Portal, because Portal can be interacting with this directly attached node without necessarily generating any **radio** traffic.

The second step was to install the ten sensor nodes in their location, and then this would show on the portal interface, as shown in Figure 5.11.The data collected is from the portal and presented on the table 5.4.

Table 5.4: Distance and signal from Main node

Node Number	Distance From The Main	Signal Strength
node 1	108.89 M	84%
node 2	142.61 M	80%
node 3	190.96 M	78%
node 4	270.82 M	72%
node 5	273.78 M	70%
node 6	258.45 M	61%
node 7	312.34 M	58%
node 8	319.00 M	45%
node 9	381.89 M	41%
node 10	506.23 M	36%

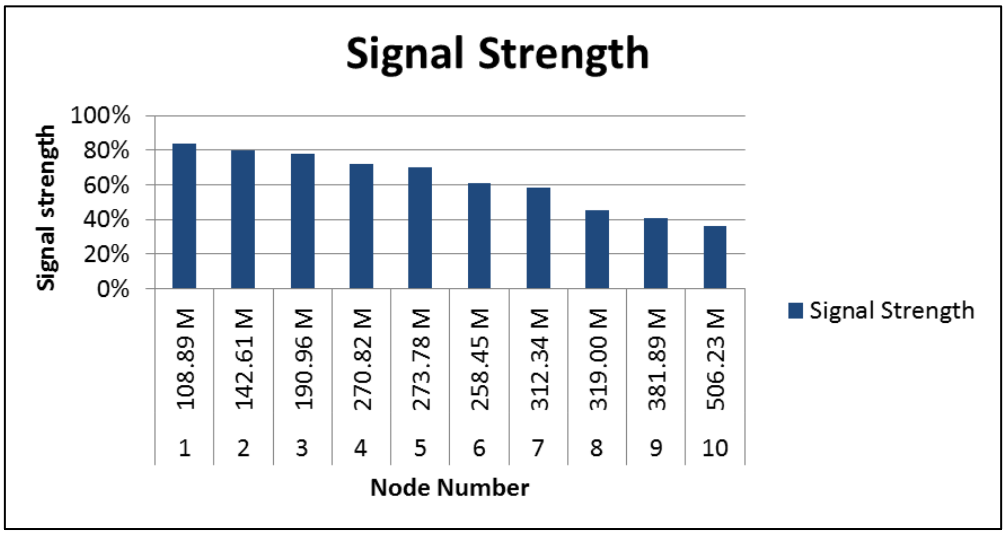


Figure 5.11: Signal Strength from Main node

By adding the data from the portal software to the distance of each sensor node, the results shown in Table 4.4 and Figure 5.11 were gained. The result shows that the sensor node number (1, 2, 3) are the nearest sensor node to the main system. As a result of that they have a stronger signal strength than the rest of the sensor nodes.

5.6 Signal Path test

The signal Path test was used to investigate the impact of the network and the signal strength for each sensor node, and if one sensor node was disabled, to locate the maximum number of routing paths for each sensor. Monitoring the link quality of each sensor node and considering the distance between each sensor node would show the possible routing paths.

If one of the ten nodes is disabled, there will not be a problem because the network topology of the RF model is based on the Mesh topology, so the signal will chose another possible path to reach the nearest node.

The first step was to disable all the sensor nodes except the sensor being tested. Then the neighbouring sensor nodes where activated respectively to see how many sensor nodes were directly connecting and how many paths each sensor node would route.

The experiment started with the main node, then node number 1 and 2 and so on until node number 10 by reading the signal strength and the distance respectively.

5.6.1Main Node

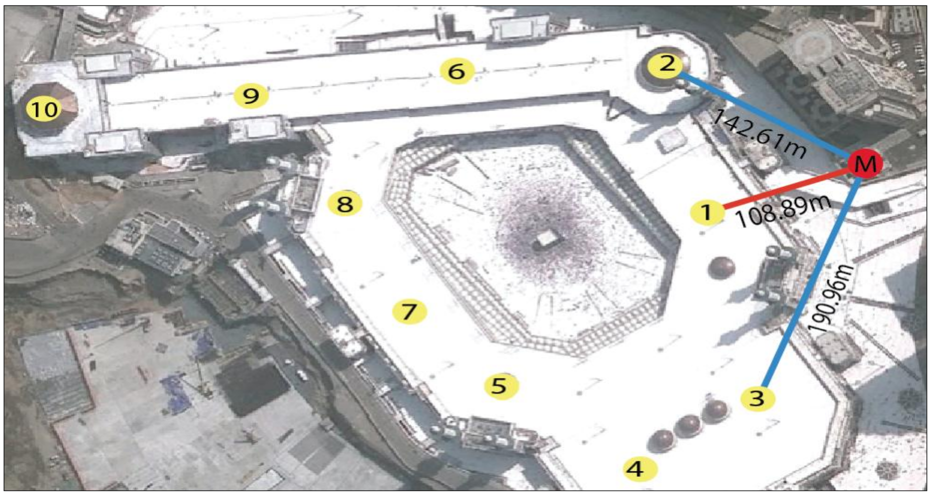


Figure 5.12: Distance from Main node

Table 5.5: Distance and signal from Main node

Node Number	Distance From The Main	Signal Strength
node 1	108.89 M	97%
node 2	142.61 M	90%
node 3	190.96 M	88%

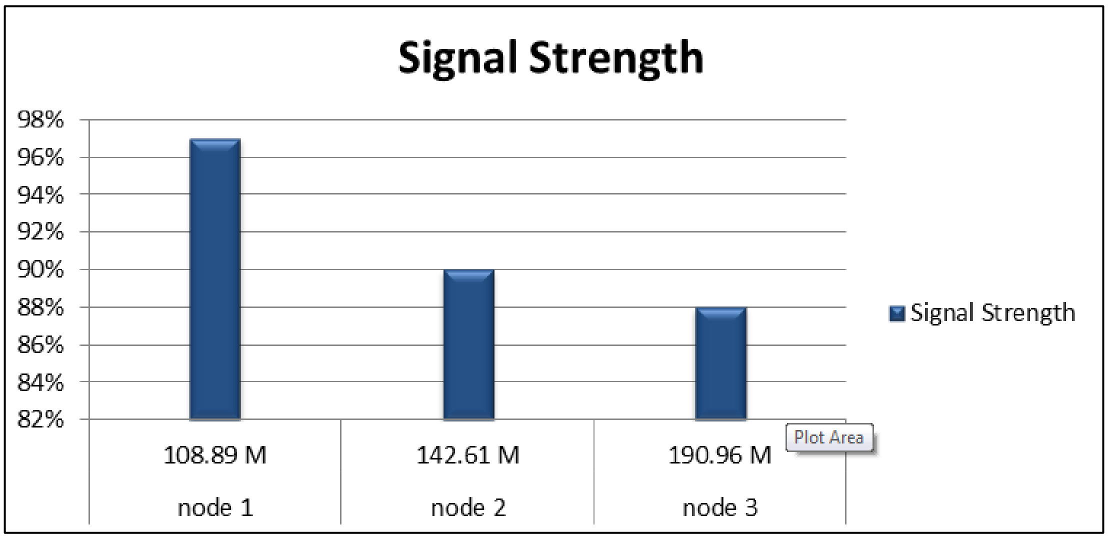


Figure 5.13: Signal strength from Main node

5.6.2Sensor Node One

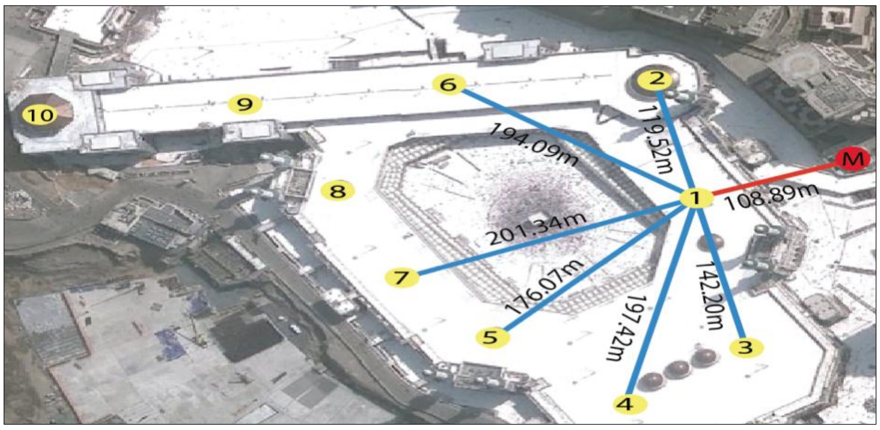


Figure 5.14: Distance from node 1

Table 5.6: Distance and signal from node 1

Node Number	Distance From Node 1	Signal Strength
Main	108.89 M	97%
Node 2	119.52 M	90%
Node 3	142.20 M	88%
Node 4	197.42 M	77%
Node 5	176.07 M	86%
Node 6	194.09 M	73%
Node 7	201.34 M	62%

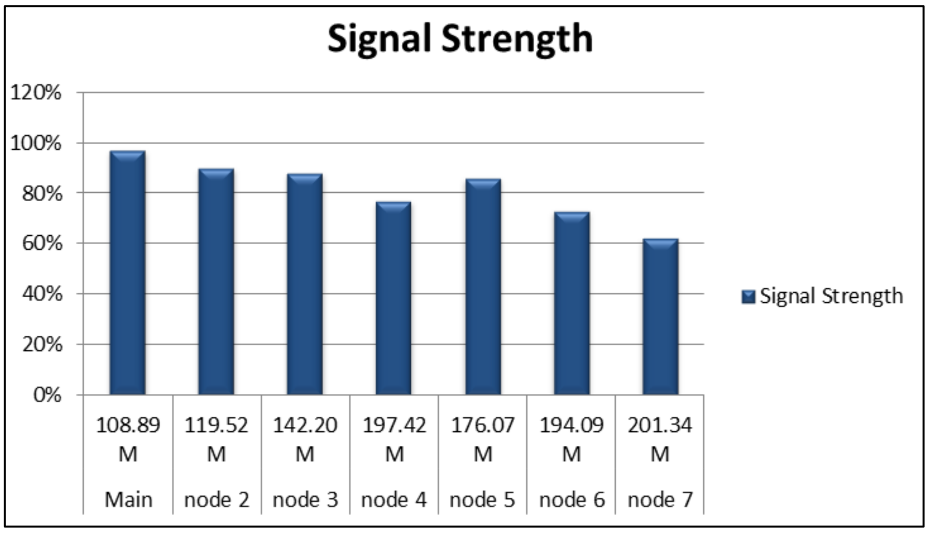


Figure 5.15: Signal strength from node 1

5.6.3Sensor Node Two

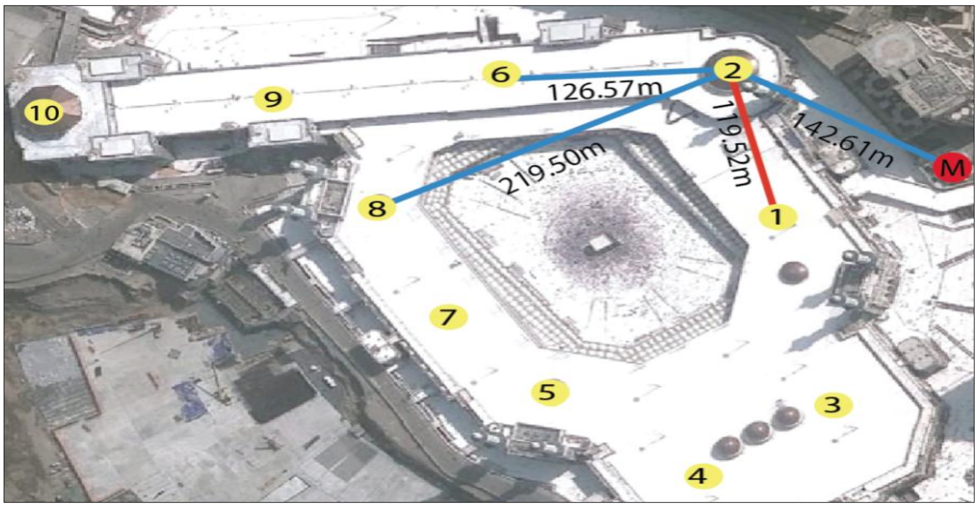


Figure 5.16: Distance from node 2

Table 4.7: Distance and signal from node 2

Node Number	Distance From Node 2	Signal Strength
Main	**142.61m**	**90%**
Node 1	**119.52**	**94%**
Node 6	**126.57**	**90%**
Node 8	**219.50**	**45%**

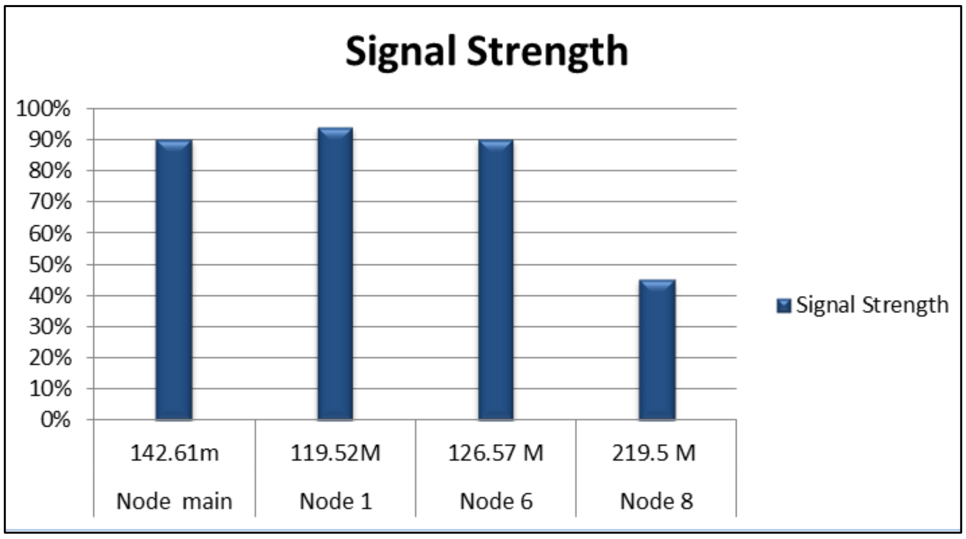


Figure 5.17: Signal strength from node 2

5.6.4Sensor Node Three

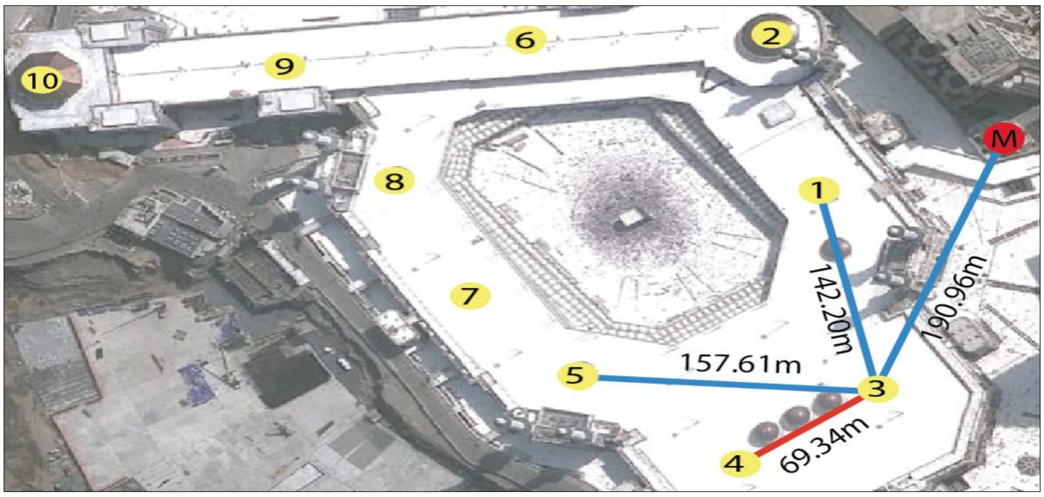


Figure 5.18: Distance from node 3

Table 4.8: Distance and signal from node 3

Node Number	Distance From Node 3	Signal Strength
Node 1	142.20 M	88%
Main	190.96 M	79 %
Node 4	69.34 M	97 %
Node 5	157.61 M	93%

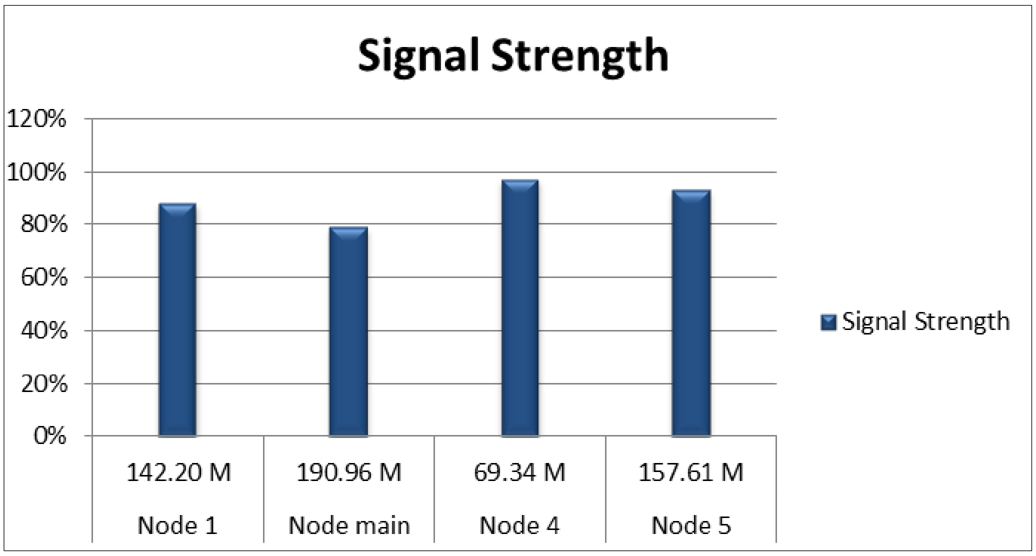


Figure 5.19: Signal strength from node 3

5.6.5Sensor Node Four

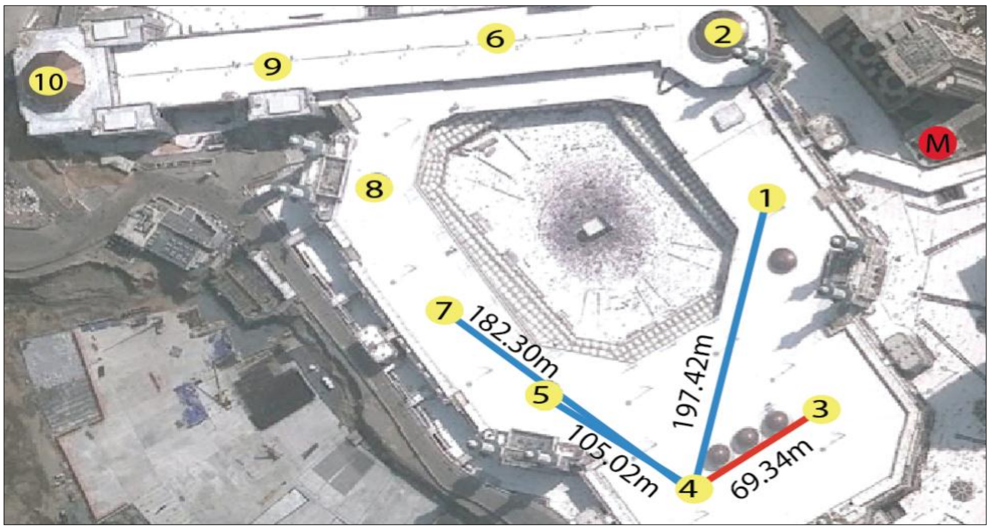


Figure 5.20: Distance from node 4

Table 5.9: Distance and signal from node 4

Node Number	Distance From Node 4	Signal Strength
Node 1	**197.42 M**	**57%**
Node 5	**105 M**	**87%**
Node 3	**69.34 M**	**95%**
Node 7	**182.30 M**	**76%**

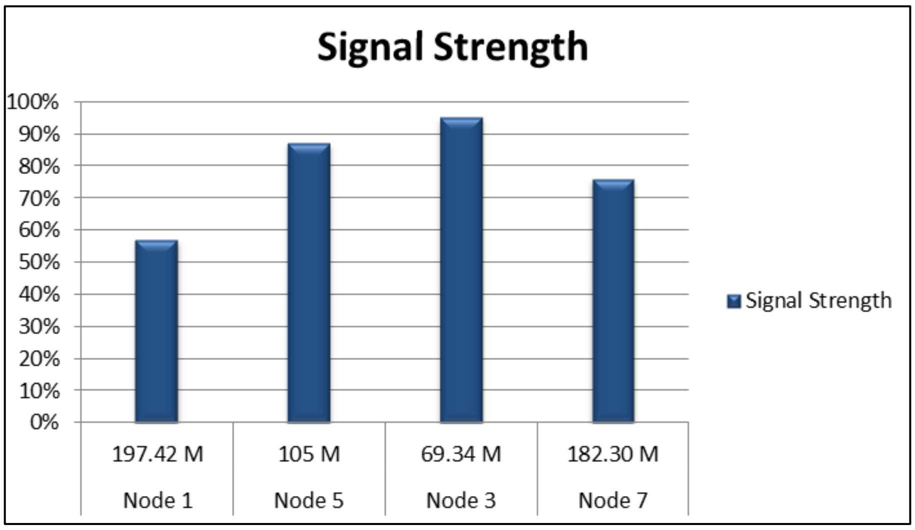


Figure 5.21: Signal strength from node 4

5.6.6Sensor Node Five

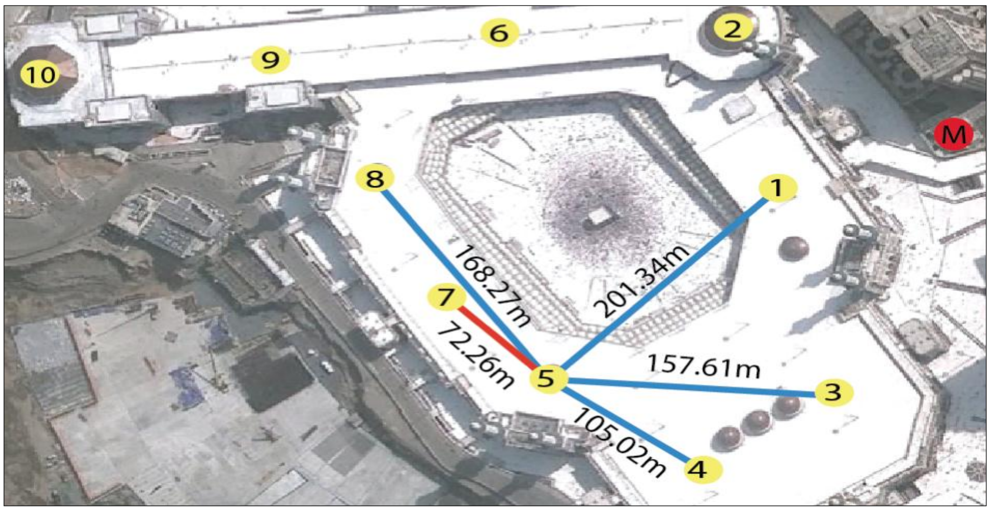


Figure 5.22: Distance from node 5

Table 5.10: Distance and signal from node 5

Node Number	Distance From Node 5	Signal Strength
Node 1	201.34 M	44%
Node 4	105.02 M	87%
Node 3	157.61 M	75%
Node 7	72.26M	94 %
Node 8	168.27 M	68%

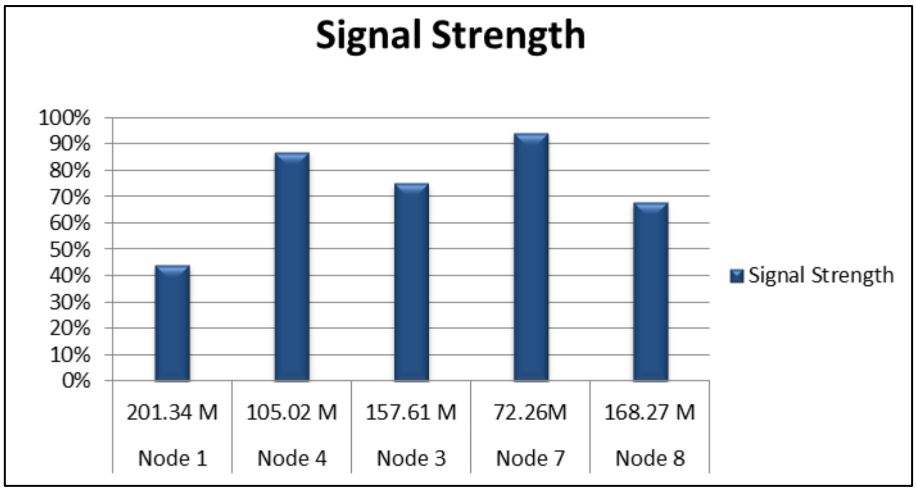


Figure 5.23: Signal strength from node 5

5.6.7Sensor Node Six

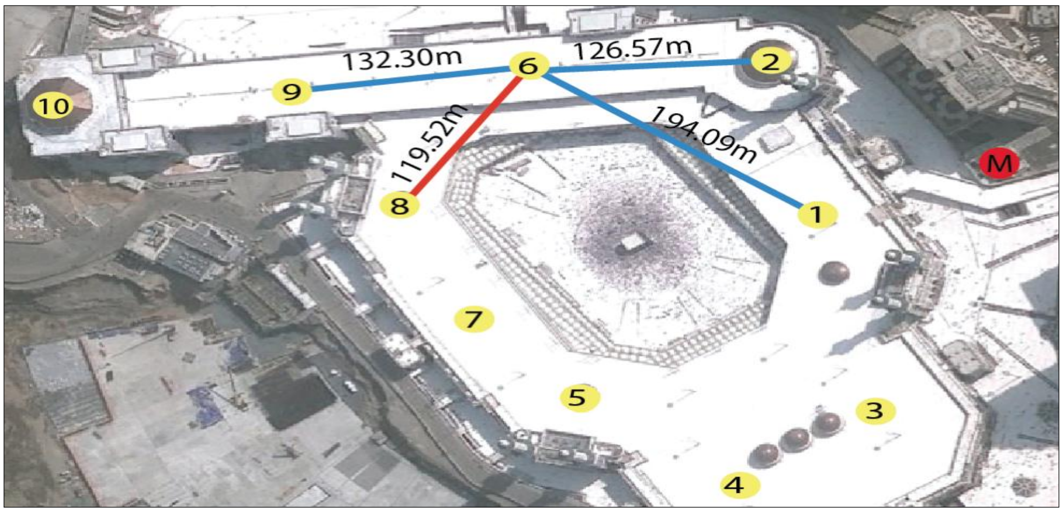


Figure 5.24: Distance from node 6

Table 5.11: Distance and signal from node 6

Node Number	Distance From Node 6	Signal Strength
Node 1	149.09 M	63%
Node 2	126.57 M	76%
Node 9	132.30 M	74%
Node 8	119.71 M	88%
Node 7	187.45M	54%

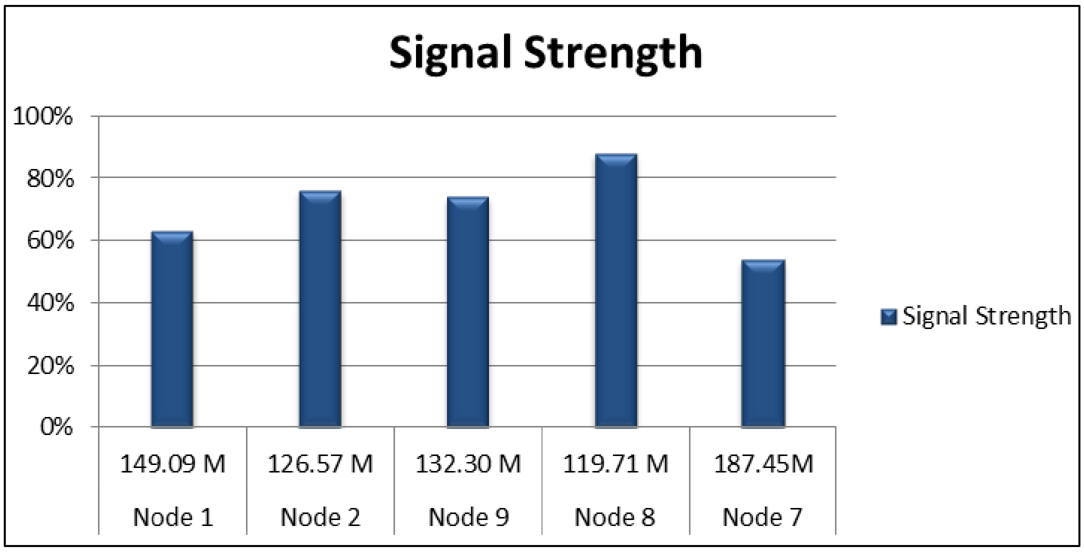


Figure 5.25: Signal strength from node 6

5.6.8Sensor Node Seven

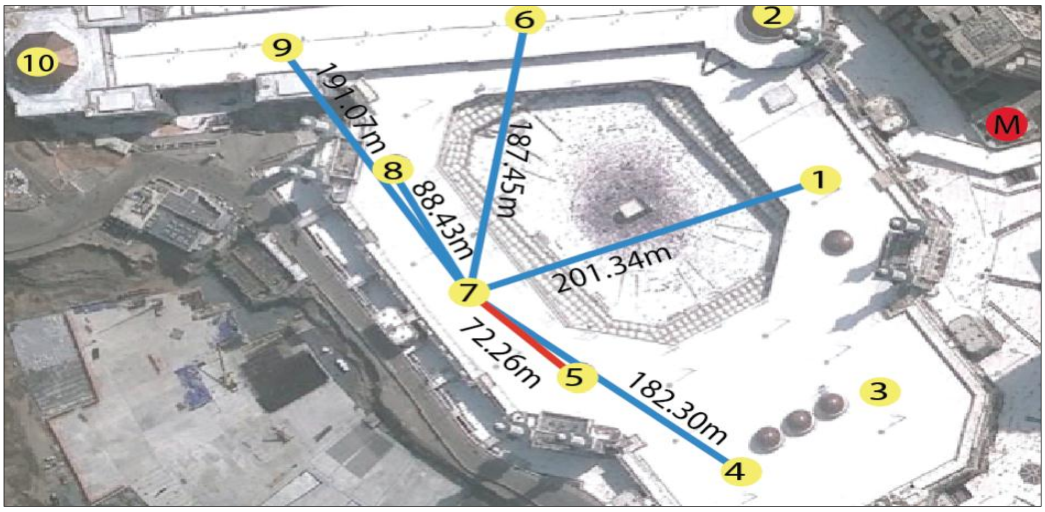


Figure 5.26: Distance from node 7

Table 5.12: Distance and signal from node 7

Node Number	Distance From Node7	Signal Strength
Node 1	201.34 M	24%
Node 5	72.26 M	96%
Node 4	182.30 M	42%
Node 6	187.45M	39%
Node 8	88.43 M	93%
Node 9	191.07 M	34%

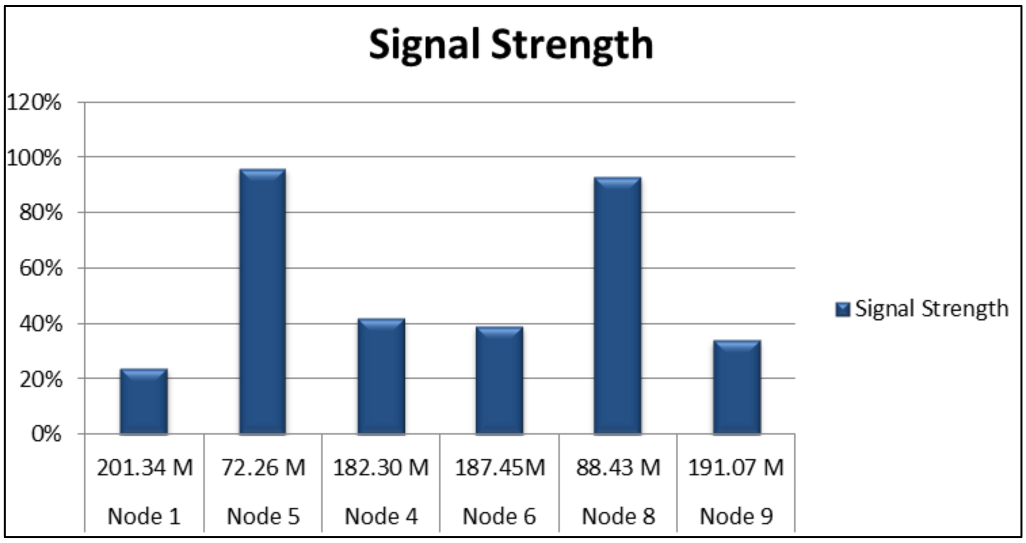


Figure 5.27: Signal strength from node 7

5.6.9Sensor Node Eighth

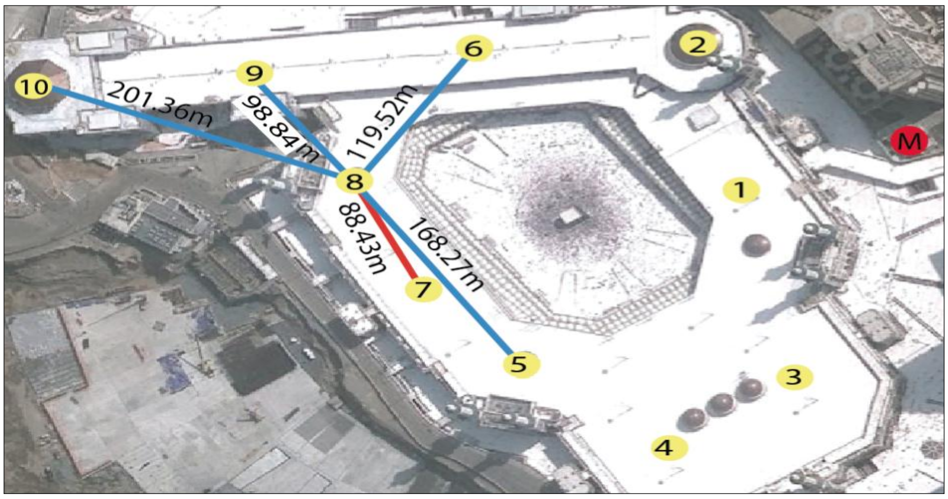


Figure 5.28: Distance from node 8

Table 5.13: Distance and signal from node 8

Node Number	Distance From Node 8	Signal Strength
Node 10	201.36 M	27%
Node 9	98.84 M	89%
Node 6	119.52 M	68%
Node7	88.43M	93%
Node 5	168.27M	45%

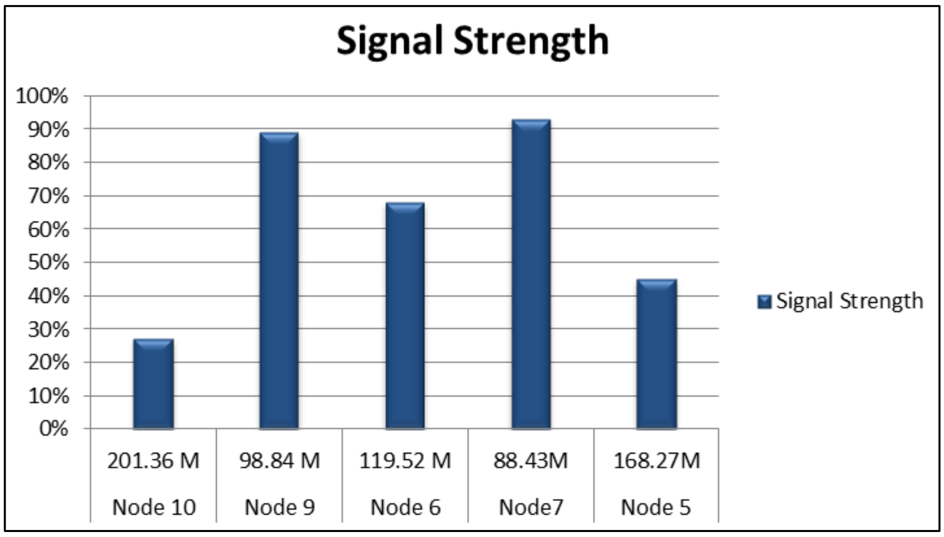


Figure 5.29: Signal strength from node 8

5.6.10 Sensor Node Nine

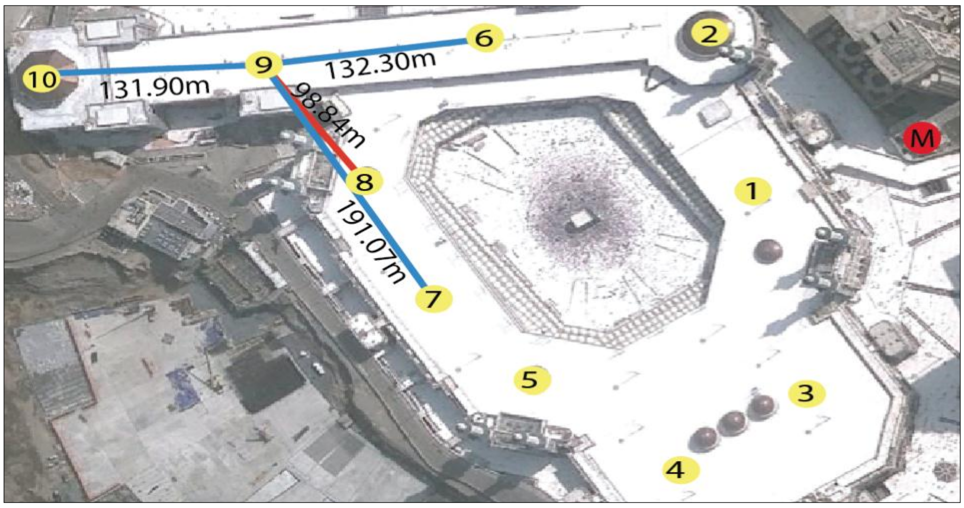


Figure 5.30: Distance from node 9

Table 5.14: Distance and signal from node 9

Node Number	Distance From Node 9	Signal Strength
Node 10	131.90 M	55%
Node 8	98.84 M	89%
Node 6	132.30 M	55%
Node 7	191.07 M	34%

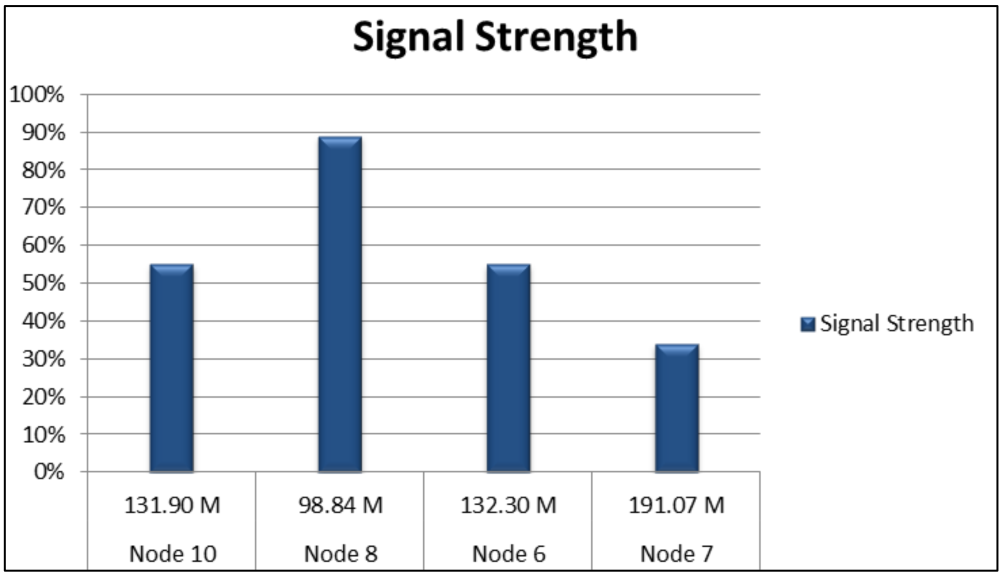


Figure 5.31: Signal strength from node 9

5.6.11 Sensor Node Ten

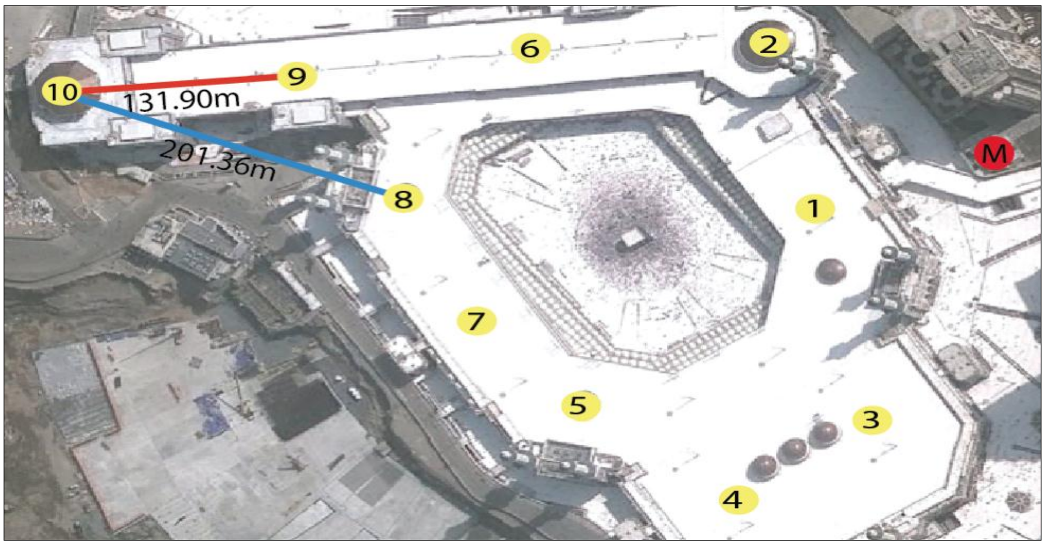


Figure 5.32: Distance from node 10

Table 5.15: Distance and signal from node 10

Node Number	Distance From Node 10	Signal Strength
Node 9	**131.90M**	**55%**
Node 8	**201.36 M**	**28%**

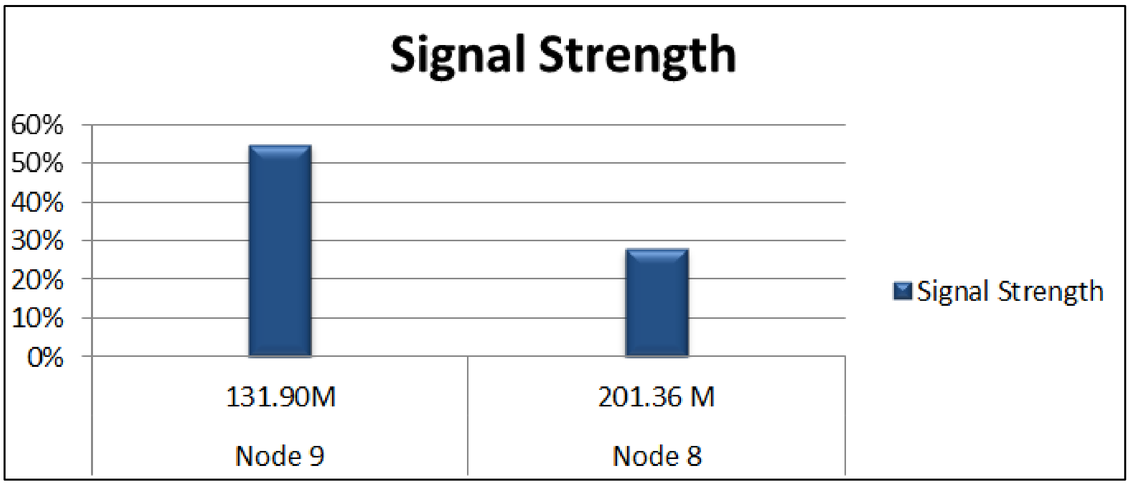


Figure 5.33: Signal strength from node 10

5.7 Discussion on the signal paths test

Table 5.16, below shows all the data collected starting with the main node through to node number 10, and showing the number of routing paths of each sensor node and which nodes they are connecting with.

The table also shows that the Main sensor has three paths to deliver or receive a signal through Nodes number 1, 2, and 3.

The sensors one, two and three are considered as the key nodes in this network. These three sensor nodes were selected in order to avoid any interruption in the network, providing instant alternative paths to transmit the signal and ensure delivery of the signal to the farthest node in the network.

Table 5.16: Number of paths

NODE	How many Paths	Nodes Connecting
Main Node	**3**	**1, 2, 3**
1	**7**	**Main, 2, 3, 4, 5, 6, 7**
2	**4**	**Main, 1, 6, 9**
3	**4**	**Main, 1, 4, 5**
4	**4**	**3, 1, 5, 7**
5	**5**	**1, 3, 4, 8, 7**
6	**5**	**9, 7, 1, 8, 2**
7	**6**	**1, 8, 9, 5, 6, 4**
8	**5**	**10, 9, 6, 7, 5**
9	**5**	**2, 6, 10, 8, 7**
10	**2**	**9, 8**
Total	**50**	**50**

It is clear that sensor nodes number 1 and 7 possess the largest number of paths to connect with the rest of sensor nodes: sensor number 1e has seven paths and sensor number 7 has six paths. There are two obvious reasons for node 1, and 7 to have more paths than other nodes: Firstly, both nodes are lying directly with Main Node.Secondly, they also situated in the middle of the network map.

The data from the table shows the remaining sensors, numbers 2, 3, 4, 5, 6, 8, 9, are sharing the numbers four and five paths in connection within the network, because it is located on the farthest point of the network map from the main sensor node, sensor number 10 has only two paths to connect and access, either through sensors eight or nine. Finally, data on the table presents a confident result which shows that for network of ten sensor nodes has 50 paths to connect and transmit signals.

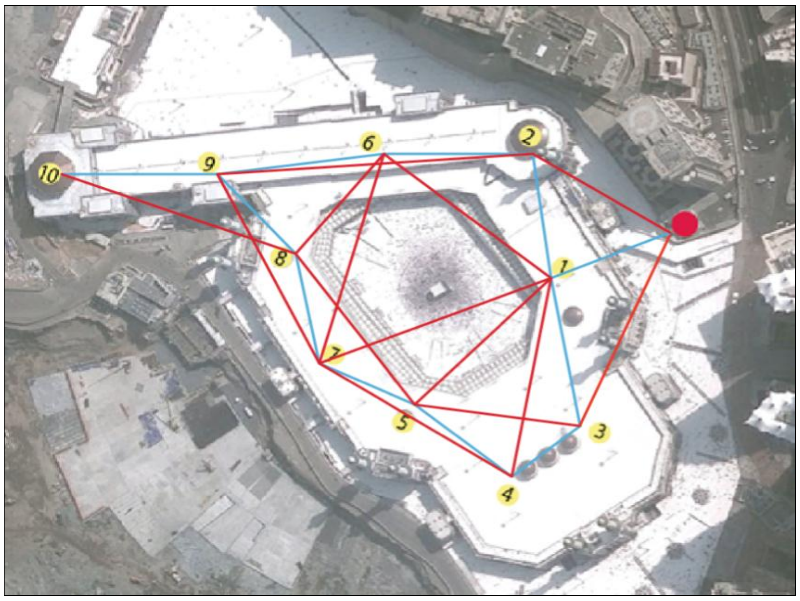


Figure 5.34: The Strongest Path

In conclusion Figure 5.34, show the network map of the experiment in Mecca , the red line shows all the signal paths connection between nodes sending or receiving. However; the blue line is the best signal path on the network.

5.8 Battery life

The sensor nodes are powered by electricity by using the power supplies cables with input voltage AC 110--240V and output voltage DC 9V Output Current: 500mA. These are connected to a protoboard, which also has a power regulator option and battery connector. Tools for the experiment are: 2X (1.5 V) AA battery, AA Cell Battery Holder, and Digital MultiMetre.

A 3V power adapter will power the station, however, the station also contains disposable AA batteries as a backup power source in case of power failure. This test investigated the power consumption by reading the power using a digital MultiMetre. The battery life test has been done in two stages: the first stage was collecting the voltage reading evrey 30 minutes and monitoring the signal quality and the second stage was collecting the voltage reading each hour,.as shown in Table 5.17 and Figure 5.35.

Table 5.17: Battery Life during 24 hours

Time	1:00	1:30	2:00	2:30	3:00	3:30	4:00	4:30	5:00	5:30	6:00
Battery Voltage	3	3	3	2.99	2.98	2.97	2.97	2.96	2.95	2.93	2.92

(A)

Time	6:30	7:00	7:30	8:00	8:30	9:00	9:30	10:00	10:30	11:00	11:30	12:00
Battery Voltage	2.92	2.92	2.91	2.91	2.9	2.9	2.89	2.88	2.87	2.87	2.86	2.85

(B)

Time	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	24:00
Battery Voltage	2.83	2.8	2.78	2.73	2.7	2.68	2.65	2.63	2.6	2.58	2.56	2.51

(C)

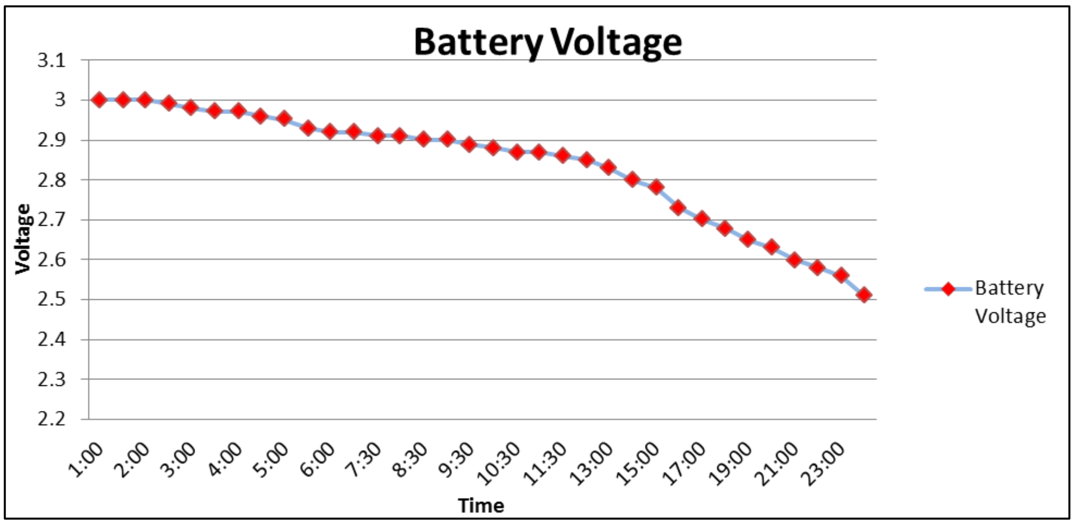


Figure 5.35: Battery Life during 24 hours

5.9 Conclusion

The experiment shows that during the 24 hours test the batteries are working very well as a backup source of energy. The two stages of the experiment indicated that the energy was consumed gradually, while the signal was retained in the same normal level of quality, with all the sensor nodes consuming what they needed to continue working.

CHAPTER 6

SUMMARY OF FINDINGS AND DISCUSSION

CHAPTER 6 SUMMARY OF FINDINGS AND DISCUSSION

6.1 Introduction

This chapter presents the final outcome and the main findings of the project and will discuss the results obtained from the project's experiments. Then it will summarise and discuss the idea of the project, and its contribution to solving the problem of the urgent need for a system to help missing people, particularly children, women, elderly people, and patients who are in need of a medical or security services during the Hajj season. Finally, it will discuss the actual implementation of the project in the Grand Mosque in Mecca and its potential effectiveness.

6.2 The Project's prototype

Figure 6.1 present the Project's prototype. There are two switches. The red switch is for the Medical alert, the green is for the security alert. The battery is included in the box.

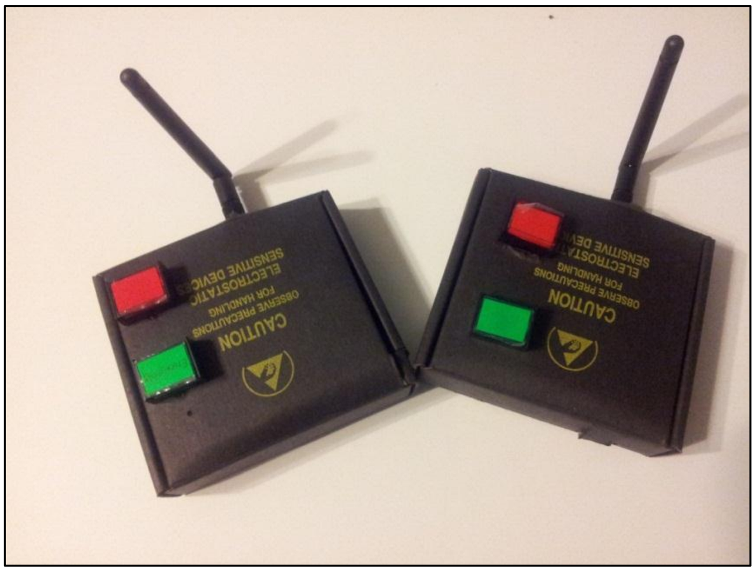

CAUTION
OBSERVE PRECAUTIONS FOR HANDLING
ELECTROSTATIC SENSITIVE DEVICES

Figure 6.1: The Project's prototype

6.3 Project’s hardware and Software

6.3.1 Hardware

Choosing the hardware was the core of this project. The selection was based on four main features which are the latest technologies, range, price, and power consumption.

Therefore the Synapse RF Engine module was chosen to be the backbone of the project. It is based on an IEEE 802.15.4 standard, low power, highly-reliable solution to embedded wireless control, monitoring network, and wireless mesh network operating system with an integrated transceiver radio data rate up to 2 Mbits/sec. The Synapse RF Engine module is a low-cost module, with a range of up to 5 Km, and a power consumption as low as 1.6 µA to enable a new generation of battery-driven systems.

6.3.2 Software

Various levels of program have been used in the project, each with specific objectives for the system operation including:

- Python programming language. This has been used to set up the codes in the station nodes.
- C# language. This has been used to design a friendly a graphical user interface (GUI) that enables users to access the network map of the project to monitor the station nodes on the real implementation.
- Synapse Portal software. This is a graphical user interface that enables users to access the RF engines. Via Synapse Portal, users can upload Python codes and monitor the activities of RF engines.
- SNAPconnect software. This works as a middleware between the bridge and the GUI software.

6.4 Field Experiment

The field experiment is considered to be the most important stage to do the rest of the tests. It contributes to the initial planning to draw the network map.

Field Experiment sites were selected by considering the area of the Grand Mosque in Mecca. Victoria Esplanade Garden in the city of Palmerston North, New Zealand, has been determined using Google Earth and it was excellent site to test the system.

6.5 Positioning of the sensors (draw the map network)

Positioning the sensors is a key step for building the main form of the network map. After selecting the field experiment, site images were dragged from Google Earth to study and draw the network map and position the sensor nodes. The distance between each sensor node is not more than 300 meters. Note that this trial is based on ten sensors only. The network map has been matched with the map of the Grand Mosque of Mecca, and the physical location of where the devices will be placed has been identified. On this basis the system is ready for implementation in Mecca.

6.6 Tools of the Experiment

Google Earth was used as a major tool to identify the places by tacking images and meters measure also used for measuring the distances in the actual place of execution, batteries have been used to provide energy to run the sensors.

6.7 Results and Discussion

After the sensor nodes were installed in the suitable site, the results were gained from the tests as explained in chapter four. The following points summarise the outcome of the tests:

- The sensor nodes are working steadily and smoothly according to the area of the grand mosque of Mecca.
- The sensor nodes are installed and located at an appropriate distance according to RF Engine's technical features.
- There were no external effects on the sensor node signals, such as electricity, mobile phone signals, or Wi-Fi signals surrounding the site.
- The results of the path signal present encouraging and a satisfactory result which shows that a network of ten sensor nodes has 50 paths to connect and transmit signals.
- All the sensor nodes are consumed what was needed to continue their work. During the 24 hours the batteries worked well as a backup source of energy.

The overall outcomes suggest that the system is ready to provide the services to the pilgrims in the Grand mosque of Mecca. The existence of adding more sensor nodes in the mosque will increase the signal strength and provide a better service to a larger number of pilgrims in the Grand mosque of Mecca.

CHAPTER 7

Conclusion and Future Work

CHAPTER 7 Conclusion and Future Work

The main problem that is faced by the pilgrims in Mecca every year is that the areas of Mecca become overcrowded in the month of Hajj and the number of missing people in the Grand Mosque area has increased. This overcrowding has raised concerns over the safety and the security of pilgrims.

By exploring the WSN technology we can conclude that there are many different WSN applications available. A wide range of applications for WSN have been implemented in different areas such as health, military, factories, oil industries, security and home automation. WSN technology also can be used to help solving the overcrowding problem in Mecca during the Hajj season.

This thesis focuses on the problems of missing people and helping those in need of urgent medical services.

The proposed solution is to take advantage of the modern technology by building a fixed station nodes using WSN within the corridors of the Grand Mosque in Mecca to help the pilgrims with security and medical services. The places for services are known in advance and set out for the pilgrims in case he or she wants a help or to make a query about any problem that could occur during Hajj time. It is a reliable and affordable project to implement in the Grand Mosque of Mecca. This project is very important and represents a solution for emergency support during Hajj.

The Synapse RF Engine module was targeted to be the main device of the project. Its selection was based on four main features which are latest technologies, range, price, and power consumption. Various levels of programs have been used in the project each with specific objectives for the system operation, for example, python programing language has been used to set up the codes of the two switches for the medical and security switches on the RF Engine node, C# language in correlation with the XML\RPC function library was used to design the GUI to monitor the status of the station nodes and to obtain wide information from each station node. Also, the simulation tools of OPNET Modeller simulator have been used to expect the network behaviour under different conditions.

The field experiment was an excellent site to test the system, and the sensor nodes were installed in suitable sites. The results were gained from the tests of the path signal present a confidence and satisfactory results.

The backup source of energy of the entire sensor nodes has been supplied over 24 hours and the results show that the batteries are working very well, where the sensor nodes are consumed what its need to continue its work.

It is recommended that the base station includes, the PC unit, situated in the chamber of the security operations of the Grand Mosque, to follow-up the main graphical user interfaces of the project by tracking the status of the station nodes and to ensure effective communication with the security guards in the Grand Mosque yards. This will increase the capabilities of quick responding to every alert, and provide the right service to the situation, whether it is security or medical.

The external design of the stations in the real implementation is expected to be equipped and prepared with the necessary signals and instructions in different languages to help the pilgrims requesting the service.

Finally, it can be concluded that the system is ready for implementation in Mecca. However, there is a consideration for real-life implementation, due to the size of the Grand Mosque, including four floors with more than 70 exits and entry doors, and the limited number of sensor nodes in this project. However, increasing the number of sensor nodes will increase the validity and the reliability of the project implementation. Figure 7.1 and 7.2 below present the over view maps of the distribution of the sensor nodes after increasing the number in all the four floors:

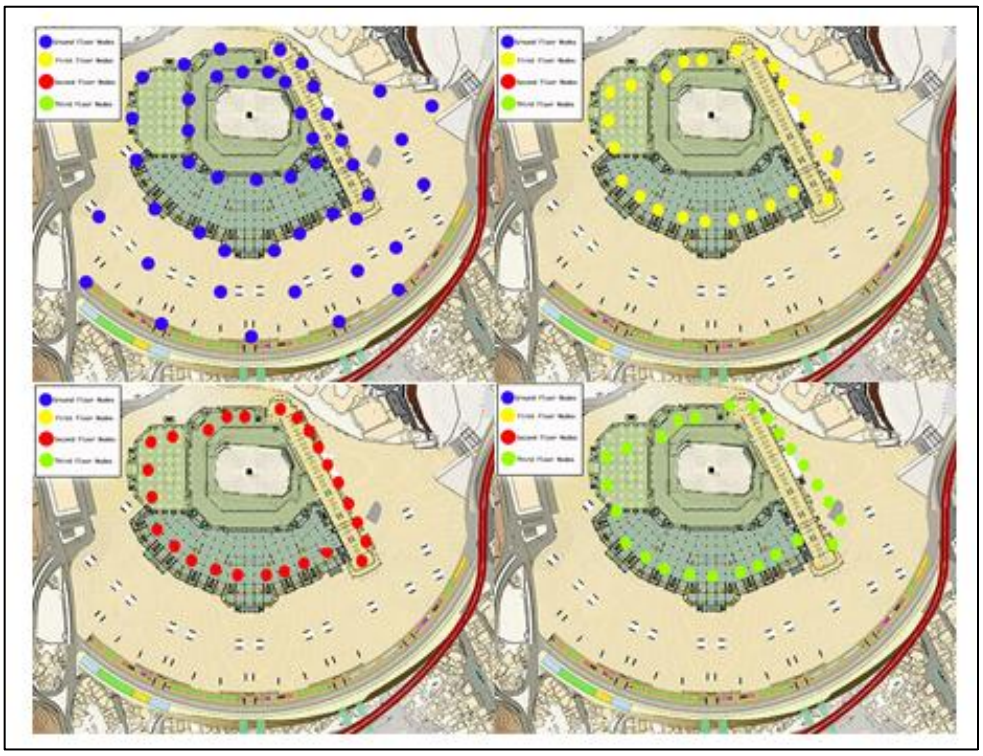


Figure 7.1: The nodes distribution in the four floors Map of The Grand Mosque

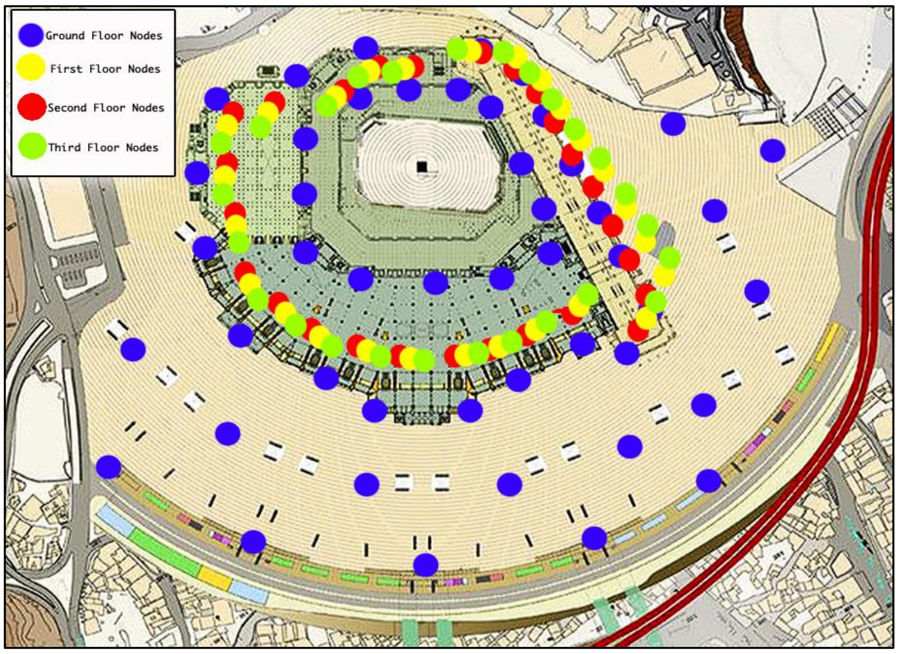


Figure 7.2: The nodes distribution in the four floors Map of The Grand Mosque

CHAPTER 8 REFERENCES

[1] The Ministry of Hajj (2011). *Location*. Available: http://www.hajinformation.com/main/h101.htm

[2] The Ministry of Hajj (2011). *Largest ever expansion of the Haram in Makkah (2008)*. Available: http://www.hajinformation.com/main/j103.htm

[3] S. A. Hameed, "ICT to serve Hajj: Analytical study," in *Computer and Communication Engineering (ICCCE), 2010 International Conference on*, 2010, pp. 1-7.

[4] Central Department Of Statistics & Information in SA (2010). *The number of pilgrims for the Years From (1995G.) to (2010G.)* .. Retrieved from http://www.cdsi.gov.sa/english/index.php?option=com_docman&Itemid=173

[5] T. Mantoro, A. D. Jaafar, M. F. M. Aris, and M. A. Ayu, "HajjLocator: A Hajj pilgrimage tracking framework in crowded ubiquitous environment," in *Multimedia Computing and Systems (ICMCS), 2011 International Conference on*, 2011, pp. 1-6.

[6] The Ministry of Health of the Kingdom of Saudi Arabia (2009). Health Services in Hajj Season. In *Health Statistical Year Book*. Saudi Arabia: Retrieved from http://www.moh.gov.sa/en/Ministry/Statistics/book/Pages/default.aspx

[7] Stankovic, J. A. (2006). Wireless Sensor Networks. Department of Computer Science, University of Virginia, 1-20.

[8] Townsend, C., & Arms, S. (2004). Wireless Sensor Networks: Principles and Applications, MicroStrain, Inc., 439-449.

[9] E. Ferro and F. Potorti, "Bluetooth and Wi-Fi wireless protocols: a survey and a comparison," *Wireless Communications, IEEE,* vol. 12, pp. 12-26, 2005.

[10] J. S. Lee, Y. W. Su, and C. C. Shen, "A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi," in *33rd Annual Conference of the IEEE Industrial Electronics Society (IECON), Taipei, Taiwan, November 5-8, 2007 B2 - 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON), Taipei, Taiwan, November 5-8, 2007 (November 05, 2007-November 08, 2007)*, ed, 2007, pp. 46-51.

[11] J. S. Lee, "Performance evaluation of IEEE 802.15.4 for low-rate wireless personal area networks," *Consumer Electronics, IEEE Transactions on,* vol. 52, pp. 742-749, 2006.

[12] R. Sokullu, M. A. Akkas, x, and H. E. etin, "Wireless Patient Monitoring System," in *Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on*, 2010, pp. 179-184.

[13] C. Ho and Z. Weihua, "Bluetooth-enabled in-home patient monitoring system: Early detection of Alzheimer's disease," *Wireless Communications, IEEE,* vol. 17, pp. 74-79, 2010.

[14] Al-Salman, M. (2010). Group of Pilgrims Monitoring by using Wireless Sensor Networks. Imam Muhammad Bin Saud Islamic University College of Computer and Information Sciences, Department of Computer Science, 4-62.

[15] M. Mohandes, "An RFID-based pilgrim identification system (a pilot study)," in *Optimization of Electrical and Electronic Equipment, 2008. OPTIM 2008. 11th International Conference on*, 2008, pp. 107-112.

[18] "TECHNICAL MANUAL SNAP Connect v1.0," ed: SYNAPSE WIRELESS INC, 2008.

[19] "REFERENCE MANUAL Portal for Version 2.2," ed: SYNAPSE WIRELESS INC, 2009.

[20] Y. Sunghyun and K. Young Boo, "A Design of Network Simulation Environment Using SSFNet," in *Advances in System Simulation, 2009. SIMUL '09. First International Conference on*, 2009, pp. 73-78.

[21] S. Lohier, A. Rachedi, E. Livolant, and I. Salhi, "Wireless Sensor Network simulators relevance compared to a real IEEE 802.15.4 Testbed," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, 2011, pp. 1347-1352.

[22] Pes, x030C, ovic, x, U., J. Mohorko, uc, and Z. ej, "Upgraded OPEN-ZB 802.15.4 simulation model," in *ELMAR, 2010 PROCEEDINGS*, 2010, pp. 281-284

[23] C. Xianhui, D. K. Hunter, and I. D. Henning, "Switched optical star-topology network with edge electronic buffering and centralized control," in *Communications and Networking in China, 2008. ChinaCom 2008. Third International Conference on*, 2008, pp. 243-246.

[24] H. Dilum Bandara and A. P. Jayasumana, "An enhanced top-down cluster and cluster tree formation algorithm for Wireless Sensor Networks," in *Industrial and Information Systems, 2007. ICIIS 2007. International Conference on*, 2007, pp. 565-570.

[25] D. Stevanovic and N. Vlajic, "Performance of IEEE 802.15.4 in wireless sensor networks with a mobile sink implementing various mobility strategies," in *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on*, 2008, pp. 680-688.

[26] A. M. Yasin, F. H. Yusoff, M. A. M. Isa, and N. H. M. Zain, "Avatar implementation in virtual reality environment using situated learning for &#x201C;sa'i&#x201D; (muslim hajj ritual)," in *Educational and Information Technology (ICEIT), 2010 International Conference on*, 2010, pp. V2-286-V2-290.

[27] S. Dhawan, "Analogy of Promising Wireless Technologies on Different Frequencies: Bluetooth, WiFi, and WiMAX," in *Wireless Broadband and Ultra Wideband Communications, 2007. AusWireless 2007. The 2nd International Conference on*, 2007, pp. 14-14.

[28] D. Cassioli, R. Giuliano, and F. Mazzenga, "Performance evaluation of high data rate UWB systems based on IEEE 802.15.3," in *Ultra-Wideband, 2005. ICU 2005. 2005 IEEE International Conference on*, 2005, pp. 678-683.

[29] H. Dae-Man and L. Jae-Hyun, "Smart home energy management system using IEEE 802.15.4 and zigbee," *Consumer Electronics, IEEE Transactions on,* vol. 56, pp. 1403-1410, 2010.

[30] I. V. Botsman, "Improvement in quality of coverage of 802.11a/b/g wireless networks," in *Microwave and Telecommunication Technology (CriMiCo), 2010 20th International Crimean Conference*, 2010, pp. 445-446.

[31] A. El Oualkadi, L. Vandendorpe, and D. Flandre, "Notice of Violation of IEEE Publication Principles<BR>System-Level Analysis of O-QPSK Transceiver for 2.4-GHZ Band IEEE 802.15.4 Zigbee Standard," in *Mixed Design of Integrated Circuits and Systems, 2007. MIXDES '07. 14th International Conference on*, 2007, pp. 469-474.

APPENDIX A - Python script

```
@setHook(HOOK_STARTUP)

def onStartup():

   # Set pin 5 as a watched input

   init(4);

   init(3);

   # Set pin 1 as an output

   setPinDir(1, True)

   writePin(1, False)


def init(pin):

   setPinDir(pin, False)

   setPinPullup(pin, True)

   monitorPin(pin, True)


@setHook(HOOK_GPIN)

def onPin(pin, isSet):

   # Report that a watched input has been triggered

   if isSet:

      if pin == 4:
        rpc('\x00\x00\x01', "security", 1)



    elif pin == 3:


        #rpc(localAddr, 'medical', 1)

        rpc('\x00\x00\x01', "medical", 1)



    else:

        print 'UNKOWN PIN'

def check():

 initVm()

 rpc('\x00\x00\x01', 'checkstations', localAddr())
```

APPENDIX B – The GUI code

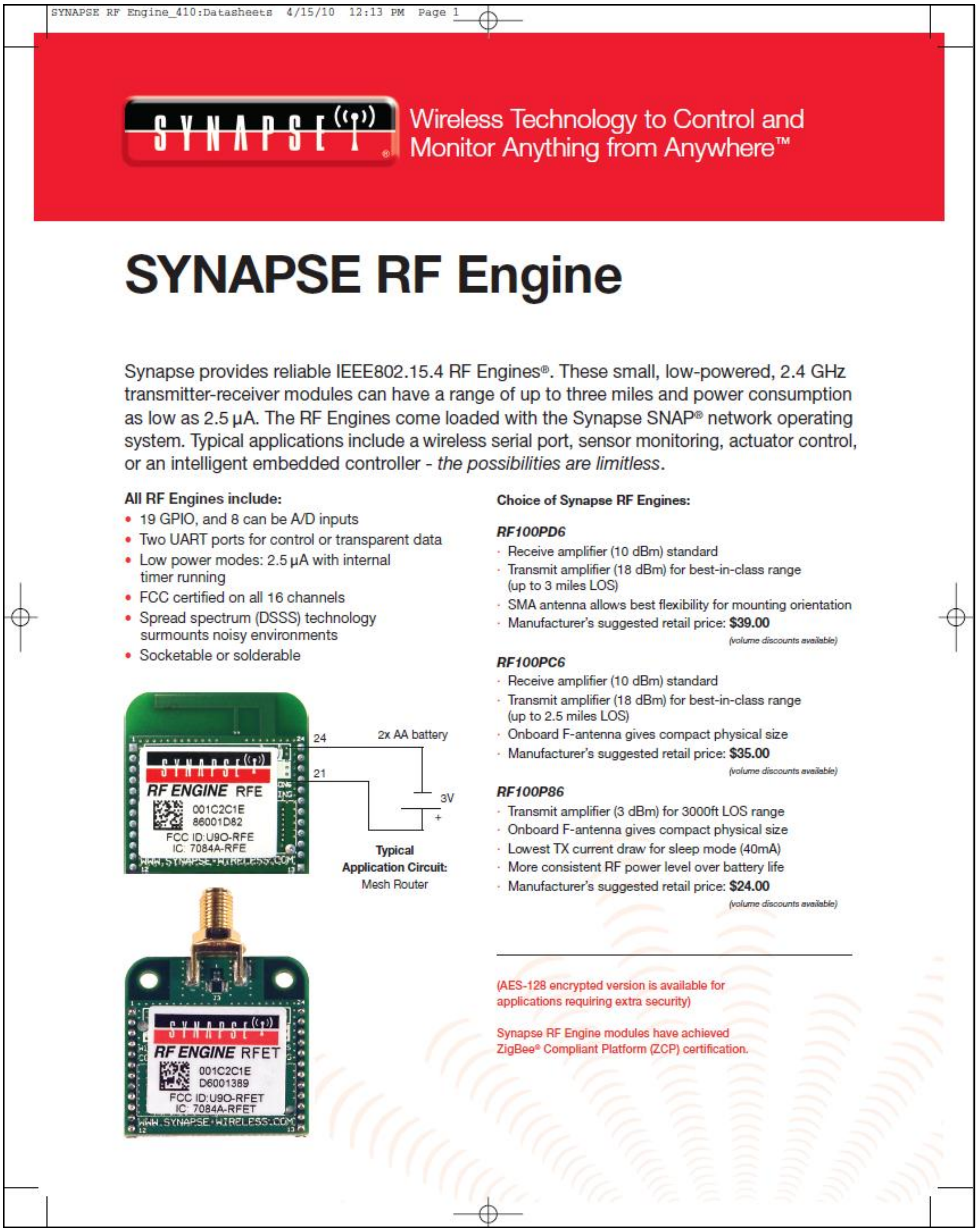
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.IO;
using System.Windows.Forms;
using CookComputing.XmlRpc;

namespace xmlrpc_client1
{
    public partial class frmMain : Form
    {
        SnapConnector proxy = XmlRpcProxyGen.Create<SnapConnector>();
        Tracer tracer = new Tracer();

        private delegate void SetTextCallback(string text);
        private delegate object GetObjectCallback();
        private delegate void SetEnabledCallback(bool state);
        private delegate void InvokeDelegate(EventResponse response);
        byte[] netAddr = { };
        double timestamp = 0;
        int[] MedicalStations= new int[10]{0,0,0,0,0,0,0,0,0,0};
        int[] SecurityStations = new int[10]{ 0, 0, 0, 0, 0, 0, 0, 0,
0, 0 };
        int[] checkStations = new int[10];


        public frmMain()
        {
            InitializeComponent();
            //Log XML-RPC packets
            //tracer.Attach(proxy);
        }

        void EventCallback(IAsyncResult asr)
        {
            bool wait = true;
            XmlRpcAsyncResult clientResult = (XmlRpcAsyncResult)asr;
            SnapConnector snapConn =
(SnapConnector)clientResult.ClientProtocol;

            try
            {
                EventResponse resp = snapConn.EndWaitOnEvent(asr);
                if (resp.methodName != null)
                {
                    //With XML-RPC.NET asynchronous responses come in
on a different thread than the GUI
```

```
                    //We need to execute these calls in the GUI
thread
                    InvokeInGuiThread(resp);
                }
                //SNAPconnect will return null on a disconnect if
there is a pending wait or if you use a timeout
            }
            catch (System.Net.WebException webEx)
            {
                switch (webEx.Status)
                {
                    case
System.Net.WebExceptionStatus.ConnectFailure:
                        SetInfoText("Unable to connect to
SNAPconnect");
                        break;
                    default:
                        SetInfoText(webEx.Message);
                        break;
                }
                wait = false;

            }
            catch (XmlRpcFaultException xmlrpcEx)
            {
                wait = false;
                // A fault was sent from SNAPconnect
                if (xmlrpcEx.FaultCode ==
SNAPconnectFaultCodes.MultipleWaitOnEvents)
                {
                    //This will happen if you call waitOnEvent
multiple times
                    //before receiving an event response back
                    SetInfoText("Cannot have multiple outstanding
calls to waitOnEvent with the same network address");
                }
                else if (xmlrpcEx.FaultCode ==
SNAPconnectFaultCodes.ConnectionClosed)
                {
                    SetInfoText("SNAPconnect's underlying connection
was closed");
                }
                else if (xmlrpcEx.FaultCode ==
SNAPconnectFaultCodes.NoConnection)
                {
                    SetInfoText(xmlrpcEx.FaultString);

                }
                else
                {
                    SetInfoText(xmlrpcEx.FaultString);
                }
            }
            catch (Exception ex)
            {
                // handle the exception
            }
            if (wait)
            {
                proxy.BeginWaitOnEvent(this.getSnapNetAddr(),
EventCallback);
            }

        }

        void InvokeInGuiThread(EventResponse response)
        {
            // InvokeRequired required compares the thread ID of the
            // calling thread to the thread ID of the creating
thread.
            // If these threads are different, it returns true.
            if (this.InvokeRequired)
            {
                if (Disposing == false)
                {
                    try
                    {
                        this.Invoke(new
InvokeDelegate(InvokeInGuiThread), new object[] { response });
                    }
                    catch (ObjectDisposedException ex)
                    {
                        MessageBox.Show(" ObjectDisposedException
Error");
                        //I hate to do this but I keep getting
objectdisposed exceptions even when checking above
                    }
                }
            }
            else if ((this.Disposing == false) && (this.IsDisposed ==
false))
            {
                netAddr = response.netAddr;
                timestamp = response.timestamp;
                try
                {
                    System.Reflection.MethodInfo info =
this.GetType().GetMethod(response.methodName);
                    if (info == null)
                    {
                        MessageBox.Show("Node " +
HexStr(response.netAddr) + " called function " + response.methodName
+ " that does not exist");
                    }
                    else
                    {
                        try
                        {
                            info.Invoke(this, response.parameters);
                        }
                        catch (System.ArgumentException argEx)
                        {
                            MessageBox.Show("Node " +
HexStr(response.netAddr) + " called function " + response.methodName
+ " with parameter types that don't match");
```

```
                        }
                    }
                }
                catch (System.Reflection.AmbiguousMatchException
matchEx)
                {
                    //Take the slower route
                    foreach (System.Reflection.MethodInfo methInfo in
this.GetType().GetMethods())
                    {
                        if (methInfo.Name == response.methodName)
                        {
                            try
                            {
                                methInfo.Invoke(this,
response.parameters);
                                break;
                            }
                            catch (System.ArgumentException argEx)
                            {
                                continue;
                            }
                        }
                    }
                }
            }
        }

        void SerialConnectCallback(IAsyncResult asr)
        {
            XmlRpcAsyncResult clientResult = (XmlRpcAsyncResult)asr;
            SnapConnector snapConn =
(SnapConnector)clientResult.ClientProtocol;
            try
            {
                bool resp = snapConn.EndConnectSerial(asr);
                if (resp == true)
                {
                    this.SetInfoText("Serial connection configured on
SNAPconnect");
                    proxy.BeginGatewayInfo(GatewayInfoCallback);
                }
                else
                {
                    this.SetInfoText("Unable to configure serial
connection on SNAPconnect");
                }
            }
            catch (System.Net.WebException webEx)
            {
                switch (webEx.Status)
                {
                    case
System.Net.WebExceptionStatus.ConnectFailure:
                        SetInfoText("Unable to connect to
SNAPconnect");
                        break;
                    default:
                        SetInfoText(webEx.Message);
                        break;
                }
            }
            catch (XmlRpcFaultException xmlrpcEx)
            {
                // A fault happend on the SNAPconnect side
            }
            catch (Exception ex)
            {
                // handle the exception
            }
        }

        void RpcCallback(IAsyncResult asr)
        {
            XmlRpcAsyncResult clientResult = (XmlRpcAsyncResult)asr;
            SnapConnector snapConn =
(SnapConnector)clientResult.ClientProtocol;
            try
            {
                bool resp = snapConn.EndRpc(asr);
                if (resp != true)
                {
                    this.SetInfoText("Unable to send RPC");
                }
            }
            catch (System.Net.WebException webEx)
            {
                switch (webEx.Status)
                {
                    case
System.Net.WebExceptionStatus.ConnectFailure:
                        SetInfoText("Unable to connect to
SNAPconnect");
                        break;
                    default:
                        SetInfoText(webEx.Message);
                        break;
                }
            }
            catch (XmlRpcFaultException xmlrpcEx)
            {
                // A fault happend on the SNAPconnect side
            }
            catch (Exception ex)
            {
                // handle the exception
            }
        }

        private void SetInfoText(string text)
        {
            // InvokeRequired required compares the thread ID of the
            // calling thread to the thread ID of the creating
thread.
            // If these threads are different, it returns true.
            if (this.txtInfo.InvokeRequired)
            {
                SetTextCallback d = new SetTextCallback(SetInfoText);
                if (Disposing == false)
                {
                    try
                    {
                        this.Invoke(d, new object[] { text });
                    }
                    catch (ObjectDisposedException ex)
                    {
                        //I hate to do this but I keep getting
objectdisposed exceptions even when checking above
                    }
                }
                d = null;
            }
            else
            {
                if ((this.Disposing == false) && (this.IsDisposed ==
false))
                    this.txtInfo.Text = text;
            }
        }

        public void myTest(int a)
        {
            //Needed so we can convert from UTC
            TimeZone localZone = TimeZone.CurrentTimeZone;
            //Convert from unix style time stamp
            DateTime time = new DateTime(1970, 1, 1, 0, 0,
0).AddSeconds(timestamp);

            SetInfoText("Success, " + a.ToString() +
                " from remote " + HexStr(netAddr) +
                " on " + localZone.ToLocalTime(time).ToString());
        }
        public void medical(int k)
        {

            //Needed so we can convert from UTC
            TimeZone localZone = TimeZone.CurrentTimeZone;
            //Convert from unix style time stamp
            DateTime time = new DateTime(1970, 1, 1, 0, 0,
0).AddSeconds(timestamp);
            string statinNetAdress = HexStr(netAddr);
            string stationNum = whichstation(statinNetAdress);
            int StationInt = Convert.ToInt32(stationNum);

            txtInfo.Text += "\r\n"+"Station " + stationNum +
                " Need medical help" + " on " +
localZone.ToLocalTime(time).ToString();
            MedicalStations[StationInt] += 1;
            mapsignal(StationInt, "alert.gif");
            for (int m = 0; m < 10; m++)
            {
                textBox1.AppendText("\r\n" +
MedicalStations[m].ToString());
            }
        }
        public void security(int k)
        {
            //Needed so we can convert from UTC
            TimeZone localZone = TimeZone.CurrentTimeZone;
            //Convert from unix style time stamp
            DateTime time = new DateTime(1970, 1, 1, 0, 0,
0).AddSeconds(timestamp);
            string statinNetAdress = HexStr(netAddr);
            string stationNum = whichstation(statinNetAdress);
            int StationInt = Convert.ToInt32(stationNum);

            txtInfo.Text += "\r\n"+ "Station " + stationNum +
                " need security help" + " on " +
localZone.ToLocalTime(time).ToString();


            SecurityStations[StationInt] += 1;
            mapsignal(StationInt, "alert2.gif");
            for (int m = 0; m < 10; m++)
            {
                textBox2.AppendText("\r\n" +
SecurityStations[m].ToString());
            }

        }


        public static string HexStr(byte[] p)
        {
            char[] c = new char[p.Length * 2];
            byte b;
            for (int y = 0, x = 0; y < p.Length; ++y, ++x)
            {
                b = ((byte)(p[y] >> 4));
                c[x] = (char)(b > 9 ? b + 0x37 : b + 0x30);
                b = ((byte)(p[y] & 0xF));
                c[++x] = (char)(b > 9 ? b + 0x37 : b + 0x30);
            }
            return new string(c);
        }

        private void btnSetupConn_Click(object sender, EventArgs e)
        {
            proxy.BeginConnectSerial((int)SerialType.RS232,
(int)4,false, this.SerialConnectCallback);


        }
        void GatewayInfoCallback(IAsyncResult asr)
        {
            XmlRpcAsyncResult clientResult = (XmlRpcAsyncResult)asr;
            SnapConnector snapConn =
(SnapConnector)clientResult.ClientProtocol;
            try
            {
                InfoResponse resp = snapConn.EndGatewayInfo(asr);
                SetInfoText(txtInfo.Text + "\r\nSNAPconnect is
running version " + resp.version);
                if (resp.connected)
                {
                    SetInfoText(txtInfo.Text + "\r\nConnected via
type " + resp.portType.ToString() + " on port " +
resp.portNum.ToString());
                    proxy.BeginWaitOnEvent(this.getSnapNetAddr(),
EventCallback);


                    SetInfoText(txtInfo.Text + "\r\nConnected and
Waiting on response...");
                }
                else
                {
                    SetInfoText(txtInfo.Text + "\r\nNOT Connected");
                }
            }
            catch (Exception ex)
            {
            }

        }

        private byte[] getSnapNetAddr()
        {
            return new byte[] { 0, 0, Convert.ToByte(1) };
        }


        private void btnPing_Click(object sender, EventArgs e)
        {

            //Send multicast PING ("vmStat") to multicast group 1
with a TTL of 5
            //use parameters 5 to node name and LQ and have responses
spread of 2 seconds
            proxy.BeginMcastRpc(this.getSnapNetAddr(), new byte[] {
0, 1 }, 5, "check", new object[] { }, RpcCallback);
            Array.Clear(checkStations, 0, checkStations.Length);
            clearmap();
        }

        public void clearmap()
        {
            foreach (Control tempCtrl in this.Controls)
            {
                PictureBox pic = tempCtrl as PictureBox;
                if (pic != null)
                {
                    if (pic.Name != "pictureBox11")
                    {
                        pic.Load("down.gif");
                    }
                }
            }
        }

        public void mapsignal(int a, string b)
        {
            string pictureBoxNum = "pictureBox" + a;
            foreach (Control tempCtrl in this.Controls)
            {
                PictureBox pic = tempCtrl as PictureBox;
                if (pic != null)
                {
                    if (pic.Name == pictureBoxNum)
                    {

                        pic.Load(b);
                    }
                }
            }
        }

        public string whichstation(string a)
        {

            switch (a)
            {
                 case "03EA4B":
                    a="1";


                 break;
                 case "03EA03":
                    a="2";

                 break;
                case "03EA13":
                    a="3";

                 break;
                case "03EA4C":
                    a="4";

                 break;
                case "03EEC4":
                    a="5";

                 break;
                case "03EECC":
                    a="6";

                 break;
                case "03EBB6":
                    a="7";

                 break;
                case "03EA17":
                    a="8";

                 break;
                case "03E9FB":
                 a = "9";

                 break;

            }


            return a;

        }
        public void checkstations(byte[] netAddr)
        {
            string statinNetAdress = HexStr(netAddr);
            string stationNum = whichstation(statinNetAdress);
            int StationInt = Convert.ToInt32(stationNum);




            checkStations[StationInt- 1] = 1;
            mapsignal(StationInt, "idle.gif");


            if (MedicalStations[StationInt - 1] == 0 &
SecurityStations[StationInt - 1] == 0)
            {
                mapsignal(StationInt, "idle.gif");
            }
            else if (MedicalStations[StationInt - 1] > 0 &
SecurityStations[StationInt - 1] > 0)
            {
                mapsignal(StationInt, "alert3.gif");
            }
            else if (MedicalStations[StationInt - 1] > 0 &
SecurityStations[StationInt - 1] == 0)
            { mapsignal(StationInt, "alert.gif"); }
            else if (SecurityStations[StationInt - 1] > 0 &
MedicalStations[StationInt - 1] == 0)
            { mapsignal(StationInt, "alert2.gif"); }






        }
        public void tellVmStat(int status, object arg)
        {
            byte statusCode = Convert.ToByte(status & 0xFF);
            byte statusVal = Convert.ToByte((status >> 8) & 0xFF);
            if (statusCode == 5) //Status code for PING response
            {
                //statusVal will equal the link quality and arg will
equal the SNAPpy script name
                txtInfo.Text += "\r\nGot PING Response From: " +
HexStr(netAddr);
            }
        }

        public void dataMode(byte[] data)
        {
            txtInfo.Text = "Received from data mode: " +
System.Text.Encoding.Default.GetString(data);
        }



        private void frmMain_FormClosing(object sender,
FormClosingEventArgs e)
        {
            proxy.disconnect(this.getSnapNetAddr(), true);
        }









        public Image alert { get; set; }

        private void frmMain_Load(object sender, EventArgs e)
        {

        }










    }
}
```

Appendix C – SYNAPSE Data Sheets

C1. SYNAPSE RF Engine



SYNAPSE RF Engine_410:Datasheets 4/15/10 12:13 PM Page 1

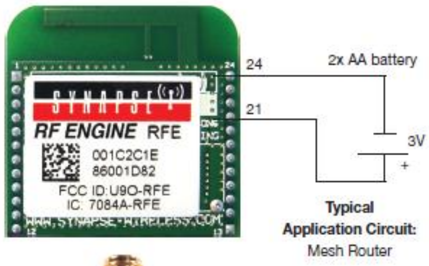
SYNAPSE Wireless Technology to Control and Monitor Anything from Anywhere™

SYNAPSE RF Engine

Synapse provides reliable IEEE802.15.4 RF Engines®. These small, low-powered, 2.4 GHz transmitter-receiver modules can have a range of up to three miles and power consumption as low as 2.5 µA. The RF Engines come loaded with the Synapse SNAP® network operating system. Typical applications include a wireless serial port, sensor monitoring, actuator control, or an intelligent embedded controller - *the possibilities are limitless*.

All RF Engines include:

- 19 GPIO, and 8 can be A/D inputs
- Two UART ports for control or transparent data
- Low power modes: 2.5 µA with internal timer running
- FCC certified on all 16 channels
- Spread spectrum (DSSS) technology surmounts noisy environments
- Socketable or solderable



Typical Application Circuit: Mesh Router

Choice of Synapse RF Engines:

RF100PD6

- Receive amplifier (10 dBm) standard
- Transmit amplifier (18 dBm) for best-in-class range (up to 3 miles LOS)
- SMA antenna allows best flexibility for mounting orientation
- Manufacturer's suggested retail price: **$39.00**

(volume discounts available)

RF100PC6

- Receive amplifier (10 dBm) standard
- Transmit amplifier (18 dBm) for best-in-class range (up to 2.5 miles LOS)
- Onboard F-antenna gives compact physical size
- Manufacturer's suggested retail price: **$35.00**

(volume discounts available)

RF100P86

- Transmit amplifier (3 dBm) for 3000ft LOS range
- Onboard F-antenna gives compact physical size
- Lowest TX current draw for sleep mode (40mA)
- More consistent RF power level over battery life
- Manufacturer's suggested retail price: **$24.00**

(volume discounts available)

(AES-128 encrypted version is available for applications requiring extra security)

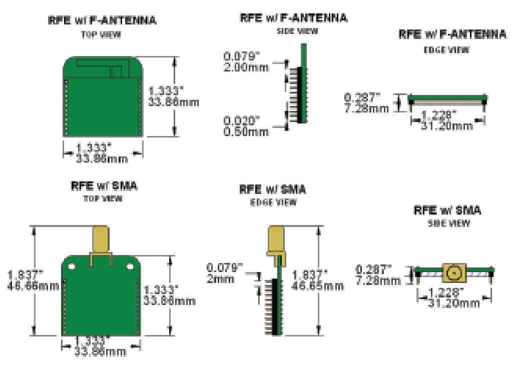
Synapse RF Engine modules have achieved ZigBee® Compliant Platform (ZCP) certification.

430-0118.01B

Wireless Technology to Control and Monitor Anything from Anywhere™

SYNAPSE RF Engine

Physical Dimensions



Specifications

Performance	Indoor Range	Up to 1000 ft. (** 200 ft.)
	Outdoor LOS Range	Up to 3.0 miles (** up to 3000 ft.)
	Transmit Power Output	18 dBm (** 0 dBm.)
	RF Data Rate	250,000 bps
	Receiver Sensitivity	-102 dBm (1% PER)
Power Requirements	Supply Voltage	2.7 – 3.4V
	Transmit Current (Typ)	110 mA (** 40 mA)
	Receive Current (Typ)	65 mA
	Idle Current (Typ)	15 mA
	Sleep Current (Typ)	2.5 µA
General	Frequency	ISM 2.4 GHz
	Spreading Method	Direct Sequence
	Modulation	O-QPSK
	Dimensions	1.333" x 1.333"
	Operating Temperature	-40 to 85 deg C.
	Antenna Options	Integrated F, External RPSMA
Networking	Topology	SNAP or ZigBee
	Number of Channels	16
Available I/O	UARTS with HW Flow Control	2 ports – 8 total I/O
	GPIO	19 total, 8 can be analog in with 10-bit ADC
Agency Approvals	FCC Part 15.247	Yes, Class B
	Industry Canada (IC)	Yes

*** RFE with receive only amp specs, all other specs apply to all RF Engines*

For more technical details, see SNAP Hardware Technical Manual on the SYNAPSE Customer Forum: forums.synapse-wireless.com

Part Selection

Part No.	Antenna	Receive Amp	Power Amp
RF100PD6	External *	Yes	Yes
RF100PC6	F type	Yes	Yes
RF100P86	F type	Yes	No

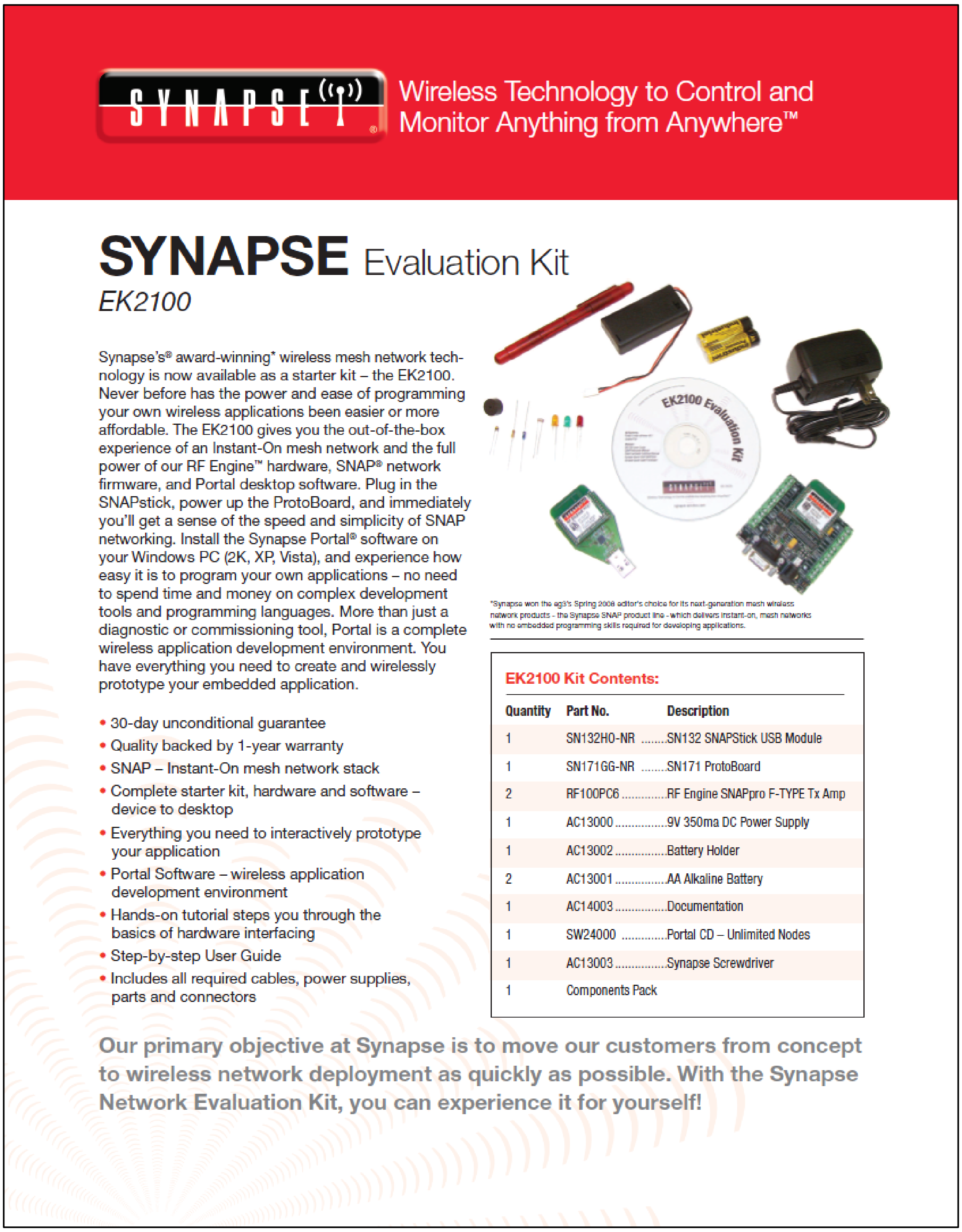
** External antenna sold separately - ask your sales representative*

Pinout

Pin No.	Name	Direction	Description
1	GND	-	Power Supply/Return
2	GPIO0_TPM1CH2	Bidirectional	GPIO, or Timer1 Channel 2 (PWM)
3	GPIO1_KBI0	Bidirectional	GPIO, Keyboard In
4	GPIO2_KBI1	Bidirectional	GPIO, Keyboard In
5	GPIO3_RX_UART0	Input	UART0 Data In
6	GPIO4_TX_UART0	Output	UART0 Data Out
7	GPIO5_KBI4_CTS0	Bidirectional	GPIO, Keyboard In, or UART0 CTS
8	GPIO6_KBI5_RTS0	Bidirectional	GPIO, Keyboard In, or UART0 RTS
9	GPIO7_RX_UART1	Input	UART1 Data In
10	GPIO8_TX_UART1	Output	UART1 Data Out
11	GPIO9_KBI6_CTS1	Bidirectional	GPIO, Keyboard In, or UART1_CTS
12	GPIO10_KBI7_RTS1	Bidirectional	GPIO, Keyboard In, or UART1_RTS
13	GPIO11_AD7	Bidirectional	GPIO, or Analog In
14	GPIO12_AD6	Bidirectional	GPIO, or Analog In
15	GPIO13_AD5	Bidirectional	GPIO, or Analog In
16	GPIO14_AD4	Bidirectional	GPIO, or Analog In
17	GPIO15_AD3	Bidirectional	GPIO, or Analog In
18	GPIO16_AD2	Bidirectional	GPIO, or Analog In
19	GPIO17_AD1	Bidirectional	GPIO, or Analog In
20	GPIO18_AD0	Bidirectional	GPIO, or Analog In
21	VCC	-	Power Supply
22	Reserved	-	-
23	RESET_L	Bidirectional	Module Reset, Active Low
24	GND	-	Power Supply/Return

SYNAPSE Wireless, Inc. 500 Discovery Drive, Huntsville, Alabama 35806
877 982-7888 · synapse-wireless.com

C2. **SYNAPSE Evaluation Kit**



Wireless Technology to Control and Monitor Anything from Anywhere™

SYNAPSE Evaluation Kit

EK2100

Synapse's® award-winning* wireless mesh network technology is now available as a starter kit – the EK2100. Never before has the power and ease of programming your own wireless applications been easier or more affordable. The EK2100 gives you the out-of-the-box experience of an Instant-On mesh network and the full power of our RF Engine™ hardware, SNAP® network firmware, and Portal desktop software. Plug in the SNAPstick, power up the ProtoBoard, and immediately you'll get a sense of the speed and simplicity of SNAP networking. Install the Synapse Portal® software on your Windows PC (2K, XP, Vista), and experience how easy it is to program your own applications – no need to spend time and money on complex development tools and programming languages. More than just a diagnostic or commissioning tool, Portal is a complete wireless application development environment. You have everything you need to create and wirelessly prototype your embedded application.

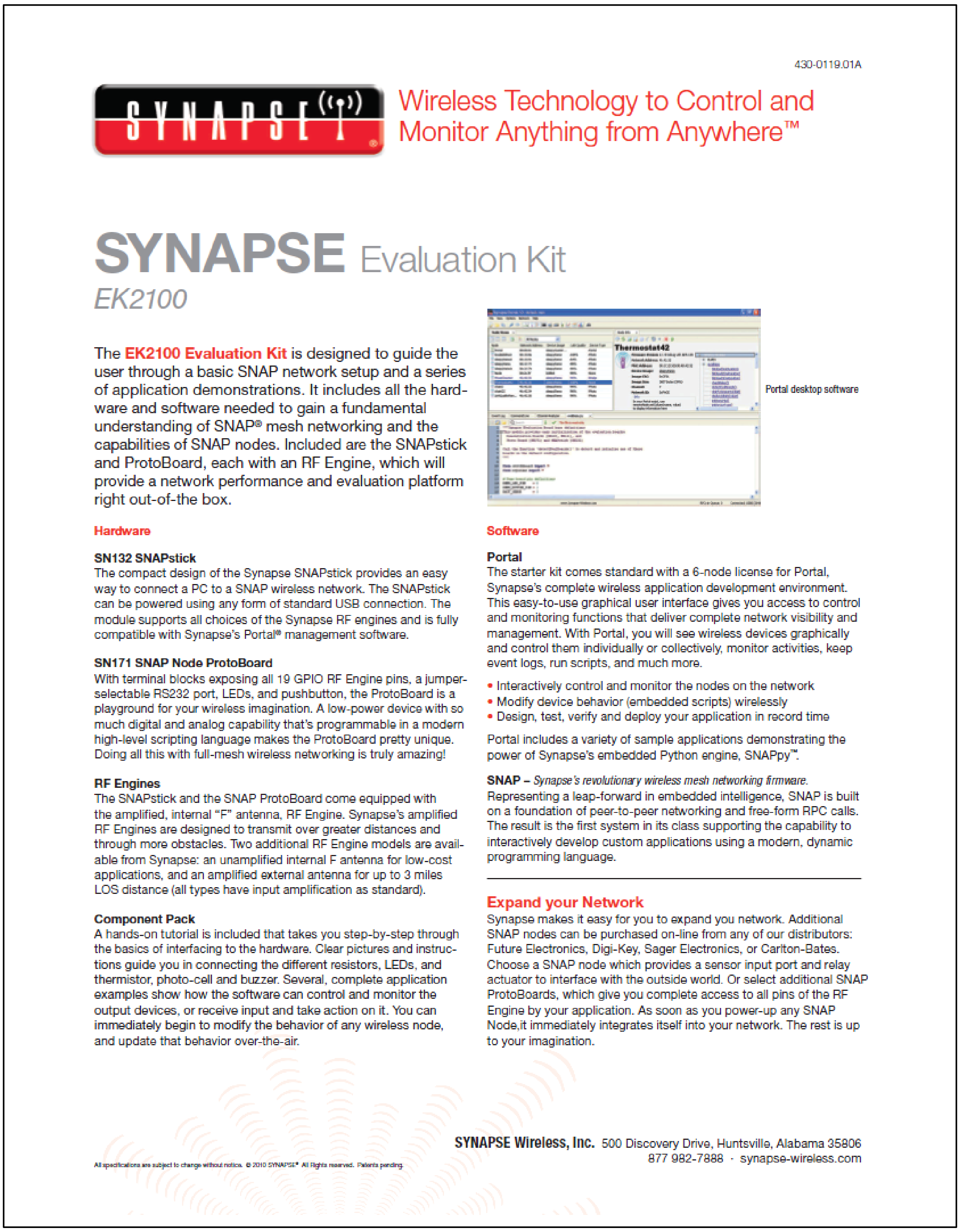
- 30-day unconditional guarantee
- Quality backed by 1-year warranty
- SNAP – Instant-On mesh network stack
- Complete starter kit, hardware and software – device to desktop
- Everything you need to interactively prototype your application
- Portal Software – wireless application development environment
- Hands-on tutorial steps you through the basics of hardware interfacing
- Step-by-step User Guide
- Includes all required cables, power supplies, parts and connectors

*Synapse won the eg3's Spring 2008 editor's choice for its next-generation mesh wireless network products - the Synapse SNAP product line - which delivers instant-on, mesh networks with no embedded programming skills required for developing applications.

EK2100 Kit Contents:

Quantity	Part No.	Description
1	SN132H0-NR	SN132 SNAPStick USB Module
1	SN171GG-NR	SN171 ProtoBoard
2	RF100PC6	RF Engine SNAPpro F-TYPE Tx Amp
1	AC13000	9V 350ma DC Power Supply
1	AC13002	Battery Holder
2	AC13001	AA Alkaline Battery
1	AC14003	Documentation
1	SW24000	Portal CD – Unlimited Nodes
1	AC13003	Synapse Screwdriver
1	Components Pack	

Our primary objective at Synapse is to move our customers from concept to wireless network deployment as quickly as possible. With the Synapse Network Evaluation Kit, you can experience it for yourself!



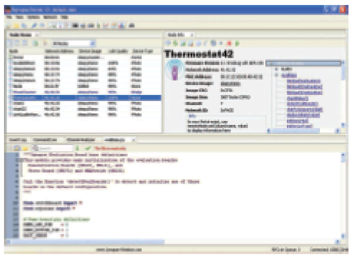
430-0119.01A

Wireless Technology to Control and Monitor Anything from Anywhere™

SYNAPSE Evaluation Kit

EK2100

The **EK2100 Evaluation Kit** is designed to guide the user through a basic SNAP network setup and a series of application demonstrations. It includes all the hardware and software needed to gain a fundamental understanding of SNAP® mesh networking and the capabilities of SNAP nodes. Included are the SNAPstick and ProtoBoard, each with an RF Engine, which will provide a network performance and evaluation platform right out-of-the box.



Portal desktop software

Hardware

SN132 SNAPstick
The compact design of the Synapse SNAPstick provides an easy way to connect a PC to a SNAP wireless network. The SNAPstick can be powered using any form of standard USB connection. The module supports all choices of the Synapse RF engines and is fully compatible with Synapse's Portal® management software.

SN171 SNAP Node ProtoBoard
With terminal blocks exposing all 19 GPIO RF Engine pins, a jumper-selectable RS232 port, LEDs, and pushbutton, the ProtoBoard is a playground for your wireless imagination. A low-power device with so much digital and analog capability that's programmable in a modern high-level scripting language makes the ProtoBoard pretty unique. Doing all this with full-mesh wireless networking is truly amazing!

RF Engines
The SNAPstick and the SNAP ProtoBoard come equipped with the amplified, internal "F" antenna, RF Engine. Synapse's amplified RF Engines are designed to transmit over greater distances and through more obstacles. Two additional RF Engine models are available from Synapse: an unamplified internal F antenna for low-cost applications, and an amplified external antenna for up to 3 miles LOS distance (all types have input amplification as standard).

Component Pack
A hands-on tutorial is included that takes you step-by-step through the basics of interfacing to the hardware. Clear pictures and instructions guide you in connecting the different resistors, LEDs, and thermistor, photo-cell and buzzer. Several, complete application examples show how the software can control and monitor the output devices, or receive input and take action on it. You can immediately begin to modify the behavior of any wireless node, and update that behavior over-the-air.

Software

Portal
The starter kit comes standard with a 6-node license for Portal, Synapse's complete wireless application development environment. This easy-to-use graphical user interface gives you access to control and monitoring functions that deliver complete network visibility and management. With Portal, you will see wireless devices graphically and control them individually or collectively, monitor activities, keep event logs, run scripts, and much more.

- Interactively control and monitor the nodes on the network
- Modify device behavior (embedded scripts) wirelessly
- Design, test, verify and deploy your application in record time

Portal includes a variety of sample applications demonstrating the power of Synapse's embedded Python engine, SNAPpy™.

SNAP – *Synapse's revolutionary wireless mesh networking firmware.*
Representing a leap-forward in embedded intelligence, SNAP is built on a foundation of peer-to-peer networking and free-form RPC calls. The result is the first system in its class supporting the capability to interactively develop custom applications using a modern, dynamic programming language.

Expand your Network

Synapse makes it easy for you to expand you network. Additional SNAP nodes can be purchased on-line from any of our distributors: Future Electronics, Digi-Key, Sager Electronics, or Carlton-Bates. Choose a SNAP node which provides a sensor input port and relay actuator to interface with the outside world. Or select additional SNAP ProtoBoards, which give you complete access to all pins of the RF Engine by your application. As soon as you power-up any SNAP Node,it immediately integrates itself into your network. The rest is up to your imagination.

SYNAPSE Wireless, Inc. 500 Discovery Drive, Huntsville, Alabama 35806
877 982-7888 · synapse-wireless.com