2012

Application framework for wireless sensor networks [thesis]

Amro Qandour
Edith Cowan University

Edith Cowan University

Copyright Warning

Application Framework For Wireless Sensor Networks

by

Amro Qandour

This thesis is presented in fulfillment of the requirements for the degree of

Master of Engineering Science

SCHOOL OF ENGINEERING

FACULTY OF COMPUTING, HEALTH AND SCIENCE

EDITH COWAN UNIVERSITY

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

ABSTRACT

Wireless Sensor Networks (WSNs) are based on innovative technologies that had revolutionized the methods in which we interact with the environment; i.e., through sensing the physical (e.g., fire, motion, contact) and chemical (e.g., molecular concentration) properties of the natural surroundings. The hardware in which utilized by WSNs is rapidly evolving into sophisticated platforms that seamlessly integrate with different vendors and protocols (plug-n-play). In this thesis, we propose a WSN framework which provides assistance with monitoring environmental conditions; we focus on three main applications which include: **a.** Air-quality monitoring, **b.** Gas-leak detection, and **c.** Fire sensing.

The framework involves four specifications: **1.** Over the air programming (OTAP), **2.** Network interconnections, **3.** Sensors manageability, and **4.** Alarm signaling. Their aim is to enhance the internetwork relations between the WSNs and the outside-world (i.e., main users, clients, or audience); by creating a medium in which devices efficiently communicate, independent of location or infrastructure (e.g., Internet), in order to exchange data among networked-objects and their users. Therefore, we propose a WSN-over-IP architecture which provides several renowned services of the Internet; the major functionalities include: live-data streaming (real-time), e-mailing, cloud storage (external servers), and network technologies (e.g., LAN or WLAN).

WSNs themselves operate independently of the Internet; i.e., their operation involve unique protocols and specific hardware requirements which are incompatible with common network platforms (e.g., within home network infrastructure). Hybrid technologies are those which support multiple data-communication protocols within a single device; their main capabilities involve seamless integration and interoperability of different hardware vendors. We propose an overall architecture based on hybrid communication technology in which data is transmitted using three types of protocols: 802.11 (Wi-Fi), 802.15.4 and Digimesh (WSN).

DECLARATION

I certify that this thesis does not, to the best of my knowledge and belief:

(i) incorporate without acknowledgement any material previously submitted for a degree or diploma in any institution of higher education;

(ii) contain any material previously published or written by another person except where due reference is made in the text; or

(iii) contain any defamatory material.

I also grant permission for the Library at Edith Cowan University to make duplicate copies of my thesis as required.

ACKNOWLEDGMENT

This thesis was made possible due to the masterly guidance of Prof. Daryoush Habibi. His wide knowledge has taught many invaluable lessons which helped me complete this work. I also thank Dr Iftekhar Ahmad for his continuous support and for being a massive source of information throughout my journey.

To my family, Mother Dana, Father Rushdi, and my siblings Raed, Hala, and especially to my youngest sister Yasmeen. I extend my gratitude for all of you and say you will always be in my heart. Finally, to my partner Emilie, I wish to say thank you for being there until the end, and for giving me all the support I need to finish this thesis.

To my friends, including the CCER team thank you all. Last but not least, to my good friends Steven and Darragh, I thank you both from the bottom of my heart.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

INTRODUCTION

WSN is a group of interconnected node devices that sense and exchange the data gathered from the natural environment; to help reveal and better understand the perceptual properties of the world around us (natural or biochemical parameters). The natural principles are governed by the laws of physics and mathematics; however, WSNs allow us to observe and monitor these parameters upfront using electronic hardware and physical sensors. This thesis focuses on the application side of WSNs; and to develop the frameworks that expose their true potential towards environmental monitoring.

WSNs have many practical applications within the health-care system, such as measuring biometric parameters of patients (e.g., heartbeats, breathing rates, and muscle activity) [1]; this type of WSN is referred to as Wireless Body Area Network (WBAN) [2]. In hospital and other health-care institutions, WBANs are used on patients to detect early signs of illnesses, and anticipate heart attacks or tumors [3]. WSNs are also used by the agricultural industry to monitor the growth-rate in fruits and vegetables using precision irrigation techniques [4]; also with investigating climate impact (greenhouse gas) on their growth [5].

WSNs are used to study non-artificial phenomenas that occur in nature; which involve geological, meteorological, and oceanographic phenomenas [6]. Earthquakes and volcanoes are geological phenomenas; destructive in nature and undeterred by any means of intervention. However, early signs can be predicted using a combination of highly sensitive sensors and WSNs [7]. Furthermore, WSNs are used to monitor safety parameters (stress, fatigue, wear) in building-infrastructure (e.g., bridges, towers, buildings) in the aftermath of earthquakes or other nature disasters (tsunamis, floods, etc.) [8].

In the sections to follow, we outline our research motivations, aims, and major contributions towards the WSN field of study.

1.1 Motivation and Research Question

The evolution of wireless and sensor technologies serve as promising signs surrounding the future of WSNs. Environmentally friendly, low-power utilization, and high-efficiency are some reasons behind the extensive research behind this technology. However, the practical configuration of WSNs is a field which has not received enough attention by researchers, many of whom have chosen theoretical investigations over actual applications. Our motivation is to take an in-depth, highly pragmatic approach, which demonstrates how to lay the foundations for developing WSN applications using off-the-shelve devices. We adopt a systematic approach for addressing communication, networking, and management of WSNs. First however, a number of challenges must be overcome, as outlined below:

1. **Communications:** Large data transfer is simply not feasible in WSNs, which unlike other wireless hardware (laptop NICs, home-routers, etc.), WSN communication protocols are designed to meet low-power specifications. This is partially due to the fact that sensor nodes do not have access to a constant source of electricity (to recharge batteries), therefore the transmission bandwidth must be one that is efficient and yet supports relaxed data-rates.

2. **Networking:** WSNs which consist of large number of nodes often lead to unpredictable results and frequent topological changes. These dynamic alterations cause an unbalance of energy consumption, channel contention, and higher collision rates among sensor devices. Multi-hopping techniques have been known to reduce the number of simultaneous one-to-one connections between sensor nodes and base-stations, however, these techniques alone are not enough to guarantee data integrity and protection against loss of information.

3. **Management:** WSNs require adept management techniques in order to expose their true potential in terms of usability, suitability and functionality in real applications. Firstly, power management is very important given that sensor nodes have limited access to energy resources such as batteries. Secondly, node failure management must be addressed appropriately given that WSNs are prone to hardware failures caused by noise interference, node device failures, and buggy software. Finally, the management of components and subsystems such as communication modules, sensors, and periphery units, must be addressed carefully given that improper use can lead to irregular node behavior and an increase of data loss.

To date, research has not been able to address these challenges collectively, but we believe that developing a general architecture which is not based solely on WSN protocols, instead it incorporates a mixture of various technologies can help overcome these challenges. Our core research is developing an architectural model for off-the-shelve devices based on practical virtues and real hardware implementation. Our aim is enhancing the services and functionality of hardware used in environmental and safety monitoring applications.

1.2 Research Aims

Our research opens up three important questions: a. What types of communications are to be used, and how will they be implemented? b. Which methods are used to connect the hardware devices, and how are they applied? And c. Which methods are used to manage the WSN?

1.2.1 Communications

WSNs adhere with the 802.15.4 specifications for the Physical (PHY) and Medium Access Control layers. The upper layers in the communication stack which include the Network (NWK) and Application (APP) layers remain either proprietary or specified by exclusive alliances such as DigiMesh or Zigbee. These protocols however, are not open source and also not supported by all sensor devices, so this causes interoperability issues between different hardware vendors. An alternate solution is using IP to connect WSNs together. The Internet layer consists of a number of protocols which are responsible for the transport of packets from the originating host to a specified destination solely based on the address. Ofcourse, TCP/IP is the leading protocol in global communications; most online activities involve TCP/IP protocols including web browsing (Hypertext Transfer Protocol-HTTP), email transmission (SMTP-Simple Mail Transmission Protocol), transferring files (FTP-File Transfer Protocol), and remote access (Telnet). Our work aims towards integrating TCP/IP with WSNs to enable access to online services on the Internet, and that includes e-mailing, long distance remote capabilities, live streaming of data, and Over The Air Programming (OTAP) on sensor nodes.

1.2.2 Networking

Data transmission is a dual process which involves a transmitter for outgoing data and a receiver for incoming data. In wireless networks, radio frequency (RF) is the communication medium used to connect network devices together in order to exchange information. Similarly, wired networks use technologies such as Ethernet over twisted pair which provides the means of communicating between networked systems. WSNs consist of a large number of sensor nodes, connected together by low power radio modules which unlike the traditional wireless devices used in our PCs, they spend majority of the time in a state of sleep, or inactivity. Our work aims towards implementing a Wireless Mesh Sensor Network (WMSN). This type of network sustains the uniformity of long distance communications by breaking connection links into a series of smaller hops, thus intermediate nodes cooperatively make forwarding decisions based on the structure of the network. We also aim to integrate the WMSN with an IP network in order to improve application capabilities in terms of access to live sensor data, web server support, Secure Shell (SSH) connections, and cloud based storage.

1.2.3 Management

Our work focuses on two management techniques: a) Node management, and b) Network management.

1.2.3.1 Node management

We aim to implement management strategies which improve performance, reliability and stability of sensor nodes. These strategies include:

1. Low-Power alarm utilization: This first technique in general is used to monitor the battery levels in the sensor nodes, but we aim to implement the notion of dynamic threshold reconfigurability, which means that tasks will only be executed while battery levels are above a certain threshold. We use this technique to increase task independency from one another, in which prevents nodes from getting stuck in endless loops when tasks fail to execute due to insufficient available energy.

2. Sleep modes: These modes include normal, deep, and hibernate. The difference between them is the amount of power that is consumed while being in active state. Deep sleep and hibernation modes have higher power efficiency than the regular sleep because in those two states node components are completely detached from the power source. We aim to use these modes to improve battery efficiency and to extend the life duration of each node.

3. Integration of algorithms: We aim to implement algorithms to support the execution with different tasks and functions in the application. We plan to develop algorithms that will support the generation of alarm signals in emergency events, enable radio communication during OTAP operations, and also to control the sensing architecture demanded by the application.

4. Event priority management: We aim to implement a task management system based on priority and importance levels. This technique will enable the nodes to react more efficiently to the events which happen in their surroundings. For example, if a node detects fire the same time it needs to access the SD module; then it needs to distinguish which event is more important a) sending an alarm, or b) save data. We need to instruct the node that when fire is detected, it needs to prioritize sending the alarm message prior to saving data into the SD card. Furthermore, we aim to apply event management techniques upon all aspects related to node functions and tasks administration.

1.2.3.2 Network management

We aim to provide a modern approach on WSN management by using various techniques for: a) accessing/storing sensor data, b) using secure transmission between networked devices, and c) synchronization of tasks in real time. The methods we use for implementing these techniques are:

1. Cloud storage: We aim to implement data storage in a cloud based network in which consists of various host workstations acting together centrally to store information from numerous WSNs.

2. Encrypted data transmission: Unsecured transmission of data causes unpredictable and unreliable operations in a WSN. Attacks on the WSN can be in many forms such as: a) manipulation of data in order to trick the network into doing something else (e.g-false alarms), b) overwhelming the network by flooding it with packets in which leads to network failure, or c) stealing data and using it for personal gains. These issues can be minimized by introducing security measures which enable networks to distinguish between real and foreign nodes. We aim to implement application and link level features in order to improve the security in WSNs. Application level entails to the communication between nodes inside the application not the network, whereas link security refers to the encryption of data-payload prior to transmission, and the only way a networked device can make sense of the data is by previous knowledge of the encryption key.

3. Real-time system: In a real-system, every networked device including all node components, sensors, and radio transmission are synchronised in real-time. This implies that WSNs respond to each command exclusively without interfering with normal network operations. These commands may include: a) retrieving sensor data, b) changing operating channels, c) uploading data to web-server, or d) performing device maintenance. We aim to implement a system which is capable of many real-time functions including alarm generation, over the air programming, database and webserver support, e-mail messaging, and online publishing (twitter, word-press, etc.).

1.3 Research Contributions

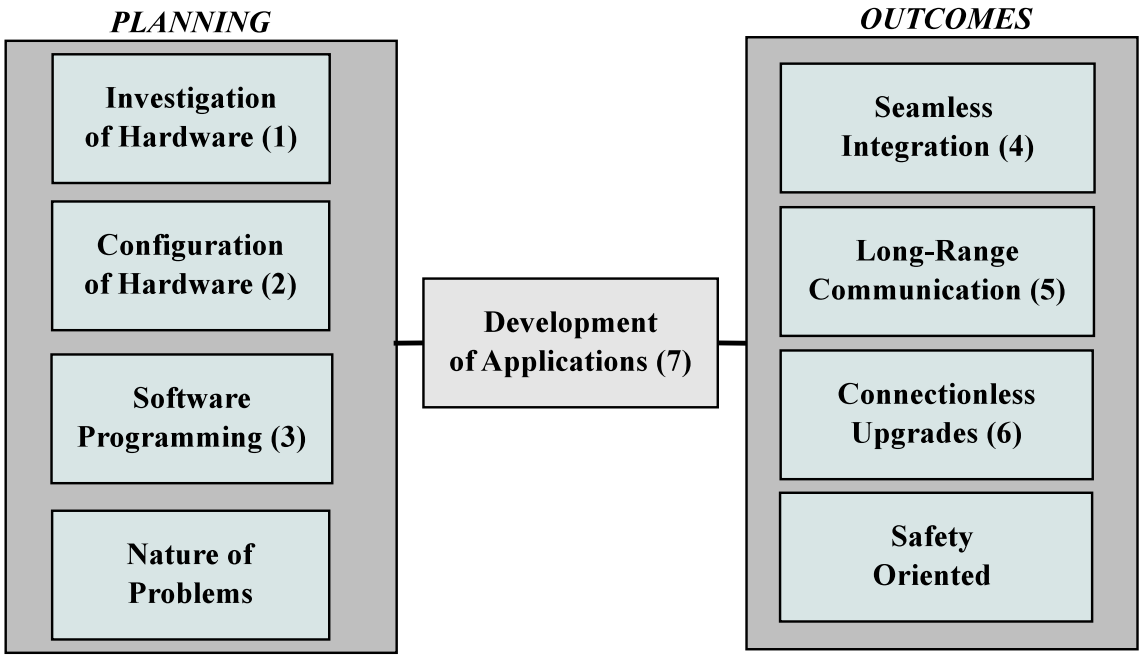


Figure 1.1: Conceptual model for research contributions.

In this research, we investigated frameworks using off-the-shelve WSN hardware to study the physical nature of the environment and monitor the safety of the surroundings. Figure 1.1 presents our contributions as a model for conceptualizing frameworks based on WSNs which are used for studying the physical nature and monitoring the safety in surrounding environments. This model has 6 critical components:

1. **Investigation of hardware**: We investigated available off-the-shelve equipment for environmental and safety applications. We then then picked the most suitable ones and solved the research challenges towards developing practical systems using those devices.

2. **Configuration of hardware**: We configured the hardware according to specific applications and practical virtues. This entails choosing the correct sensors, radio modules, and node accessories for each individual device.

3. **Software programming**: We coded every piece of software used by our nodes using open-source API (Application Programming Interface) and IDE (Integrated Development Environment). The software that we developed are used in algorithms, alarm generation, events management, and general sensing requirements.

4. **Seamless integration**: We have configured the WSNs to integrate seamlessly with IP networks in our implementation of the Mesh-Hood (M-Hood) system. The M-Hood system consists of numerous sensor nodes which are deployed to create a primary mesh network, and access points (APs) which are bridged together using wireless connections to provide real-time monitoring and Internet services.

5. **Long range communications**: We have developed methods which address remote connectivity in WSNs. These methods are based on WSN-over-IP which can be applied to improve user control of WSNs, and access to data from remote locations.

6. **Connectionless upgrades**: We have integrated over the air programming capabilities for local (on-site) and external (remote) WSNs. For external locations, we use IP to connect to the main base-station (WSN gateway) and perform connectionless upgrades (no wires) on the nodes. Furthermore, we have prepared the nodes to accept the local over the air requests without interfering with the normal operations of the WSN unless instructed by an administrator.

7. **Development of applications**: From the previous findings, we developed environmental applications based on systematic architectures which deal exclusively with communication, networking, power management, sensor integration, remote connectivity, and alarm generation. We provide a systematic approach which explains the fundamental prerequisites and real-configurations for each architecture we use in our applications.

1.4 Thesis Outline

The structure of this thesis is as follows:

- In **Chapter 2**, we familiarize the audience with WSN background which includes relevant information on the communication protocols, routing techniques, power consumption requirements, and real-time operations. We then discuss the relevant literature and works which is relevant to this research.

- In **Chapter 3**, we develop the major system architecture of the WSN. More specifically, we define the fundamental system components used in over the air programming, networking and communication specifications, sensor integrations, alarm generation, and remote connectivity.

- In **Chapter 4,** we develop the real systems used in environmental applications using the knowledge and understanding of the previous chapter. Then we propose the techniques we use for sensing the environment, generating alarms based on different emergencies, and finally the functions for handling the risks associated with a particular event.

Chapter 2

BACKGROUND AND LITERATURE REVIEW

In this chapter, we discuss the background of WSNs which includes information on the communication protocols, routing techniques, power requirements in low-power applications, and the use of real time operating systems in sensor networks. The next part of this chapter holds more complicated discussions on various architectures, applications, and research challenges in WSNs.

2.1 Wireless Sensor Network Fundamentals

In this section, we discuss the foundation of many important concepts which helped WSNs to be recognized as one of the most promising technologies of today and for future generations to come.

2.1.1 Communication Protocols for Wireless Sensor Networks

Traditional wireless sensor networks comply with ad-hoc specifications in addressing various network functionalities such as routing and data transmission. The evolution in network specific protocols have improved the functionalities which are needed for handling complex and demanding networks. In the WSN domain, development of the IEEE-802.15.4 standard protocol has revolutionized this technology. The main focus of this protocol is to provide low power consumption and reliable throughput for short range communications. In addition, other proprietary protocols such as Zigbee, and Digimesh have been developed specifically for WSN technology to enhance usability and improve their services. This section will outline the main communication protocols in reference with the OSI (Open System Interconnection) model specifically for the IEEE-802.15.4 and the Zigbee protocols.

2.1.1.1 IEEE-802.15.4 protocol

WSNs commonly adhere with the 802.15.4 specifications for the physical and MAC layers. The upper layers in the communication stack are not addressed in the 802.15.4 protocol. Therefore sensor networks are not subjected to specific protocols in these upper layers, they remain either proprietary owned or specified by exclusive alliances such as Z-Wave and Zigbee.

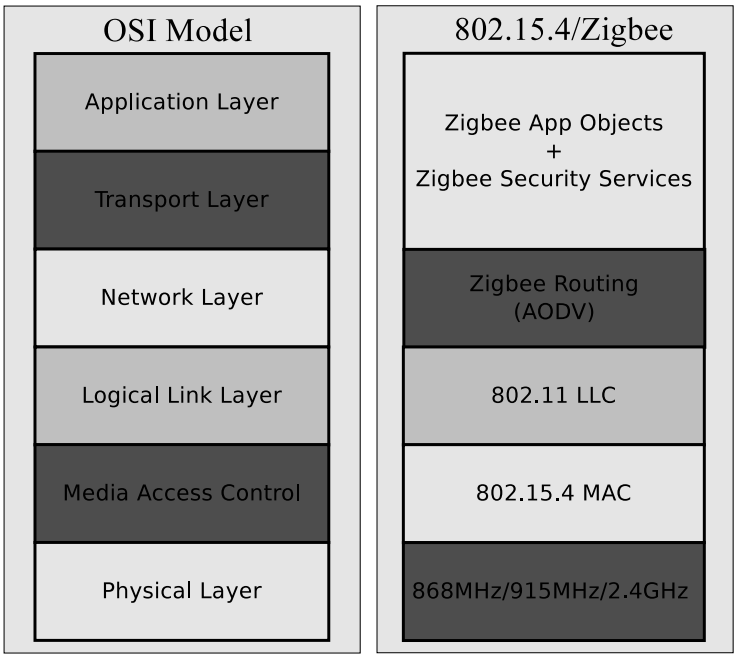


Figure 2.1: The Open System Interconnection (OSI) model.

The IEEE 802.15.4 defines a communication specifications for the Physical and MAC layer of the OSI model as illustrated by Figure 2.1. The focus of this protocol is to provide short range wireless networking with relaxed data rates and low energy consumptions. Furthermore, the frequencies which are defined in this standard are divided into 3 main bands:

- 868-868.6 MHz
- 900-928 MHz
- 2.4-2.48 GHz

The IEEE 802.15.4 standard [9] supports different kinds of modulation schemes as highlighted by Table 2.1, which includes BPSK (Binary Phase Shift Keying), O-QPSK (Offset-Quadrature Phase Shift Keying), ASK (Amplitude Shift Keying), and DSSS (Direct Sequence Spread Spectrum). Also these modulations allow communication bandwidths in the range of 20 Kb/s to 250 Kb/s.

Many sensor nodes use the Direct Sequence Spread Spectrum (DSSS) to modulate the information before being sent to the physical layer. This modulation scheme is very popular because it causes less interferences in the bands used, and improves the Signal-to-Noise ratio. Furthermore, this modulation scheme works by splitting the transmitted signal to 4 different signals where it occupies a larger bandwidth but it uses a lower power density for each signal.

Frequency (MHz)	Modulation	Bit rate (kbps)
868	BPSK	20
915	BPSK	40
868	ASK	250
915	ASK	250
868	O-QPSK	100
915	O-QPSK	250
2400	O-QPSK	250

Table 2.1: The IEEE-802.15.4 modulation schemes.

The IEEE-802.15.4 protocol specifies the Carrier Sense Multiple Access-Collision Avoidance (CSMA-CA) scheme to avoid all nodes from transmitting at the same time. In this method each node will listen to the medium before transmitting and avoids sending information if the energy is found to be higher than a specific level. In addition, GTS (Guarantee Time Slots) is used to give each node a specific time slot to send its information, and also relies on a coordinator to synchronize the time slots. Energy detection functions are included in the 802.15.4-PHY parameters (PLME-ED) which are used to detect the amount of energy (activity, noise, interference) over the network.

The MAC layer in the 802.15.4 protocol defines two basic modes of operation: beacon mode and non beacon mode. The beacon mode is time dependent and used for time synchronization between the nodes. In addition, when beacon mode is set, it sends a superframe (Figure 2.2) which defines the time periods for active transmission and inactive transmission. Optimizing the beacon mode lowers the duty cycles of nodes and improve energy efficiency. Non-beacon mode is asynchronous in nature and time independent. When this mode is set, the nodes will periodically poll the coordinator to see if there is any data requests.

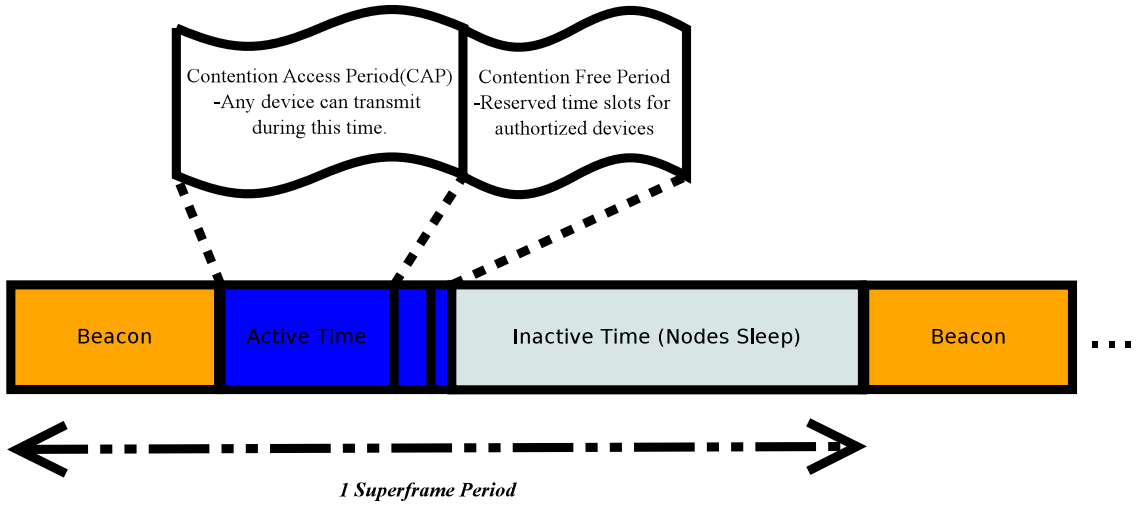


Figure 2.2: Beacon mode superframe.

The 802.15.4 is thought to be a protocol to get point to point and energy efficient communications. As mentioned earlier, it only specifies the parameters for the physical and MAC layers only. Meanwhile upper layers are defined by the Zigbee protocol.

2.1.1.2 Zigbee protocol

The Zigbee protocol defines specifications for the Network and Application layers of the OSI model illustrated by Figure 2.3. A Zigbee stack consists of many sub-modules, the application layer is divided into 3 sub-layers: the Application Sublayer (APS), the Application Framework (AF), and the endpoints (nodes). In essence, the APS module performs data transport services to all the end points, also it interfaces the network layer with the upper application layers of the Zigbee stack. Furthermore, the Application Framework (AF) layer is used to forward the packets sent by the APS to a specific node from its database.

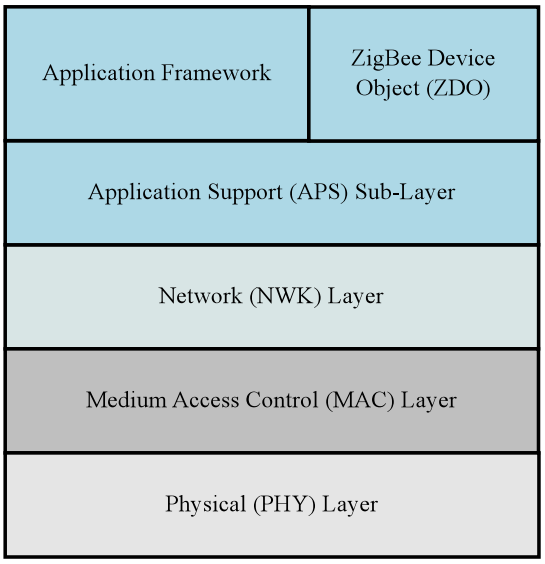


Figure 2.3: Zigbee stack.

The network layer performs a number of functions including:

- Routing of outgoing/ingoing packets.
- Device discovery.
- Association and authentication.
- Data encryption.
- Network maintenance.

Zigbee deals with the transmission process as follows, the APS sends data to the network layer via a data request. The data request frame contains the destination address, whether or not routing discovery is allowed, and also specifies the maximum number of hops the frame is allowed to travel. Zigbee frames contain two types of addresses: the 802.15.4 source/destination MAC address, and the Zigbee network source/destination address. The MAC address is used to determine the real next hop destination. The network layer then carries out a series of tests to determine whether the frame is a broadcast or to a device. If it is not a broadcast frame, then it checks the neighbour table for matching destination addresses. If a destination address is found, it would transmit the

data in a single hop. However, if the destination address is not found in the neighbour table, routing algorithms are used to determine the exact path for the destination. Routing tables are checked for matching next hop and destination addresses and if that's unsuccessful route discovery is performed. Once the route has been determined, the packets will be forwarded through multiple nodes to reach its final destination.

From a receiver perspective, the radio retrieves the frame and stores it in a buffer. The MAC layer decodes the payload header and passes it onto the network layer. Once the frame is in the network layer it would be processed according to the type of frame. Frames arriving in the network layer are classified as two types: data frames, and command frames. Each of one these unique frame types are processed differently. Data frames are treated either as destination frames, where the receiving node is the final destination address in which case it is passed onto the application layer. Broadcast and multi-hop frames are the remaining types. The second type of frames in the network layer are the command frames which are used to handle network maintenance, link maintenance, and association or dissociation in the network.

Zigbee also offers network management services such network formation, network discovery, and network join. A Zigbee network consists of 3 devices: coordinator, router, and end-device. The coordinator is a router that starts the network and performs scanning functions, selecting communication channels, and setting network ID. The router, or parent node is a device capable of routing functions. The end device, or child node is a device that has limited resources and not capable of performing any routing decisions. Child nodes are used to send information about the environment to the parents nodes where processing and storing of the information can take place. From a management view, network formation can only be performed by a coordinator device. This function will send a request to the MAC layer to perform an energy scan on the available channel. When the scan is completed, the network formation function will decide on the most suitable channel for the new network and a new network ID will be set.

Network discovery services are used to discover existing networks on the same channel. It is used mainly when devices are looking new networks to join. The function works by calling the MAC layer's active scan function which broadcast a beacon request. Devices on the network will then respond to this request by transmitting an 802.15.4 beacon superframe. The superframe contains information about the MAC address and the Zigbee network parameters such network ID, amount of routers and end devices allowed to join. The device which made the network discovery request then decides whether to join the network or look for a new one on a different channel.

2.1.1.3 DigiMesh (DM) protocol

The Digi-Mesh (DM) protocol is a proprietary owned stack which provides management and networking functions in the WSNs. The main feature in this protocol is it allows mesh network configuration and topologies. The mesh network topology is extremely useful in WSN architecture as it enables longer communication distances to exist between the nodes and the base-station through a routing technique known as *Ad-hoc On-demand Distance Vector* (AODV). Furthermore,

DigiMesh incorporates additional features including [10]:

1. Self-healing: node association or dissociation does not affect the network performance, and therefore does not cause a failure.

2. Peer-to-peer architecture: nodes are not subjected to obligatory relationships as in parent-child constraints, and there is no hierarchical based network architecture which is present in the Zigbee protocol.

3. Routing capabilities: AODV routing is used in the DM protocol to improve the distances between nodes and base-stations.

4. Route discovery: this technique uses broadcast messages to determine the route for transmitting data from the source node to the final destination. Therefore there is no intermediate requirement to maintain a routing map.

5. Selective acknowledgments: only destination nodes are able to reply to route requests.

6. Reliable data delivery: the DM protocol helps overturn communication problems that exist in difficult terrains such as dense vegetation, building walls, etc..

7. Node synchronization: low power sleep modes and synchronised wake up are supported by this protocol.

2.1.2 Routing Techniques

When great distances separate the nodes from the base-stations/end-devices, more powerful radio transmissions are required. However, the more power we use, the more energy we waste, and that is exactly what WSNs try to avoid; using more power. Therefore, the need of efficient routing techniques becomes imminent, and fortunately enough, there are some which are already adopted by the Zigbee and DM protocols.

2.1.2.1 Ad-Hoc On-demand Distance Vector (AODV) routing

This form of routing is used in the DM protocol to relay data from the transmitting nodes to the base-stations or end-devices [11]. In a mesh-network topology which have a similar arrangement as Figure 2.4, only a single node is in communication distance with the base-station, AODV routing is used to map the transmission path using multihop algorithms, and also maintains a list of previously used paths in routing-tables.

Path discovery is achieved by means of flooding the network with broadcast messages until the final destination is discovered. The path is determined for the delivery of data from the source node to the base-station where it could take an N-amount of hops before it is received. In the DM protocol, it is possible to specify the maximum amount of hops allowed between routing nodes until the final destination, and once this value is exceeded the data get dropped. In addition, DM

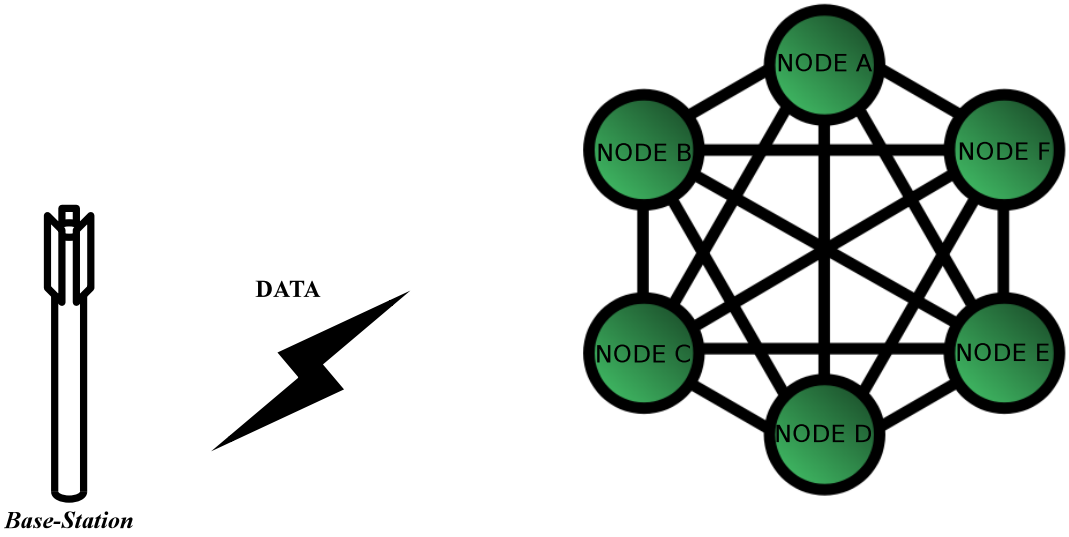


Figure 2.4: AODV in mesh networks.

enables multi-route requests in order to select between various paths which are discovered; the source node broadcasts a *Route Request* (RREQ) message which is received by any node within communication range. Various paths are determined based on the routes found by each node, then a *Route Reply* (RREP) gets transmitted back by the destination node which contains a list of all the paths discovered. The source node receives the RREP and the path with the shortest *Round Trip Time* (RTT) is selected to deliver the data.

The IEEE-802.15.4 protocol does not specify any routing techniques for data transmission as this standard deals exclusively with the physical and MAC layers in communications. Ultimately, it is up to the user to define the routing algorithms used by network devices and their applications. Both DM and Zigbee protocols provide additional upper layer support including specifications for the network (NWK) and transport layers. In addition, they both address lower layers (in the MAC and PHY) using the original 802.15.4 specifications standard.

2.1.3 Power Consumption in Wireless Sensor Networks

Energy is a primary concern in WSNs. Sensor nodes have limited energy to supply and hence they must make intelligent decisions which can help conserve energy. It is found that general knowledge on the current consumption of the different elements on the sensor nodes will improve the node lifetime by up to 52% [12]. Therefore, it is important to realize techniques that minimize power consumption in the microprocessor, analog-to-digital converter (ADC), and radio. The methods discussed in [13] suggest that power consumption is greatly affected by the ADC, micro-controller, and radio of a sensor node. Several methods are put forward in order to optimize network performance without the trade off in increasing energy demand. Efficiency can be improved using several methods which are summarized below:

- An ADC is an electronic device that converts analogue signals into digital signals proportional to the magnitude of the voltage or the current. Different types of digital coding schemes can be used such as binary, gray code or two compliment's binary. Furthermore, the ADC requires constant energy supply to maintain normal operation, hence if a device suffers from low efficiency setbacks then the ADC will continue to draw current from the power source to maintain operation and eventually depletes the power source.

- Efficient programming of the microprocessor can reduce energy demand from the node components such as the external sensors, and the radio. The energy is reduced by promptly instructing the radio component to switch off when data is not being sent or received. In addition, other node components such as sensors, GPS/GPRS modules are turned off when they are not in use.

- The radio component is the biggest energy consumer in a wireless sensor node. In WSN the radio operates in 4 different modes: receive, transmit, idle, and sleep. Radio idle is the biggest contributor to energy loss out of the remaining modes. Furthermore, energy loss is also apparent when the different modes are changing from one state to another. The key to lowering power consumption in the radio component is by reducing the duty cycle. Duty cycle is the ratio between the total on time sending and receiving data to the total time in 1 complete cycle. In order to achieve a low duty cycle the radio must sleep the majority of the time, then wake up quickly to process information and finally go to sleep again.

2.1.3.1 Improving energy efficiency for WSN

The report from [13] also discusses a number of techniques which are applied to the OSI networking protocol stack on power management solutions. The OSI model consists of a number of interrelated protocols implemented in current architectures of modern computing systems. Furthermore these protocols provide the basic structure of how each different layer interact with one another starting from the physical layer (PHY) all the way to the application layer. The application layer typifies the use of software applications in the windows operating system environment as an example, while the physical layer defines the electrical and physical characteristics of the hardware. The other functions include routing of data which occurs in the transport layer, error-detection techniques in the data-link layer, and the transferring data from one source to its destination is in the network layer. It is possible to improve power consumption of sensor networks through techniques applied onto these different layers. These techniques can be summarized as follows:

- Load partitioning is a technique applied on the application layer which in relation to WSN means that power commutation and data processing are performed at the central nodes rather than the remote nodes. This method entails that remote nodes are responsible for sensing and transmitting data only. Later the information are requested by the central node where it would process the data accordingly. Another method known as context information

is based on the level of activity expected in a network. If there is no activity it would shut all the radio components of the network.

- Techniques to try to minimize the number of retransmission of packets due to a faulty connection in the wireless link are applied in the transport layer.

- Designing of intelligent routing algorithms and adaptive routing techniques are used in the networking layer of the OSI model. In multi-hop routing, every node act as a routing device in order to route information from the source through multiple nodes and finally to the final destination. This technique is proven to reduce energy consumption used by sensor nodes. There are two methods used to reduce power consumption in the data link layer: Automatic Repeat Request (ARQ) and Forward Error Correction (FEC). These methods improve efficiency by reducing the transmission overhead. Both ARQ and FEC reduce the number of error packets at the receiving node consequently conserving more energy.

- Sleep scheduling is a technique applied in the MAC layer of the OSI model and the main function is to optimize the duty cycle by tweaking the on and off time values of nodes which reduce the effects of idle-listening. Synchronous sleep scheduling requires clock synchronization of all nodes within the network. Asynchronous sleep scheduling uses continuous transmission to wake up the receiving node when needed.

- Proper hardware design will reduce the amount of current leakage in electronic devices and that will ensure a longer lifetime is achieved. Remote Access Switch (RAS) is technique used in the physical layer to reduce power consumption. It does so by waking up the receiver only when it has data destined to it.

2.1.4 Real-Time Operating Systems (RTOS)

An operating system is simply a group of programs that are connected together to provide the user functionality of the hardware system. WSNs use different types of operation systems in comparison to desktop computer (e.g-Windows); sensor nodes generally lack the memory space and raw processing power to use an operating system like Windows. Therefore the operating system must be sufficiently small to fit in the memory of the sensor node whilst it handles the tasks the same way the desktop or laptop OS does. The most popular of those operating systems are Contiki and TinyOS.

2.1.4.1 Contiki

Contiki is an open source, highly portable, multi-tasking operating system for memory-efficient networked embedded systems and wireless sensor networks [14]. Contiki is implemented in the C language and has been ported to a number of micro-controllers such as the MSP430 by Texas Instruments. It is built around an event-driven kernel, and other features including dynamic program loading/unloading, and pre-emptive multi-threading.

2.1.4.2 IPv6 stack

The internet layer of the OSI model consists of a number of protocols which are responsible for the transport of packets from the originating host to a specified destination. The IP protocol in specific is used for delivering packets from the source host to the destination solely based on their addresses. Contiki's implementation of the TCP/IP protocol is the μIP which is suitable for sensor nodes and other resource-limited devices [15]. It is designed to have only the absolute minimum of required features for a full TCP/IP stack, and focuses on the TCP, ICMP and IP protocols. The difference between IPv6 and IPv4 is the amount of address space. IPv6 has a total address space of 2^{128} addresses compared to 2^{32} in IPv4. Currently, Contiki is one of few operating systems that support IP networks.

2.1.5 Communication Performance Measurements

We rely heavily on scientific instruments in order to measure, analyze, and record data concerning communication systems. Hardware-oriented equipment such as a digital storage oscilloscope (DSO) is heavily used in analyzing RF signals and transmission frequencies of radio modules; to verify their operation within specific requirements. Moreover, network simulators are indefinitely one of the most important tools around; they utilize numerous communication protocols fundamental in many network deployments [16]. The main parameters which determine the performance quality of communication systems are outlined as follow:

2.1.5.1 Energy estimation model

Total energy use is estimated based on key parameters which include: microprocessor energy utilization, transmission energy, sensors usage, and consumption by periphery components (e.g., external memory, boards, etc.).

$$\frac{E}{V} = I_m * t_m + I_l * t_l + I_r * t_r + I_t * t_t + \sum (I_{ci} + t_{ci}) \tag{2.1}$$

Equation 2.1 is a general energy estimation model which is utilized towards computing the total consumption by the Contiki-OS kernel [17]; the parameters which are affected are as follow: a. CPU usage, b. LPM (Low Power Mode), and c. Transmit/Receiver (Tx/Rx).

2.1.5.2 Radio Reception Throughput (RRT)

RRT desribes is a relative measurement which determines the quality of transmission in RF transceivers. RRT does not reflect the data rates in communication [18]; rather it is a quantitative reflection towards transmission between interconnected devices.

$$RRT\,(\%) = (R_x/T_x)_{packets} * 100 \tag{2.2}$$

Equation 2.2 specifies the RRT parameter in terms of the ratio between received packets and total number transmitted.

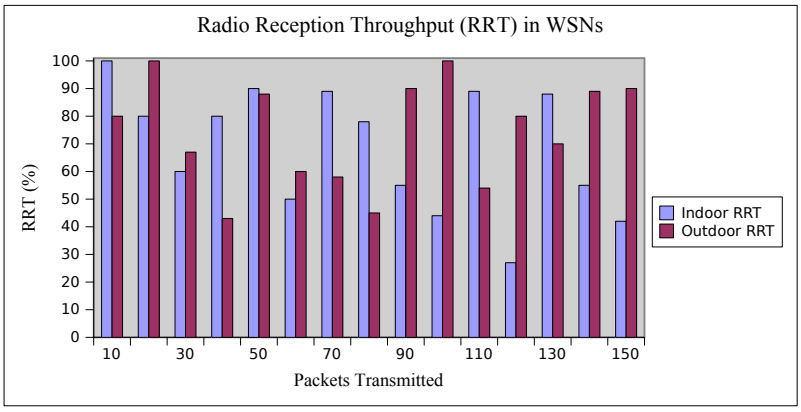


Figure 2.5: RRT demo based on WSNs.

In Figure 2.5, the RRT is measured in two separate environments; i.e., indoor and outdoor. These measurements are based on point-to-point (p2p) communication links between the source (Tx) and the receiver (Rx). As mentioned previously, the RRT does not reflect the real data-rate, however it provides a measure of quality regarding the communications in a specific environment.

2.1.5.3 Radio duty-cycle

Duty-cycle is a time-based parameter which applies to the process in which the radio device is handled [19] [20]. There are two key parameters involved in the calculation of radio duty-cycle which include: total time (t_{total}) and the active duration (t_{tx} and t_{rx}); radio receive and transmit duty-cycles are expressed by Equations 2.3 and 2.4 respectively.

$$D_{Tx} = t_{tx}/t_{total} \tag{2.3}$$

$$D_{Rx} = t_{rx}/t_{total} \tag{2.4}$$

Duty-cycle algorithms specify key techniques towards developing energy-smart systems; WSNs have utilized these methods to improve efficiency and transmission energy in the radio modules [21]. WSNs are at their peak efficiency (i.e., in terms of energy usage by the radio component) while the duty-cycles are at their lowest specifications; by optimizing durations specified in sleep (t_s) and wake (t_w) parameters [22].

In the discussion to follow, we review the work of authors in the WSN field, which involves architecture design, applications, and main challenges.

2.2 Systems Architecture, Applications, and Challenges

Recent advances in sensor technology and wireless communication have inspired significant research interest in WSNs due to their promising potential to support an extensive range of applications. A new generation of WSN technology is becoming more apparent to the community, and they are demonstrating an excellent integration capabilities with a wider range of communication protocols and applications services. Application oriented technology including WSNs are currently used in many sections of the community including in environmental monitoring [23], assistance with Activities of Daily Living (ADLs) [24], smart metering [25], and human safety applications [26].

A. V. Andras Nasa et al. [27] proposed a technique that used WSNs for pollution monitoring. Their technique measures the concentrations of gases in the exhaust systems of mobile vehicles (cars, buses). The data will be used to understand effects of pollution on the air quality in industrial cities. Authors however, do not use techniques to deal with a large number of nodes, and in particular the infrastructure requirements based on their methods. *R. North* [28] proposed another technique to support the management of transport and air quality. His technique incorporates IP networks with Wi-Fi, Zigbee, and mobile-3G communications to improve the interoperability between different technologies. The author however, does not specify the techniques for power management given that devices are required to transmit data on frequent basis, also the inefficient nature of Wi-Fi/3G communications in terms of energy use. WSNs are also used indoors to monitor the health in patients and provide them with assistance in ADLs [29]. *L. J. Wu Zhengzhong et al.* [30] proposed a system based on low power RF transceivers and sensors for monitoring the indoor air quality. Their techniques improve the power efficiency in the WSN by lowering the duty-cycle in the radio transceiver units. Authors however, do not specify which of the available Real Time Operating Systems (RTOS) they used to implement their IP based sensor network, and they also failed to address the communication architecture between the IP gateway and the nodes. The technological demands poised by the industry can sometimes be challenging and far-reaching. Nowadays applications based on WSNs require new type of technology that can withstand tough weather conditions, offers dynamic resiliency to changes, and uses long/short distance communications at very low power specifications. WSNs are now found in the toughest and most brutal environments such as mines [31], underground facilities [32], and offshore rigs. *S. C. Mohammad reza Akhondi et al.* [33] explores the applications of WSNs in the oil and gas resource industry. The author argues for the use of sensor network technology to monitor the pipeline integrity, natural gas leaks, corrosion, equipment self-status, and prediction in reservoir levels. WSNs offer a very marketable prospect in the eyes of leading industries and organisations, which in turn fueled the research into safety and environmental monitoring applications including chemical processes [34], waste disposal management, materials and products transportation [35], and bush-fire monitoring [36]. *L. Liu* [37] realizes the important roles WSNs play in the development of environmental applications, and argues for the need of an architecture that is based on a hierarchal structure and sub-facets functions in order to address the main objectives of WSNs including

customized-service functionalities, power efficiency, and adaptive to dynamic changes. The author also argues that research into the architecture of WSNs is fundamentally weak, which is a matter that needs to be addressed in the future. The main concerns raised by the author include firstly the mapping of operations between the environment, network, and application. Secondly, lack of sustained power resource to guarantee a high quality of service and efficient use of bandwidth. Thirdly, the architecture of WSNs should be scalable in size, network functions, and network performance. Lastly, the use of service-customized to meet with different application needs. Let us take this opportunity to recap on the shortfalls of WSNs; first and foremost, we present the audience with the limitations of existing schemes which are used in applications involving WSN technology [38, 39]. These shortcomings are outlined below:

1. The limited energy supply from the sensor nodes will restrict the lifetime of the WSN.
2. The transmission range between sensor nodes is limited, thus a large number of nodes would be required to cover greater areas .
3. Many-to-one communication patterns in which multiple nodes send their data to a single sink can cause unbalanced energy consumption, channel contention, and higher collision rates.
4. WSNs are prone to failures due to many factors such as noise interference, node device failures, and inefficient programming of software.

In the next few chapters, we will address several challenges based on the literature discussed earlier. We take exceptional notice towards architecture design of applications which involve environmental monitoring, thus we will be presenting three different systems: **a.** Air pollution monitoring, **b.** Fire monitoring in indoor and outdoor environments, and **c.** Gas leak detection. We target the use of a unified structure which specifies low-power communication protocols, multi-hop routing techniques, and integration with TCP/IP networks in order to enhance real-time utilization and event multi-tasking. We will also make use of distinct algorithms for enabling efficient sensing operations, which target improving reliability of data and reducing the number of false alarms during emergencies. Finally, we will incorporate over the air programming (OTAP) and long distance connectivity together in order to address network errors, faults and troubleshooting from remote locations.

Chapter 3

APPLICATION FRAMEWORK

In this chapter, we will present the audience with a systematic framework which specifies major components in the communication architecture to enable real-life implementations of WSNs and their applications. We direct our efforts towards developing a unified system based on off-the-shelve hardware, in which we use to address fundamentally the practical side of WSNs. Figure 3.1 represents a conceptual model which embodies the architectural design of WSNs in general form. There are four major components of this system which include:

1. Networking architecture: Our work emphasizes on relationships between networked devices in WSNs. We focus our attention on methods which assist with network formation, configuration of radio modules, and functionality of hardware platforms (specific to WSNs).

2. Alarm architecture: The audience will acknowledge the multiple techniques we utilize for generating alarm signals. We target online services (Twitter, word-press), mobile communications (SMS, voice calls), and wide community services including personalized web-servers, daily e-mail updates, and live data access.

3. Power management: Our work fixates on management strategies towards energy optimization and supportive algorithms for reducing power consumptions in sensor nodes. We utilize a dynamic detection technique which assists with the execution of node functions based on the available energy in the reservoir. In addition, we target the use of sleep modes (regular, deep sleep, hibernate), and incorporate Real-Time-Clock (RTC) to provide synchronization for the sleep and wake durations.

4. Sensing architecture: We seek to inform the audience with regards to the calibration of sensors, implementation in the environments, and finally the interpretation of measurements.

We will begin this chapter with an introduction on the available hardware which specializes in applications based on WSNs, then we move towards more advanced discussion which involves the main system architecture. Many of the sections include pieces of software codes, these were developed to guide the audience (whom are interested) through the necessary steps to replicate this work.

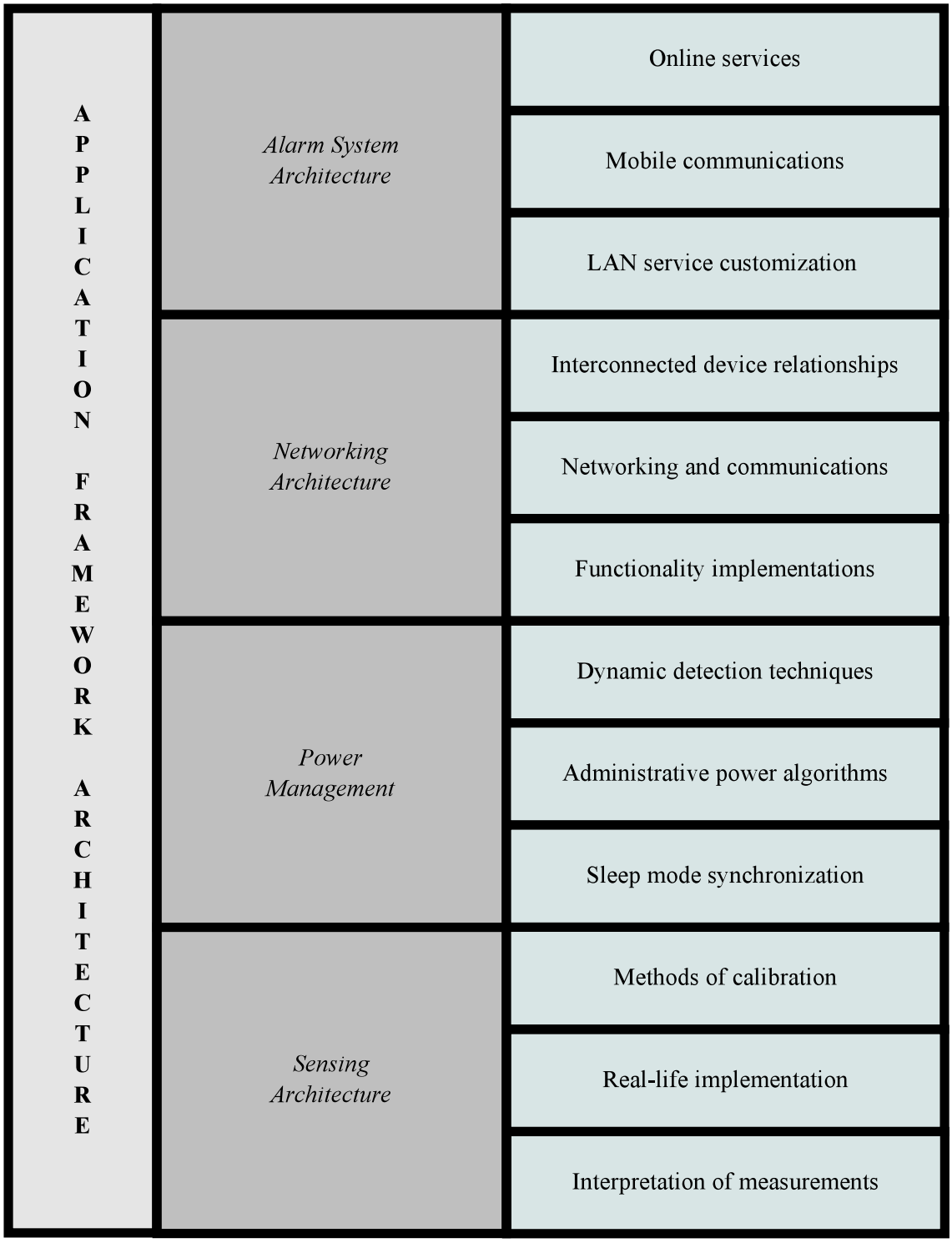


Figure 3.1: Unified framework model.

3.1 Overview of Hardware

Popular WSN sensing platforms such as the TelosB (Berkeley), MicaZ (CrossBow), and Waspmote (Libelium) have all made their mark on the WSN community. The TelosB is one of the earliest, and most widely used platform model around the world. The telosB and micaz nodes demand little

power to operate in real life and they both share a simple approach in terms of their hardware design. However, unlike the Waspmote, these nodes do not support a large variety of sensors (by default), and that makes them less suitable in real applications. In addition, the Waspmote supports a variety of radio communication standards and modules (802.15.4 / Zigbee / DM / WiFi / Bluetooth), where the TelosB or MicaZ only use an 802.15.4 radio transceiver. Since our research is focused more towards application development, we opted with the Waspmote platform as the main hardware.

3.1.1 Waspmote

The Waspmote [40] is an open source Zigbee and IEEE-802.15.4 sensor platform which is based on a user-friendly modular architecture for rich application development. The classification of the external boards are as follows: gases, events, agriculture, smart metering, smart parking, and smart cities. These boards hold various types of sensors which are mounted on the Waspmote to give it this compact feeling and neat touch. In addition, Waspmotes support a number of protocols and standards of communications such as, the IEEE-802.15.4, Zigbee, Digimesh, Bluetooth, and RF-XSC. The ability to integrate many protocols into a single node provides capabilities for implementations of multi-WSN architectures where independent communications can exist within a single network. The main features of the Waspmote node include: module architecture, simple sensor integration, support of numerous communication protocols, secured transmission using AES (Advanced Encryption Standard), long range communication capabilities [41], low power consumption MC (ATmega1281 microcontroller), and finally support to over the air programming functions (OTAP).

3.1.2 Meshlium

Meshlium [42] is a powerful WSN router which combines many communication technologies into one single device. The supported technologies include: Wi-Fi (2.4 GHz and 5 GHz), Zigbee, GPRS, Bluetooth, and GPS. It is based on a 500 MHz processor with 256 MB of RAM and runs a Debian/Linux Operating System (OS). It is designed to withstand harsh environments using an aluminum casing to protect it from dust and water. Furthermore, Meshlium allows a large number of high level services to be managed such as, sending alarm SMS and e-mails, accessing the World-Wide-Web (WWW) services, and managing MYSQL databases for data storage. The main functions of Meshlium include:

1. Initiating of short and long distance data communication.
2. Routing data multiple communication protocols.
3. Capturing and storing of data from the sensor network
4. Local network control and link maintenance.

3.2 Communication Networks

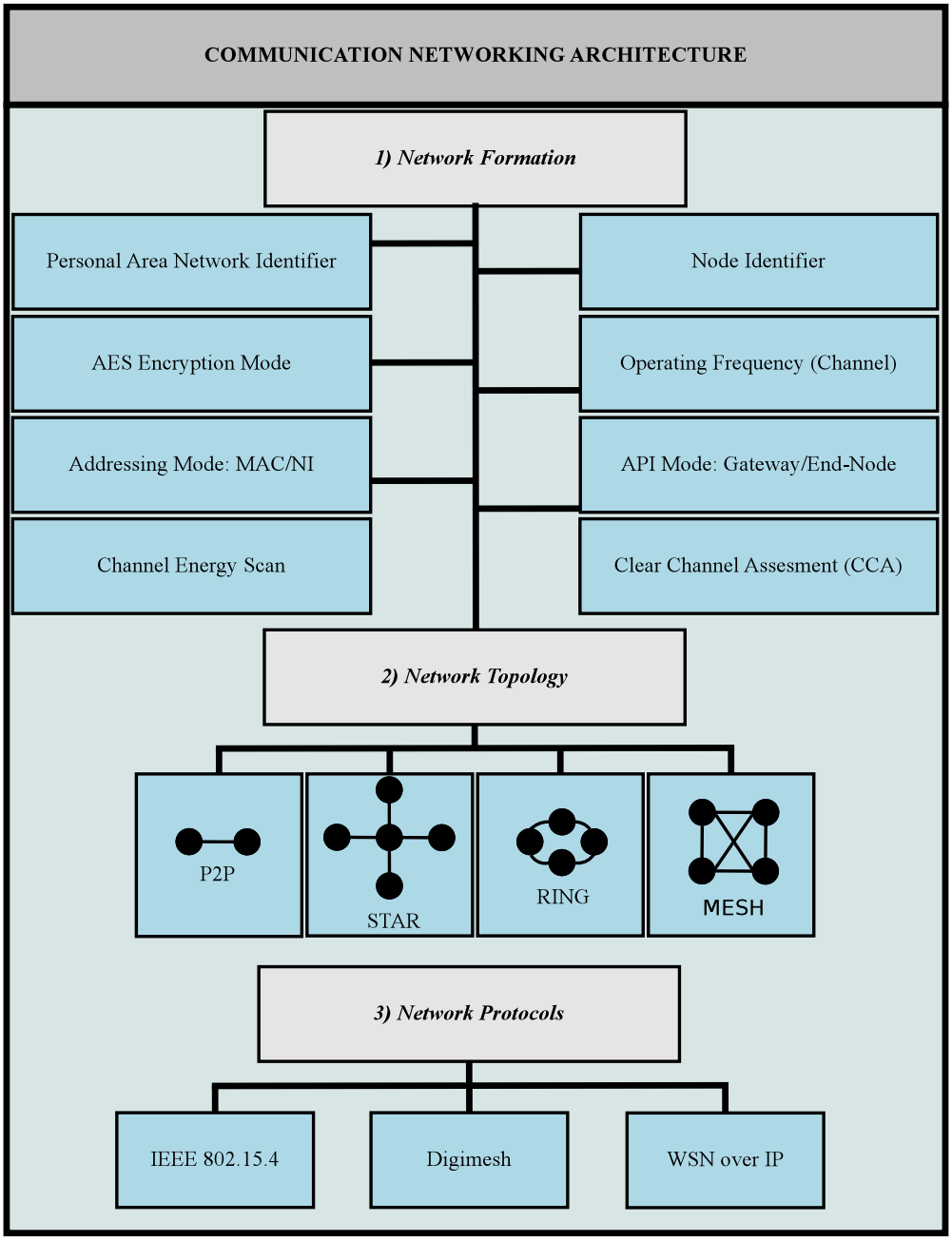


Figure 3.2: Communication architecture building blocks.

In this section, we propose an architecture which specifies the networking principles towards WSNs. Our work focuses on three major components which include network formation, topologies and protocols. We present our model in Figure 3.2 which summarizes the elements that define the WSN architecture.

We will begin our discussion with a brief introduction on how to setup the initial environment of the operating system. Firstly, the installation of X-CTU (XBee Configuration and Test Utility) is necessary in order to configure the radio modules for the WSN. To enable compatibility in Linux (mint/ubuntu):

Step 1: Install the WINE (Wine Is Not an Emulator) package to allow compatibility of Windows based applications on linux distributions. In terminal, type the following and then proceed with installation of the X-CTU:

```
sudo apt-get install wine
```

Step 2: Create soft-links between USB and COM ports by typing the following commands:

```
ls -lt /dev/ttyUSB*
ln -s /dev/ttyUSBx ~/.wine/dosdevices/COMy
//Note: ttyUSBx where x represents the USB port number
  COMy: where y can be any port number to bind with the USB //
```

In the next series of discussions, we seek to clarify the processes involved in the formation of WSNs, which include configuring the network parameters (operating channel, data transmission), data security (payload encryption), and the methods of transmission (unicast, broadcast, multicast). Then, we turn our attention towards network topologies and implementations, which include point-to-point (P2P), ring, start, and mesh architecture. Finally, we deal with WSNs over IP where we focus on methods of integration and real-time deployment.

3.2.1 Network Formation

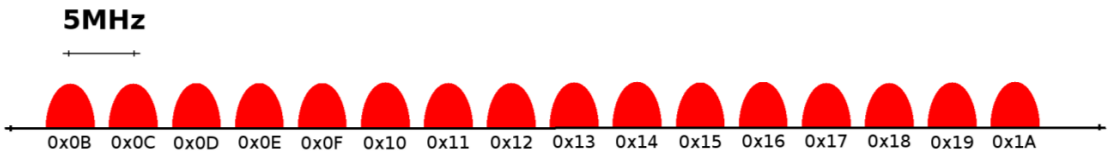


Figure 3.3: The 2.4 GHz spectrum for the IEEE-802.15.4.

The shortfall of the 802.15.4 protocol is linked with its omission of network layer (NWK) specifications that ensure homogeneity of the WSN and devices. Therefore, our main priority is to guarantee appropriate configuration in the setup of transceiver units prior to their use in real-life applications. The Digimesh (DM) protocol on the other hand specifies NWK layer functionality which enables self-managed networking operations such as routing, transmission retries, joining and leaving networks, selecting communication channels, etc. We target its use in mesh architecture which requires routing abilities and many-hops communication in order to maintain the network.

Table 3.1 outlines the requirements for setting-up a WSN based solely on 802.15.4 communications. Now, we turn our attention towards addressing these conditions by configuring the radio

802.15.4 WSN Setup Conditions	
Condition 1:	All devices must operate on the exact communication channel
Condition 2:	All devices must have the same Personal Area Network (PAN) Identifier
Condition 3:	All devices must have the same AES encryption key
Condition 4:	All transmitting devices must use API-mode2

Table 3.1: Configuration requirements in radio transceivers.

units using the appropriate programming techniques, which include:

3.2.1.1 Communication channel selection

Algorithm 3.1 Communication channel setup for 802.15.4 radios.

```
void Channel_Setup(){
  xbee802.init(XBEE_802_15_4,FREQ2_4G,NORMAL);
  xbee802.ON();
  xbee802.setChannel(0x0D); //Channel D is seletected = 2.410-2.415 GHz
}
```

The XBee radio modules operate within 16 channels over the 2.4 GHz spectrum as shown in Figure 3.3. We program the nodes to operate in the same communication channel in order to prevent data loss during the communication process. Algorithm 3.1 enables radio communication on channel-D only, but it is also possible to select other channels. It is important to examine each radio device individually to avert the possibility of data loss.

3.2.1.2 Personal Area Networks (PAN)

Algorithm 3.2 Personal Area Network (PAN) configuration.

```
void PAN_Setup()
{
  uint8_t  PANID[2]={0x12,0x34}; //PAN ID=0x1234
  xbee802.setPAN(PANID); //Setting the PAN-ID
}
```

The PAN-Identifier (ID) has a unique purpose in WSNs; it enables co-independent relationship in which an entire network-system can be divided into individual networks each associated with a unique PAN-ID and each has different duties. We target its practical application during deployment of applications which are based on multiple network topologies. We use it to distinguish between nodes and to control individual groups without affecting the entire network. Algorithm 3.2 represents the basic application on how to configure the PAN-ID of a radio unit.

3.2.1.3 Data payload encryption

Algorithm 3.3 AES encryption setup.

```
void Encryption_Mode()
{
  xbee802.encryptionMode(1); // Enabling encryption of data payload
  xbee802.setLinkKey("amroquandourECU1"); // 16 bytes key= amroquandourECU1
}
```

Node authentication and data validation are implemented with Advanced Encryption Standard (AES), which supports $128b$, $64b$, or $32b$ encryption code (b refers to bits). We target secure transmission between wireless nodes by enabling network-node authentication and payload encryption. Algorithm 3.3 enables the security features in transmission of data.

3.2.1.4 Node identification

Algorithm 3.4 Node-ID setup.

```
void NodeID_Custom()
{
  xbee802.ON();
  xbee802.setNodeIdentifier("Building1Node1"); //set the NodeID.
  int sizeID = sizeof(xbee802.nodeID); // read the node ID.
  for(int i=0 ; i<sizeID; i++){
    XBee.print(xbee802.nodeID[i]);
  }
}
```

Node-IDs are digital signatures which allow us to distinguish node devices in WSNs. We propose the implementation of an architecture to address the nodes using unique IDs which are based on type (transmitter, receiver, gateway, etc), location (outdoor, indoor, etc.), and functionality (type of sensors, routing device, etc.). Algorithm 3.4 is the method that we use to set the Node-ID in each radio unit.

3.2.1.5 Data Tx

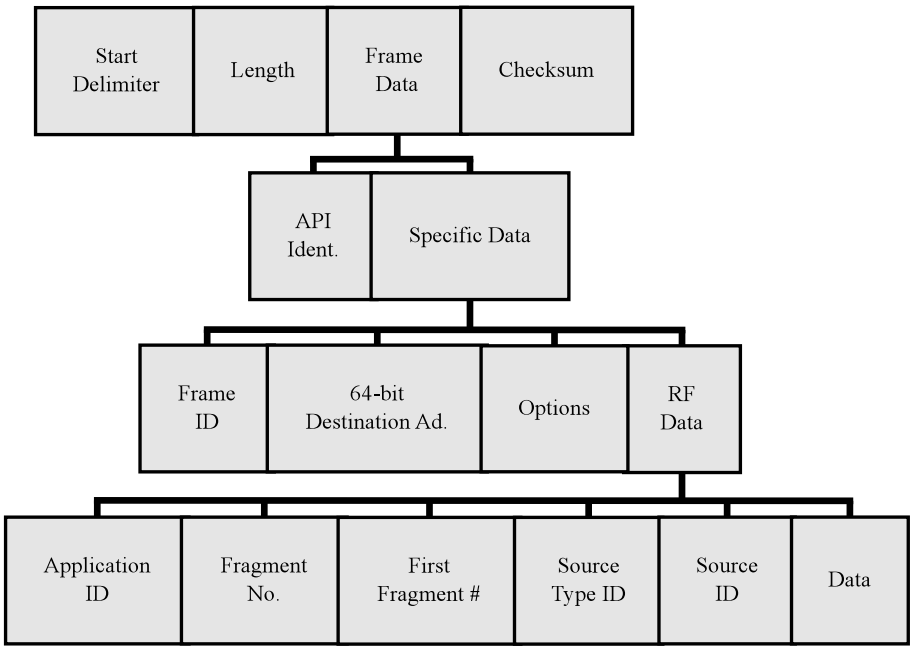


Figure 3.4: API-Frame structure of the transmitted signal.

When a node transmits data to another device; the radio device encodes the data in a special arrangement referred to as API frame. The frame depicts the data structure abstraction which specifies the order of bytes (data) in the RF signal as shown in Figure 3.4. We outline the interactions of Tx frames in the application layer in Algorithm 3.5. There are three important functions that we need to discuss:

1. Initializing the packet structure: In the Waspmote, it is referred to as 'packetXBee', which initiates the API structure.

2. Preparing the packet: In this function, we setup the transmission parameters which include the destination address, transmission methods (broadcast, unicast, multicast), number of fragments (if the size of data exceeds the maximum length), addressing type (MAC and network), and packet identifier (to distinguish different data such as fire, pollution, etc.).

3. Transmission of packet: Once the structure is complete; we transmit the signal, release the pointer, and free the dynamic memory to avoid future complications.

3.2.1.6 Data Rx

The receiver captures over the air radio signal; begins extracting data from within the API frame (Figure 3.4), and then parses them to the application layer (for users). The receiving process is outlined in Algorithm 3.6, which demonstrates the methods we use to selectively extract data fields from the API frame including source MAC address, RSSI, packet-ID, and RF data.

Algorithm 3.5 Tx data setup.

```
//FUNCTION 1:
void Packet_Structure()
{
//paq_sent = num_of_elements * len = 1 * packetXBee
  packetXBee* paq_sent=(packetXBee*) calloc(1,sizeof(packetXBee));
}

//FUNCTION 2:
void Preparing_Packet()
{
  paq_sent->mode=UNICAST; //Broadcast, Unicast, Multicast
  paq_sent->packetID=0x52; //Setting packet identifier
  xbee802.setOriginParams(paq_sent, MAC_TYPE); //Address type=MAC
  xbee802.setDestinationParams(paq_sent, <Dest.MAC>, data, MAC_TYPE,
                               DATA_ABSOLUTE); //No fragments
}

//FUNCTION 3:
void Send_To_Destination()
{
  xbee802.sendXBee(paq_sent); //Transmit data to<Dest.MAC>
  free(paq_sent); //free paq_sent from memory.
  paq_sent=NULL; //release pointer.
}
```

Algorithm 3.6 Rx data setup.

```
void Receiving_Parameters()
{
  if( XBee.available() ){
    xbee802.treatData();
    if( !xbee802.error_RX ){
      while(xbee802.pos>0){
        //Reading the Src_MSB=macSH & Src_LSB=macSL array elements
        Utils.hex2str(xbee802.packet_finished[xbee802.pos-1]->macSH,
                      macHigh, 4);
        Utils.hex2str(xbee802.packet_finished[xbee802.pos-1]->macSL,
                      macLow, 4);
        //Reading the Packet-ID byte and saving it in _PACKID
        uint8_t _PACKID = xbee802.packet_finished[xbee802.pos-1]->packetID;
        //Reading the RSSI
        uint8_t RSSI_ = xbee802.packet_finished[xbee802.pos-1]->RSSI;
        //Reading the actual Data
        char* Data = xbee802.packet_finished[xbee802.pos-1]->data;
      }
    }
  }
}
```

XBees Reserved Bytes (Hexadecimal format:0x..	
Start Delimiter:	0x7E
Escape Character:	0x7D
Transmit On (XON):	0x11
Transmit Off (XOFF):	0x13

Table 3.2: XBee reserved characters.

The receiver distinguishes between two intermediate frames by reading the frame initializer (Start delimiter) byte. If start delimiter bytes (or any reserved system bytes in Table 3.2) exist in the make-up structure of a data frame; an escape operation to change the data format will be required. We can illustrate this operation in the following example:

Example 3.1. 1) Tx Frame Structure:
Destination Add. ={0x00 0x13 0xA2 0x00 0x40 0x69 0xBC}
Byte to escape = 0x13 (reserved for software flow control)

2) Tx Escape Operation:
0x13 XOR 0x20 = {0x33}

3) Rx Frame Structure after Escape:
Destination Add. = {0x00 0x7D 0x33 0xA2 0x00 0x40 0x69 0x180 0xBC}
Byte to escape = 0x33 (indicated by 0x7D escape initializer)

4) Rx Escape Operation:
0x7D XOR 0x20 = {0x13} (orignal byte prior to escape)

3.2.2 Network Topology

We propose deployment of WSNs based on the topologies shown in Figure 3.2. Their configuration will be presented in the following order:

3.2.2.1 Point to Point

A point to point (p2p) topology is one which has single links between the sources (Tx) and the destination (Rx). Essentially, the STAR topology consists of many p2p connections where multiple transmitting devices share a common gateway. We implement p2p communications by using purely the 802.15.4 protocol. Algorithm 3.5 and 3.6 provide p2p communications based on the 802.15.4 protocol.

3.2.2.2 Ring

A ring topology consists of links between intermediate nodes forming a single pathway for data through each device. We implement the routing mechanism which handles the packets that through

802.15.4 Routing Protocol	
Function	*Description*
Initialize radio	Powers the radio module in order to accept data
Extract data	Extracts data from the RF signal and save it in an array
Scan network	Performs a network scan in the communication channel in order to identify the neighbouring nodes
Analyze RSSI	Measures the RSSI between the neighbouring nodes and selects the node with the best reception
Forward data	Transmit data to the selected node

Table 3.3: 802.15.4 routing functions.

the nodes in the ring network. As mentioned earlier, the 802.15.4 protocol does not specify routing methods, hence we developed our own protocol to perform the routing tasks. The protocol is based on the five functions highlighted in Table 3.3.

3.2.2.3 Mesh

A mesh topology consists of numerous nodes which communicate in two ways: a) direct links and b) indirect links. Direct links correspond to an immediate connection (single hop) between nodes, whereas indirect links utilize multiple hops in order to communicate. The reader should note that we opt with Digimesh (DM) not 802.15.4 in order to deploy a mesh sensor network as the latter does not support any self-routing protocols or a network stack. We also need to clarify that in order to use the DM protocol; firstly, the XBee radios must be flashed with the DM firmware using X-CTU prior to use in real-life deployment. Secondly, the ability to use two separate protocols (in our case 802.15.4 and DM) depends on the internal hardware of the radio unit (use series-1 modules).

3.2.3 Network Architecture

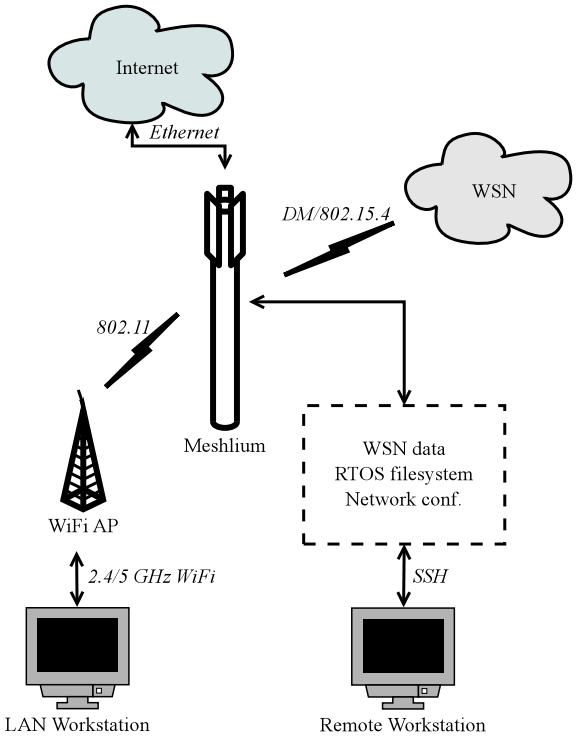


Figure 3.5: IP-WSN system architecture.

In this section, we propose a structure for an IP-WSN as shown in Figure 3.5, which is based on the following communication protocols: a) TCP/IP, b) 802.15.4/DM, and c) 802.11. The aim of this system is twofold:

1. Facilitate remote access to the gateway (meshlium) through SSH connections and enable users to adjust network parameters, read sensor data, and perform maintenance tasks.

2. Establish a shared resource management scheme in the Local Area Network (LAN) to provide a simplistic mechanism for data access and alarm services.

Firstly, we will address the work in the gateway node, then we indulge ourselves with remote communication technologies, and finally we discuss a suitable application for this architecture.

3.2.3.1 Method for handling data in the gateway node

We refer to the meshlium node in 3.5 as the gateway of the WSN. Its main functions include: a) stores and organizes the sensor data, b) provides internet access to clients, and c) exports data to databases and web-servers. The process we are about to discuss involves firstly the construction of Tx-data frame, and then we address the methods for handling them in the gateway. The Tx frame inherits a customized structure as highlighted in 3.6; our methods focus on dissecting the Tx frame into individual data components, integrate them into an external database, and ultimately produce a dynamic webserver where data is continuously updated.

Algorithm 3.7 Data processing method in the gateway.

```
void parseFrame(struct xbee_frame *p_frame) {
  char *_Temp;
  char *_CH4;
  char *pstart;
  int k = 0;
  int error=0;
  char key[] = "***" ; //Start delimiter

  pstart=strpbrk(aux,key); //Locates the start delimiter.

  if (pstart == NULL) error = 1;
//TEMPERATURE FIELD:
  if (!error){
    _Temp = strchr(pstart, '&'); //locates the end delimiter.
    if (_Temp != NULL) {
      k = _Temp - pstart; //Determines the width
      strncpy(p_frame->Temp,pstart+3,(k-3)); //Save into Temp
      p_frame->Temp[k-3] = '\0'; // end of data component
    } else
    error = 1;
  }
  pstart=_Temp + 1; // Pos. of next data component.
//METHANE FIELD:
  if (!error){
    _CH4=strchr(pstart,'&'); locates the end delimiter.
    if (_CH4 != NULL) {
      k=_CH4-pstart; //Determines the width
      strncpy(p_frame->CH4,pstart, k); //Save into CH4
      p_frame->CH4[k] = '\0'; //end of data component
    }else
    error = 1;
  }
  pstart=_CH4 + 1; // Pos. of next data component.
```

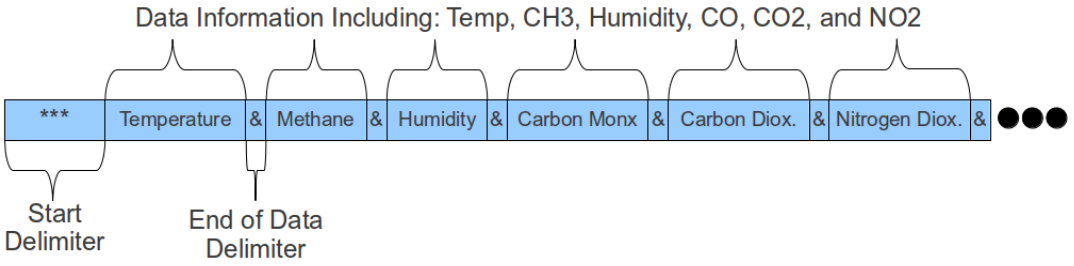


Figure 3.6: Data frame structure.

Recall from Figure 3.4 the structure of the entire frame. From our point of view, we are only interested with the contents of RF-data field which contains the real data measurements. There are three important fields in the structure from Figure 3.6 which include:

1. Start delimiter: 3 byte field "***" which marks the beginning of the RF-data frame.

2. End delimiter: occupies 1 byte "&" which marks the end of each data component.

3. Data component: contains the real measured value of a sensor.

As the frame gets inspected by the gateway; first, it determines the initial position of the frame (start delimiter), then it locates the end delimiter in order to calculate the length of the data component, and finally it copies the contents of the data component in a separate variable. We highlight this process in Algorithm 3.7 which refers to the temperature and methane gas fields only.

3.3 Over the Air Programming

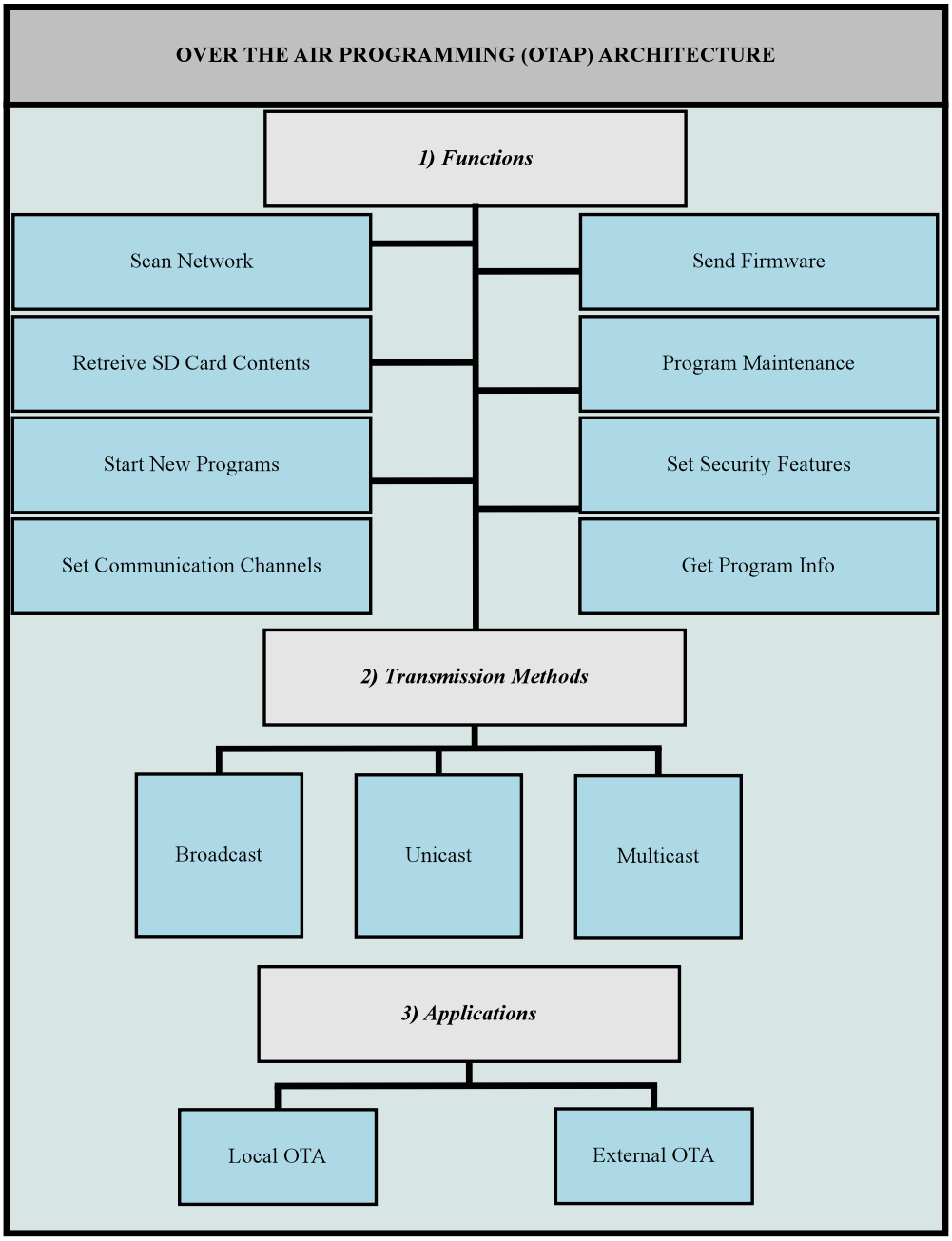


Figure 3.7: Over the air (OTA) architecture.

In this section, we propose the system architecture of Over the Air Programming (OTAP) which is shown in Figure 3.7. The OTAP concept is used extensively in WSNs to update firmware and patch security vulnerabilities in sensor devices. Our aim includes: a) outline the main functions of the OTA-shell, b) address the transmission methods used in OTA, and c) integrate OTA with WSN applications.

3.3.1 Functions

The main functions of OTA will be presented in the following order:

3.3.1.1 OTA scan network

This function scans the communication channel and obtains the nodes which operate within it. In return, we obtain the node's MAC address, identifier (NI), program information, and OTA status. The method of use and the output is demonstrated by Example-3.2.

Example 3.2. Scan Network:

```
./otap -scan_nodes --mode BROADCAST --time 10
```

```
Total Nodes: 2 - Time elapsed: 15s
0   - Node 0013a200406918c9 - NODE1 - prog000 - READY
1   - Node 0013a200406918bc - NODE2 - updater - READY
```

3.3.1.2 OTA retrieve boot-list

This OTA command retrieves the list of programs that are stored inside the node's μSD memory card as shown by Example-3.3.

Example 3.3. Retrieve Boot-list:

```
/otap -get_boot_list --mode UNICAST --mac 0013a200406918c9
```

```
Bootlist Node 0013a200406918c9 - length: 8
0   - PID:updater  - Date:2011/06/21 20:31
2   - PID:pwrlevl  - Date:2011/06/16 01:25
4   - PID:prog001  - Date:2011/06/21 20:31
5   - PID:prog002  - Date:2011/06/16 01:25
6   - PID:prog000  - Date:2011/08/18 01:43
7   - PID:prog005  - Date:2011/09/09 20:13
```

3.3.1.3 OTA send new programs

This OTA command upgrades the software database of the node device. In addition, it can be used to set encryption mode, modify authentication keys, and change the communication channel of the node. In Example-3.4, we outline the syntax of the OTA scan command.

Example 3.4. Send Syntax:

```
./otap -send --deliveries <n> --file <name.hex>
--mode <UNICAST|MULTICAST|UNICAST> --new_authkey <Access_Key>
--new_channel <00|0x00> --new_enckey<Encryption>
--pid <stored_name> --send_mode <BINARY|ASCII>
```

3.3.1.4 OTA start new programs

This OTA command initiates a specific program (PID) from the node's μSD memory as shown in Example-3.5.

Example 3.5. Start Program Synatx:

```
./otap -start_new_program --mac<MAC_ADDRESS> --macs_file <list.txt>
--mode <UNICAST|MULTICAST|BROADCAST> --pid <program_name>
```

3.3.2 Transmission Methods

The OTA shell permits three types of data transmission mechanisms as shown in Figure 3.7, which includes: a) broadcast, b) unicast, and c) multicast. First, we analyze each type, then we highlight the conditions that match their suitability, and finally discuss their applications.

3.3.2.1 Broadcast

In broadcast transmission, the OTA data is sent to multiple nodes at once; this applies to all nodes within a single hop from the transmitter node. Let us assume that in a WSN there are 100 nodes; we use broadcast transmission when each node requests the same OTA data. In addition, the number of bit-stream (data) transmissions is equal to the amount of nodes in a WSN (100 nodes = 100 copies of the data). Therefore, broadcast transmission is reserved to important firmware updates and similar application architecture (same sensors, radio transmission, etc.).

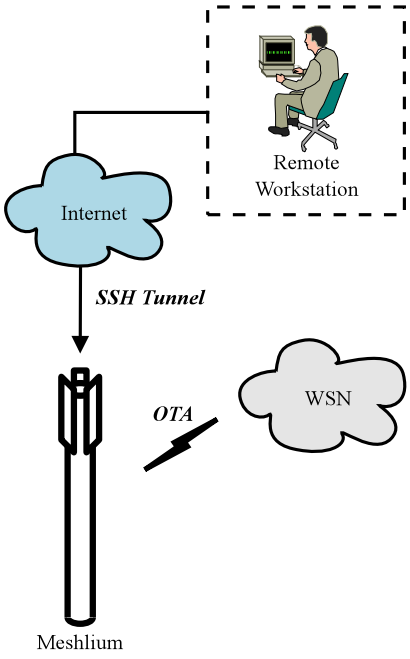


Figure 3.8: OTA WSNs applications.

3.3.2.2 Unicast

Unicast transmission is similar to point to point architecture in which communication is apparent between two devices at any given moment; an OTA transmitter, and the node device. It is suitable when WSNs consist of varying node architectures; that is, when each node involves a different application.

3.3.2.3 Multicast

Multicast is defined as multiple unicast links; in this type however, there is only one OTA-transmitter and multiple node devices. Unlike broadcast, we can target a specific group of nodes when OTA is required. It is suitable if we have a group of nodes which share the same type of sensors, radio modules, and applications.

3.3.3 OTAP Applications

We propose two applications using OTAP: a) local OTA node, and b) OTA over remote links. In terms of functional abilities, their roles are identical (functions). However, their integration in WSNs are different. The main functions of these two applications are:

1. Maintain a software database in the node devices.
2. Apply firmware upgrades towards node devices.
3. Modify the communication parameters in the radio devices including communication channels, PANs, authentication and encryption keys.

3.3.3.1 OTA in WSNs

Our first application involves performing OTA in the same environment as the WSN (i.e., same location). There are two setup requirements involved prior to the use of the OTAP shell which include: a) OTA Tx configuration, and b) node device configuration. The transmitter unit's (OTA Tx) configurations include:

* The API mode must be set to mode=1 (using X-CTU).
* The communication channel in which the nodes operate on.
* The authentication key between the two applications (application layer security).
* The PAN identifier of the node devices.
* The encryption key of the radio unit (NWK layer security).

Algorithm 3.8 Node device OTA configuration.

```
void Checking_for_OTA()
{
  xbee802.checkNewProgram();
  if( XBee.available() )      {
    xbee802.treatData();
    while( xbee802.programming_ON  && !xbee802.checkOtapTimeout() )
{
      if( XBee.available() )          {
        xbee802.treatData(); }
    }
  }
}
```

The second requirement regarding the configuration of the node device is fulfilled by adding Algorithm 3.8 into the main application code. This enables the OTA request interrupt subroutine in order to accept the radio packets, re-build the data stream, and save it in the μSD memory.

Our second application involves long distance OTA operations as shown in Figure 3.8. In this arrangement, we transform the network gateway (meshlium) into an OTA-Tx transmitter unit which transmits at a much higher power than regular nodes (longer range). We access the gateway through SSH (secure shell) from a remote workstation (i.e., outside the network) in order to perform OTA transmission. This method however involves two requirements which include: a) configuration of gateway as OTA-Tx, and b) configuration of network in order to SSH requests to the gateway IP address. The processes which enable OTA-Tx mode in the gateway are highlighted by Example-3.6.

Example 3.6. Changing Tx modes in meshlium:

1) Access the SSH environment of the gateway:

```
ssh <IP_ADDRESS_OF_MESHLIUM>
```

2) Disable (not power down) the XBee radio module temporarily:

```
/etc/init.d/ZigbeeScan.sh stop
```

3) Access the radio module to change its mode:

```
capturer S0 38400
+++ // probes the module
atap 1 // change to mode 1
atwr // write new values
```

3.4 Power Management

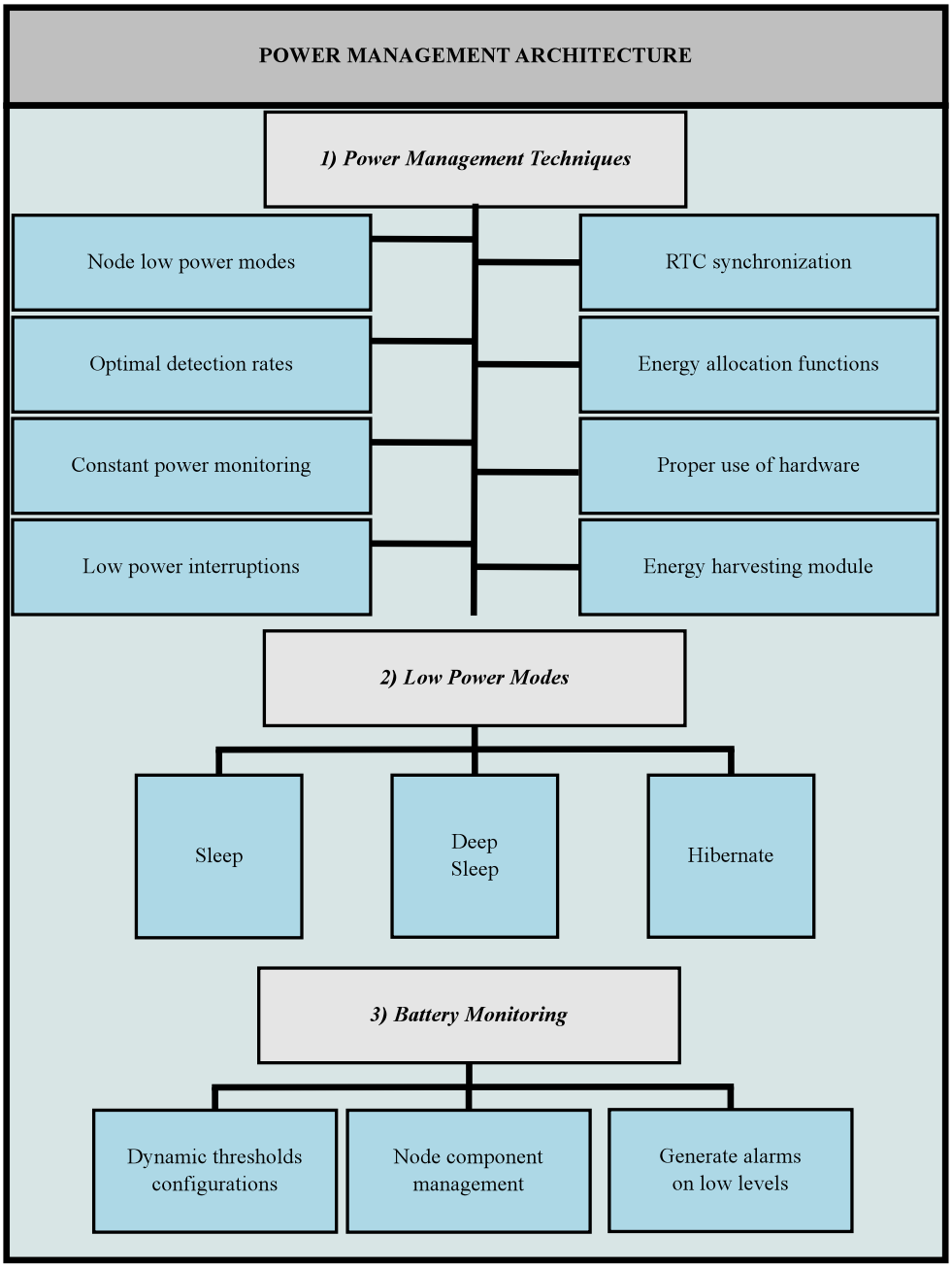


Figure 3.9: Power management architecture.

Energy is a primary concern in WSNs; sensor nodes have a limited supply of energy which has to last over a certain length of time. In this section, we propose the power management architecture shown in Figure 3.9 which focuses on addressing the energy requirements towards WSN applications. Our main aim involves optimizing the rate of change in the node battery's lifetime by deploying three low power techniques which include:

* Utilization of low-power modes: This consists of three sleep modes including normal sleep, deepsleep, and hibernation.

* synchronised detection rates: which applies to the number of sensor measurements per unit time.

* Event driven execution: which utilizes a battery detection method to check the energy availability prior to processing node functions.

We begin our discussion first by outlining the RTC's (real time clock) role towards synchronous communications, then we address the applications of the three low-power modes from Figure 3.9, and finally we develop the algorithms which monitor the node's battery levels.

3.4.1 Optimal Detection Rates

The lifetime expectancy in a node device (for WSNs) depends largely on: a) Hardware architecture, b) programming of the device, and most important c) handling methods towards additional node components such as GPRS, GPS, sensors, and radio modules.

Synchronous Detection Rates
Mode 1: Based on date (i.e., dd:mm:yy).
Mode 2: Based on day of the week (i.e., Mon-Sun).
Mode 3: Based on time (i.e., hh:mm:ss).

Table 3.4: RTC alarm modes.

Many of the components mentioned earlier demand exuberant amounts of power which could deplete the battery very rapidly; therefore, we regard their management as top priority to ensure the battery's longevity. Our work takes into considerations regulation methods towards "power-hungry" components, this implies disconnecting the modules from the power source while they are not being used (to save power) and optimizing the detection rate. We opt with the RTC to synchronize the time requirements for utilizing the functions by supplementary modules and as presented in Table 3.4. There are two steps necessary prior to administering the RTC with the applications which include:

3.4.1.1 Configuring the frequency required for detection

RTC programming involves two unique and independent alarms to assist us with developing the algorithms. These alarms can be configured according the specifications of Table 3.4. In Algorithm 3.9, we initialize the RTC, set the current time and date, and finally we active the alarm. As it is currently configured, the RTC causes an interrupt to occur once every minute. We address the contents of the interrupt subroutine next.

Algorithm 3.9 RTC detection frequency configuration.

```
void setting_time_parameters()
{
  //Powers the module
  RTC.ON(); //Powers the module

  //Writes the time & date parameteres into the internal registers
  RTC.setTime("11:08:26:06:12:00:00");

  //This alarm sets detection to 1 minute intervals
  RTC.setAlarm2("06:12:01", RTC_OFFSET, RTC_ALM2_MODE4);
}
```

Algorithm 3.10 RTC subroutine configuration.

```
void RTC_Subroutine()
{
  if( intFlag & RTC_INT ){ //Captures the interrupt flags

    //Calls the SensingProgram for execution.
    SensingProgram();

    //Resets the RTC flag to 0
    intFlag &= ~(RTC_INT);

    //Clears the alarm flags in order to recapture the alarm
    RTC.clearAlarmFlag();

    //The alarm is set for another minute from this moment
    RTC.setAlarm2("06:12:01", RTC_OFFSET, RTC_ALM2_MODE4);
  }
}
```

3.4.1.2 Configuring the interrupt subroutine

The subroutine is the area of the program where the main node duties are performed such as sensing, sending, and saving of data. Algorithm 3.10 corresponds to the subroutine which is generated by the RTC interrupt; it consists of several commands and functions including the sensing program (only then the sensors are activate), interrupt and alarm flags (need to be cleared to enable the capture of the next interrupt), and finally we set the new time of the succeeding interrupt.

3.4.2 Event-Driven Execution

We propose an event driven management system for handling the physical processes of the main components (sensor boards, GPRS/GPS modules, etc.) in the node devices. The system we suggest attends the needs towards establishing regulatory schemes in order to conduct sensing and data transmission routines in WSNs. The main advantages of these regulations are:

* **Point-A:** Prevents the total collapse in the node operations.

* **Point-B:** Provides alarms when the remaining power has reached dangerous levels.

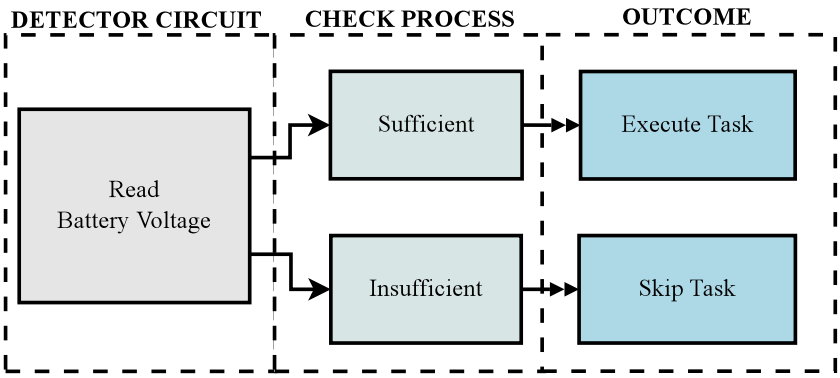


Figure 3.10: Voltage detection process.

Point-A refers to the methods in which we structure the main program in order to manage the execution of functions; these methods involve the utilization of a low battery interrupt which activates when the voltage across the battery detector circuit falls below the minimum voltage requirement specified in each component (each component has different current and voltage ratings). Consequently, we prevent a total collapse in the node by systematically disconnecting modules (components) when the battery can no longer provide them with sufficient energy to perform their tasks as shown in Figure 3.10.

	802.15.4 Routing Protocol				
Functions	*Sensors*	*GPRS*	*GPS*	*SD*	*XBee*
Last Execution Time	16:23	04:22	19:20	15:44	18:33
Last Execution Date	21.11.2011	16.10.2011	02.12.2011	19.05.2011	17.09.2011
Last Power Reading	4.1 V	3.9 V	3.6 V	3.2 V	3.45 V
Estimated Life Time (days)	40	38	27	3	10

Table 3.5: Live database format.

We realize the need for a reliable technique which notifies the WSN of a low battery event. The challenge we face does not concern the type of alarm we intend on using; whether it SMS, voice, e-mail is irrelevant. Our work focuses instead on the presumptions which suggest that node devices do not have sufficient energy to even generate the alarm. We propose that we integrate a live-database (on the internet) which includes the status of individual components as highlighted in Table 3.5 which includes last execution time/date, last power reading, and life time estimation.

Algorithm 3.11 Low battery interrupt configuration.

```
void Battery_Settings {

  // Configuring the minimum threshold for the battery in volts
  PWR.setLowBatteryThreshold(3.0);

  //Setting the pin for the battery interrupt
  enableInterrupts(BAT_INT);
}
void Battery_Settings {

  // Configuring the minimum threshold for the battery in volts
  PWR.setLowBatteryThreshold(3.0);

  //Setting the pin for the battery interrupt
  enableInterrupts(BAT_INT);
}
```

Finally, Algorithm 3.11 describes the software implementation for configuring the low battery interrupt service in the sensing functions. For example, we have configured a threshold of $3.00\,V$ which suggests that an interrupt will occur only after the voltage falls below the threshold limit. In reality, we dynamically configure the low battery interrupt for all components in order to handle their functional duties.

3.5 Sensors Management

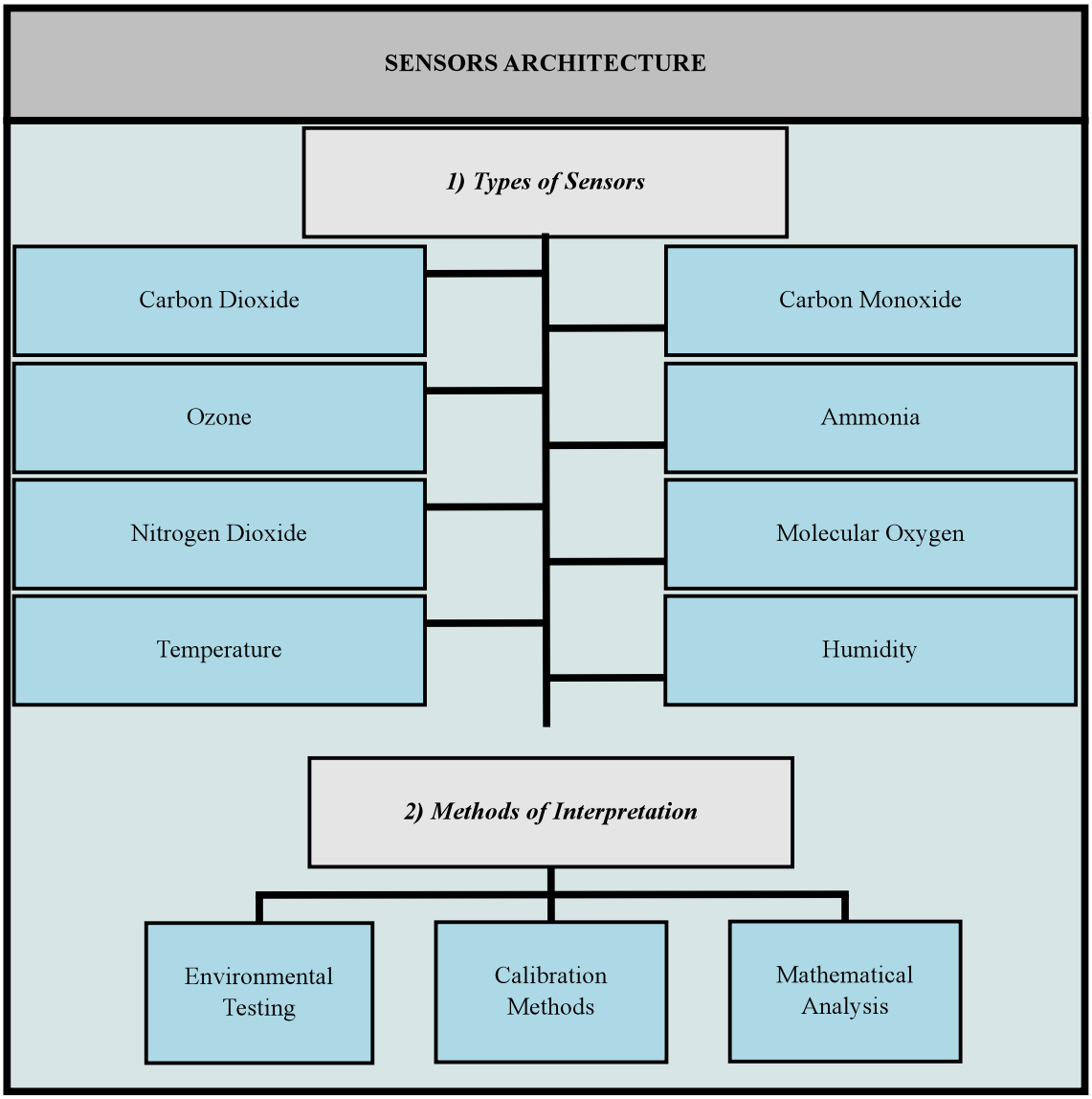


Figure 3.11: Sensors architecture.

In this section, we propose the sensing architecture shown in Figure 3.11, which focuses on the calibration and mathematical analysis of several gas sensors we use to measure the physical environment, including toxic sensors (e.g., nitrogen dioxide) and non-toxic sensors (e.g., temperature).

3.5.1 Calibration and Mathematical Analysis

Prior to deploying the sensors in the real world, we calibrate each one according to their general characteristics. This process involves three important factors:

* Normal testing conditions: These are specified by the manufacturer of each type of sensor which includes environmental conditions such as temperature, calibration gas(es) (other than the gas-sensor itself), minimum load resistances, and location (high/low altitude).

* Normalized plots: Which represent the mathematical relationships of the sensor that we use to compare to our results.

* Units of measurement: Each sensor is unique; some are based on resistance while others are read in voltage, therefore it is imperial to understand the units which apply to the sensors.

Summary of Parameters in Gases Board			
Sensor	*Sensor Resistance* Ω	*Equation(If applicable)*	*Nominal Voltage*
Temperature [43]	-	$Temp = 100 * (V_{OUT}) - 50$	-
Humidity [44]	-	$Humidity = (32 * V_{OUT}) - 26$	-
Molecular Oxygen [45]	-	$Oxygen = (5 * V_{OUT})/2$	0.6 V
Nitrogen Dioxide [46]	2.828 $k\Omega$	-	0.45 V
Air Contaminants-1 [47]	11 $k\Omega$	-	0.2 V
Air Contaminants-2 [48]	12 $k\Omega$	-	0.2 V
Ozone [49]	10 $k\Omega$	-	1.3 V

Table 3.6: Summary of gas parameters.

In Table 3.6, we provide a general summary regarding the calibration parameters that apply to the following sensors:

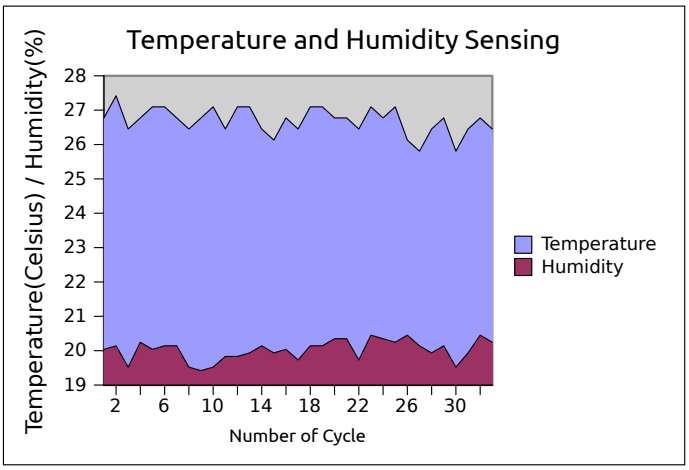


Figure 3.12: Temperature/Humidity sensor output response.

1) Temperature and Humidity: Both temperature and humidity sensors adhere to a linear relationships which are expressed in Table 3.6. Algorithm 3.12 outlines the configuration of the temperature and humidity sensors which converts the output voltage to degrees (C^o) and relative humidity (%) respectively (Figure 3.12).

Algorithm 3.12 Temperature/Humidity sensor configurations.

```
void get_Temp(){
  SensorGas.setBoardMode(SENS_ON);
  Temp = ((100*SensorGas.readValue(SENS_TEMPERATURE)) - 50); //Calibration.
  sprintf(Temperature,"Temperature=%d", Temp);
}

void get_Humidity(){
  SensorGas.setBoardMode(SENS_ON);
  Humd = ((32*(SensorGas.readValue(SENS_HUMIDITY))) -26); //Calibration.
  sprintf(Humdity,"Humidity=%d", Humd);
}
```

2) Molecular Oxygen: The oxygen sensor measures concentrations of molecular O_2 in air between $(0 \sim 30)\%$. Under normal conditions (i.e., breathable air), the O_2 concentration varies between $(21 \sim 30)\%$ (depending on the geographical location); this is equivalent to the sensor's output voltage 0.6 V, in which is noticeable at the maximum resolution maximum (i.e., signal amplification by 100). We start noticing a decrease in O_2 levels when the sensor's voltage begins to fall; this is expressed by the oxygen equation which is expressed in Table 3.6.

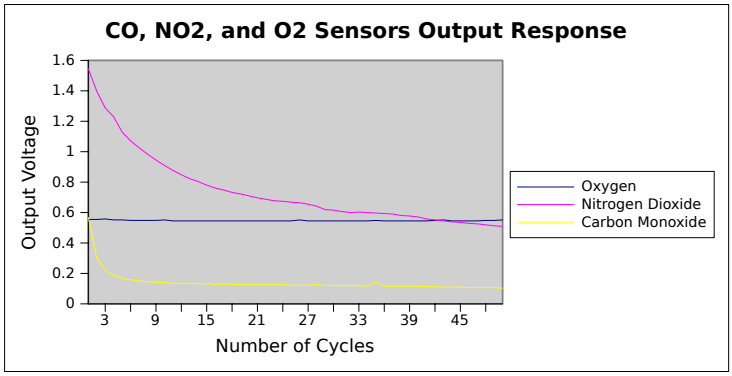


Figure 3.13: Sensors output response.

Algorithm 3.13 Oxygen sensor configuration.

```
void get_Oxygen(){
  SensorGas.setBoardMode(SENS_ON);
  SensorGas.configureSensor(SENS_O2, 100); //Maximum sensor resolution
  SensorGas.setSensorMode(SENS_ON, SENS_O2);
  O2 = SensorGas.readValue(SENS_O2); //should read 0.6V.
  sprintf(Oxygen,"Oxygen=%d", O2);
}
```

In Algorithm 3.13, we demonstrate the oxygen sensor's configuration at maximum resolution, which implies amplifying the output signal of the sensor (i.e., gain) by 100 times.

3) Nitrogen Dioxide:

This sensor is sensitive to the presence of NO_2 concentrations in air between $(0.05 \sim 5)$ *ppm*.

Algorithm 3.14 Nitrogen Dioxide sensor configuration.

```
void get_Nitrogen_Dioxide(){
  SensorGas.setBoardMode(SENS_ON);
  SensorGas.configureSensor(SENS_SOCKET2B,1,2.828);
  SensorGas.setSensorMode(SENS_ON, SENS_SOCKET2B);
  NO2 = SensorGas.readValue(SENS_SOCKET2B);
  sprintf(NitrogenDioxide,"Nitrogen_Dioxide=%d", NO2);
}
```

Under normal conditions, sensor resistance (R_S) is approximately 2.2 $k\Omega$ which can be calculated from Equation 3.1.

$$R_S = \left(\frac{V_{INPUT} * R_{LOAD}}{V_{OUT}}\right) - R_{LOAD} \tag{3.1}$$

Our tests conclude the following results:

- Output voltage (V_{OUT}) fluctuates between (0.4 $\sim$ 0.5) V using a load resistance (R_L) of 2.2 $k\Omega$.
- In theory, our results yield to sensor resistance (R_S) of approximately 2.828 $k\Omega$; thus agrees with the normal testing conditions. Furthermore, this configuration is applied in Algorithm 3.14 which outlines the setup of the NO_2 sensor in the surrounding environment.

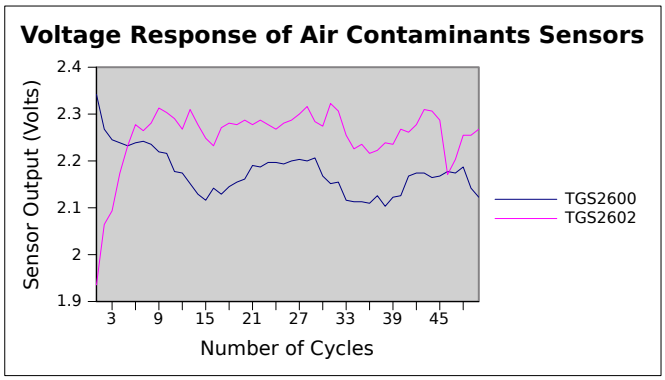


Figure 3.14: Air contaminants sensors voltage response.

4) Air Contaminants and Ozone: There two types of air contaminants sensors highlighted in Table 3.6 which are sensitive to presence of methane, carbon dioxide, iso-butane, ethanol, hydrogen, ammonia, hydrogen sulphide, and toluene in air. Under normal air concentrations, sensor resistance (R_S) is approximately (10 $\sim$ 100) $k\Omega$. Figure 3.14 represents the output response for both sensors while they are subjected to normal air.

The next sensor involves the O_3 (ozone) gas in which is sensitive to concentrations in air between (10 1000) ppm. Air resistance (R_S) under normal testing conditions of this sensor varies between (3 $\sim$ 60) $k\Omega$. During the calibration of this sensor; we made the following observations:

Algorithm 3.15 Air contaminants sensor configuration.

```
void get_Air_Pollution(){
  SensorGas.setBoardMode(SENS_ON);
  SensorGas.configureSensor(SENS_SOCKET2A,1,11);
  SensorGas.configureSensor(SENS_SOCKET4A,1,12);
  SensorGas.setSensorMode(SENS_ON, SENS_SOCKET2A);
  SensorGas.setSensorMode(SENS_ON, SENS_SOCKET4A);
  TGS2600 = SensorGas.readValue(SENS_SOCKET2A) ;
  TGS2602 = SensorGas.readValue(SENS_SOCKET4A) ;
  sprintf(AirPollution,"TGS2600=%d|TGS2602=%d", TGS2600, TGS2602);
}
```

- Output voltage (V_{OUT}) reached stability at approximately 1.30 V in which took over 70 power cycles sampled every 30 seconds.
- We use a load resistance (R_L) equivalent to 11 $k\Omega$, which yields to air resistance of 10 $k\Omega$. Thus agreeing with the theoretical observations of the O_3 sensor.

Algorithm 3.16 Ozone sensor configuration.

```
void get_Ozone(){
  SensorGas.setBoardMode(SENS_ON);
  SensorGas.configureSensor(SENS_SOCKET2B,1,10);
  SensorGas.setSensorMode(SENS_ON, SENS_SOCKET2B);
  O3 = SensorGas.readValue(SENS_SOCKET2B) ;
  sprintf(Ozone,"Ozone=%d", O3);
}
```

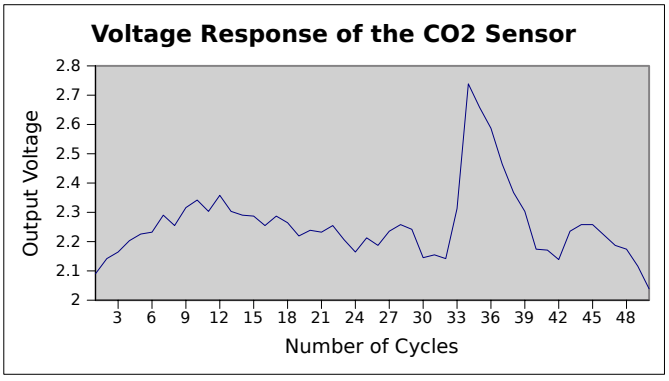


Figure 3.15: Carbon Dioxide sensor voltage response.

5) Carbon Dioxide: The carbon dioxide (CO_2) sensor is an expectational case in which detection of CO_2 is based on changes in the electromotive forces (ΔV) of its output voltage. We know that a well-ventilated space (outdoor) has approximately 350 ppm of CO_2 in air, which is equivalent to $\Delta V = (220 \sim 490)\ mV$ according to specifications in [50]. The method in which we carried the calibration of this sensor involves amplifying the output by the highest possible gain (of 3.5 times) which yields to a voltage of approximately 3.3 V (maximum). This implies the new fixed

reference voltage which decreases by rising concentrations of CO_2in air. In Figure 3.15, we plot the variations in ΔV in air using a gain of 3.5 in order to achieve the highest possible resolution without saturating the sensor.

3.5.2 Log-Log Plots

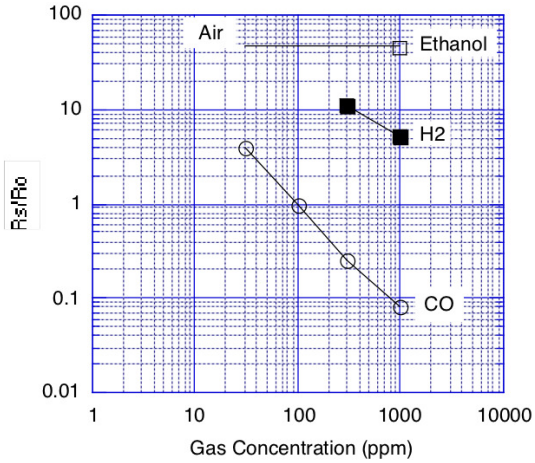


Figure 3.16: Log-Log plot of carbon monoxide sensor.

The log-log graphs are commonly used to express the mathematical relationships in the gas sensors, which is usually related to gas concentrations with respect to output voltage or resistance gain (R_s/R_o). For example, Figure 3.16 represents the log-log plot of the carbon monoxide sensor. We apply mathematical reasoning in expressing the relationship of the CO concentrations based on the following method:

1) Defining the variables: To calculate the relationship of a linear equation we need two points on the line. Similarly, when we are dealing with log-log plots; the same methods are adopted, therefore we pick any two points from Figure 3.16:

$$P_1 = (100, 1) \quad P_2 = (30, 4)$$

2) Calculating the gradient: Similar to linear equations except the gradient is calculated in the logarithmic scale as such:

$$m = \left(\frac{log(Y_2) - log(Y_1)}{log(X_2) - log(X_1)}\right) = \left(\frac{log(4) - log(1)}{log(30) - log(100)}\right) = -1.1514$$

3) Estimating y-intercept: Again very similar method to linear equations except we use the logarithmic scale:

$$b = y - mx = log(1) - (-1.1514 * log(100)) = 2.3028$$

4) Expressing the formula: A log-log plot adheres with the following expression:

$$y = x^m * 10^b$$

$$R_S/R_O = x^{-1.1514} * 10^{2.3028}$$

5) Apply equation: Note the output voltage of carbon monoxide sensor from Figure 3.13 which yields to the following observations:

- Output voltage of the CO sensor initially is $V_{OUT-CO} = 0.6\ V$
- We calculate the sensor resistance from Equation 3.1 which yields to $R_S = 73.3\ k\Omega$ using $R_L = 10\ k\Omega$ and $V_{PP} = 5\ V$.
- We calculate the resistive gain (R_S/R_O) from $R_S = 73.3\ k\Omega$ and $R_O = 13.3\ k\Omega$; therefore $R_S/R_O = 5.5$.

Therefore the concentration of carbon monoxide at $R_S/R_O = 5.5$ is $Concentration_{(CO)} = 22.75\ ppm$.

3.6 Alarm System Architecture

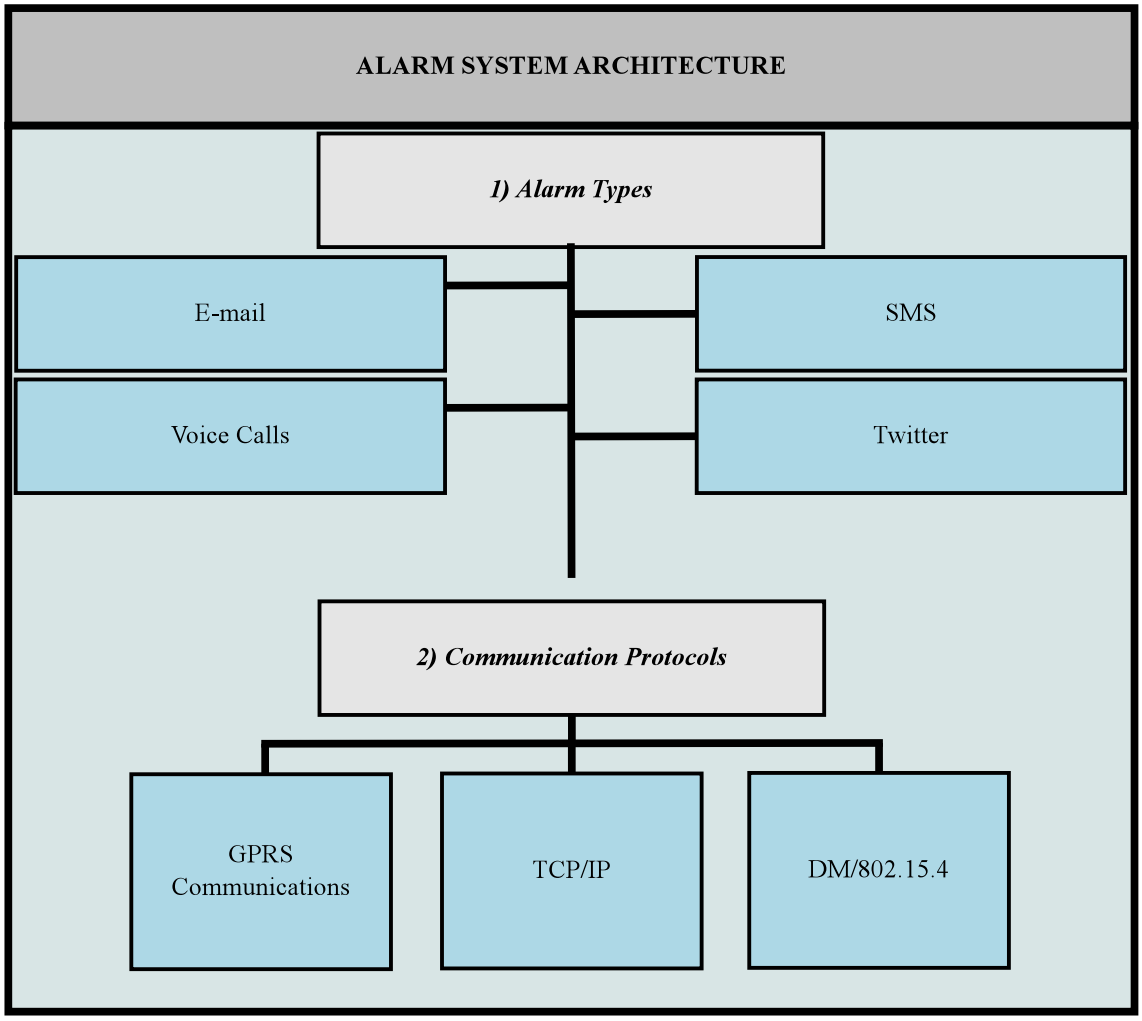


Figure 3.17: Alarm system architecture.

In this section, we propose the alarm system architecture shown in Figure 3.17, which specifies four types of alarms including e-mail, SMS, voice-call, and twitter. The first part of this discussion focuses on sensors; specifically, their formulas and threshold limits. And later, we apply the system in real-life in order to measure its functional performance. We aim to use three types of technologies in order to generate the alarm signals which includes: a) GPRS communications, b) TCP/IP, and c) DM/802.15.4 protocols.

3.6.1 Sensors Threshold Configuration

We learnt from Section 3.5 about the different types of sensors including their calibration methods and mathematical analysis. Now, we present their applications specifically in the lead up to the alarm signal that is generated by the node device when sensor values exceed their limits. But first, we look at the main formulas which relate to the concentrations of different gases already discussed in the previous section.

Concentration Formulas of Gas Sensors	
Gas Type	R_S/R_O
Nitrogen Dioxide (NO_2)	$x^{1.756} * 10^{2.698}$
Carbon Monoxide (CO) [51]	$x^{-1.151} * 10^{2.302}$
Ammonia (NH_3) [52]	$x^{-0.602} * 10^{-0.096}$
Methane (CH_4) [53]	$x^{-0.431} * 10^{1.593}$
Ozone (O_3)	$x^{0.866} * 10^{-1.722}$

Table 3.7: Gas sensors concentration formulas.

Algorithm 3.17 GPRS module configuration.

```
void initiate_GPRS(){
  GPRS.ON(); //Powers the GPRS module.
  while(!GPRS.check()); //Connects to the GSM network.
  GPRS.setTextModeSMS(); //Sets the text mode in the GPRS module.
}

void Emergency_Alarm(){
  if (CO > threshold_CO){
  if(GPRS.sendSMS("Waspmote alarm. CO levels high","Mobile")){
    check_flag=1;}
  GPRS.OFF(); //Switch off the module
}

void Check_GPRS(){
  if(check_flag != 1){
    SendUsingXBee();
  }
}
```

In Table 3.7, we summarize the main formulas of different gas sensors that are based on logarithmic equation format. In a real-life deployment, we configure each node device to generate an alarm the moment a threshold limit is exceeded. On-board each node, a GPRS module is fitted which provides the alarm signals through SMS notifications or a voice call made to a designated mobile recipient(s). We take into considerations the energy requirements from the GPRS unit, therefore our aim is to minimize the number of false alarms that are initiated. There are three important functions in Algorithm 3.17 which include:

* Initiating of the GPRS module: this process should only take place when the GPRS module is needed; otherwise it should always be switched off.

* Configuring the service: Which implies the type of function we require of the GPRS module (i.e., SMS, voice-call, upload to FTP).

* If neccessary a backup option should be provided incase the GPRS module fails to connect to the GSM network.

The second type of the alarm system involves TCP/IP communications, in which we configure

meshlium as an IP gateway in order to provide e-mail and twitter alarm messages. This is illustrated in Example 3.7.

Example 3.7. E–mail and Twitter integration:

```
void Send_E-mail(){
  if (error == 0) {
    int pid;  //init child process
    pid = fork();
      if (pid == 0) {
        FILE *file;
        file = fopen("/mnt/user/zigbeeStorer/message.txt", "w");
        if(file==NULL) {
          fprintf(stderr,"Error: can't create file.\n");
        }
        else {
          fprintf(file,"SENSOR VALUES");
          printf("Message file created. \nSending mail alert.\n");
          fclose(file);
          system("ssmtp <e-mail_address> < /.../message.txt");
        }
      }
  }
}

void Send_To_Twitter() {
  char order[255];
  char *user = "twitter_user";
  char *pass = "twitter_pass";
  sprintf(order, "twurl /1/statuses/update.xml -s -u -d %s:%s
          status=\"Mote:%d Detected value %d\"", user, pass,
          mote_id, sensor_value);
  system(order);
  }
```

Conclusion

In this chapter, we proposed an application framework for WSNs which specifies: a) communication networks and protocols, b) Over the air programming capabilities, c) management of power resources, d) sensor management, and d) alarm generation system. We addressed a number of concerns which are related to off-the-shelve WSN hardware including:

- Network formation: First, we specified the main communication protocols first, then we outlined their specific purpose with general WSN hardware, and finally we discussed the main functionalities towards the use of personal area networks identifiers (PAN-IDs), data encryption, and node identifiers (NIs).
- Network topology: We implemented a number of network topologies which are relevant to WSNs including mesh, p2p, and ring.
- Over the air programming: We developed OTA applications which are associated with two types of networks; first is the OTA over remote links in which the application operates over TCP/IP networks in order to perform the OTA functions, and second application is designed for local WSNs where they can be accessed from within the same infrastructure (i.e., person on-site).
- Optimized detecting rates and event-driven execution: Which are specified by our power management schemes that aim at lowering the power consumptions in the WSN devices.
- Calibration and mathematical analysis of gas sensors: We present the audience with the methods in which we calibrated the gas sensors. We also provided detailed analysis on logarithmic equations, how to interpret them, and then apply them to the sensors output data to obtain their concentrations in air.
- Alarm system technologies: Which include GPRS communications to provide SMS and voice call alarms, also we showed the methods for integrating WSN data into live services including Twitter, MYSQL databases, and FTP hosts.

Chapter 4

ENVIRONMENTAL MONITORING SYSTEMS

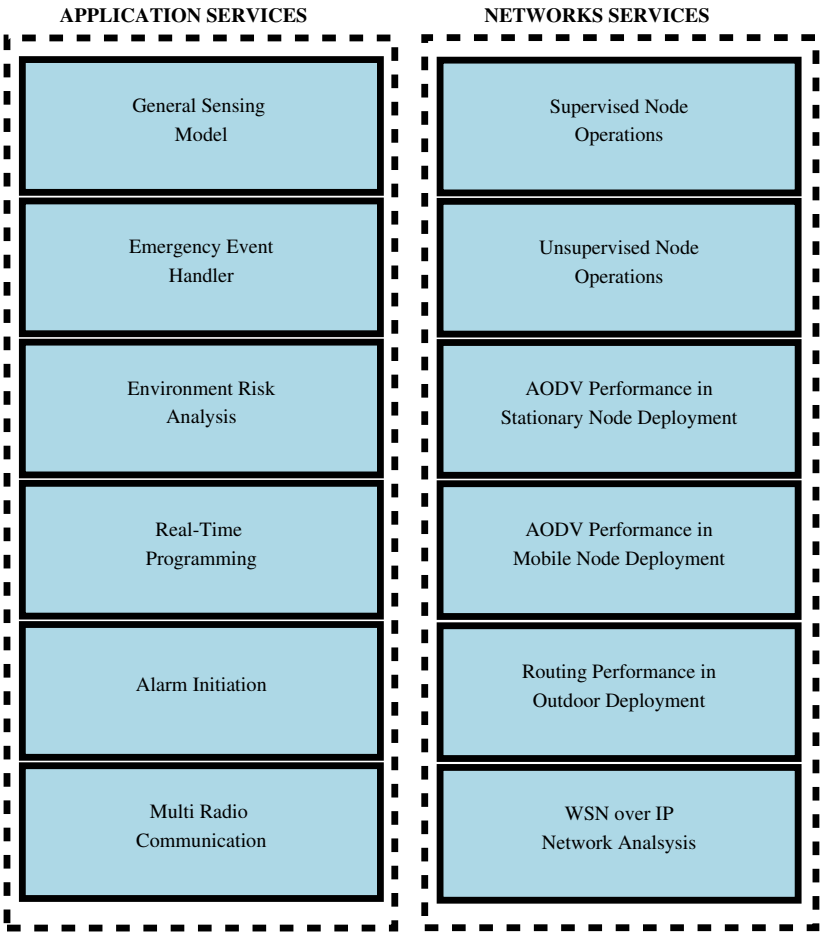


Figure 4.1: Application functionality model.

In this chapter, we develop environmental applications which focus on health monitoring, air quality safety, and fire detection WSNs. Additionally, we propose a system which integrates WSNs and home network infrastructure in order to provide assistance with ADL (Assistance Daily Living)

activities to home patients. We also address the technical configurations of hardware such as the routing requirements in outdoor/indoor network deployment, multihop data transmission, and optimization of power resources in the node devices. Finally, we conduct benchmark examination which involves real-time network utilization, power consumption analysis, and packet delivery success rates in real-world scenarios. In Figure 4.1, we differentiate the contrasting functionalities between: a) application based services, and b) networks related functions; in order to assist the audience with understanding the exact nature of our investigation.

4.1 Application Services

This section describes three important services that form the monitoring body (structure) of WSNs. The monitoring body implies the functional medium where the base operations (of the application) originate; here, we specify the general sensing model, feedback functions, real-time event management, and environmental risk analysis.

4.1.1 Sensing Model

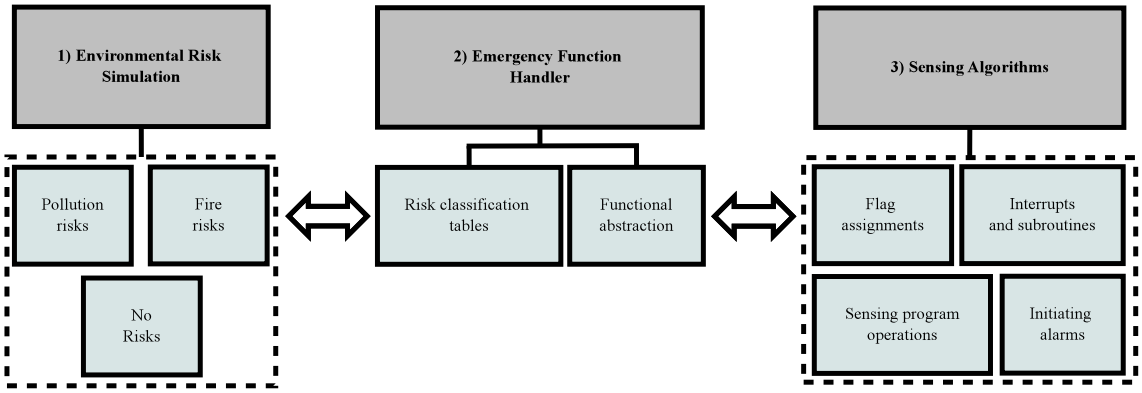


Figure 4.2: General sensing model.

We propose the general sensing model shown in Figure 4.2, which consists of three main parts:

* Environmental risk simulation: The risks are influenced by environmental parameters such as temperature in surrounding space and gas concentrations in air.

* Emergency function handlers: These functions are classified based on the risk involved; we specify three types of risks which include pollution, fire, and no risk. Feedback functions handle each risk seperately (i.e., depending on the type of event).

* Sensing algorithms: They provide the functional abstraction of the entire sensing model including real-time sensing programs, event based interrupts management (involves alarm flags), and handling the alarm generation process.

Risk Classification	
Type	*Description*
High risk	A high risk indicates a high probability of fire and pollution in the surrounding environment.
Medium risk	A medium risk indicates the presence of abnormal gases in the surrounding environment; however, these levels are not considered lethal.
No risk	A no risk inidicates normal conditions which does not require any special attention.

Table 4.1: Environmental risks classification.

4.1.1.1 Environmental risk simulation

In Table 4.1, we specify three types of environmental risks which include high, medium, and low. In the real world, each node device generates a unique response that corresponds with the type of risk present in the environment; high risks are prioritized over medium and low risks, and so on. In summary, the main aim of environmental risk simulation is twofold:

1. To better understand the environment we are dealing with.
2. To prepare us deal with environmental encounters more efficiently; thus improving the reliability of the WSN system.

4.1.1.2 Event specifications

3-bit Alarm Data Structure			
Methane-bit	*Temperature-bit*	*Oxygen-bit*	*Risk Classification*
0	0	0	No risk
0	0	1	Medium risk
0	1	0	Medium risk
0	1	1	High risk
1	0	0	Medium risk
1	0	1	High risk
1	1	0	High risk
1	1	1	High risk

Table 4.2: 3-bit alarm flag structure.

In Table 4.2, we construct a 3-bit flag structure (lookup table) which constitutes of methane, temperature, and oxygen bits. Here, we assume the environmental conditions are affected by the presence of CH_4 and O_2 gas, also air temperature in the surrounding domain; therefore we base the risk model only on these three parameters. Each bit (flag) is triggered by the node device when the sensors exceed or recede from a specific threshold level. E.g., oxygen levels reduce during fire (due to combustion); therefore, the oxygen-bit triggers when oxygen concentrations falls under

17% (under normal conditions 21%) in the air. A methane leak indicates a pollution risk, therefore the methane-bit is triggered. Similarly, rising temperatures could also signal a fire, especially when a methane leak also occurs at the same time; therefore, we trigger the temperature sensor after it exceeds a certain threshold. Feedback functions are unique in a sense that each type of event involves different techniques and handling methods; e.g.:

- High risk: Regarded as the most important event, therefore our priority to treat it as such; in this case, the best option is to use the GPRS module to initiate a voice-call and wait for the call to be attended to. If the latter alarm fails, we transmit an SMS message or upload an urgent twitter notification to the web. Finally, if all options fail then we activate the radio-unit (XBee) and transmit the alarm signal; an acknowledgment packet will be re-transmitted by the base-station to signal the alarm has been received.

- Medium risk: This event requires its handler function to transmit an alarm using the XBee radio unit and wait for an acknowledgment. The exclusion of GPRS functions is due to the power consumption demands by this module.

- No risk: This event does not require any special alarm, instead its handler function is required to treat the data by storing it in μSD memory.

4.1.1.3 Sensing Algorithm

The sensing algorithm is a software representation of the risk model and emergency handler functions which we discussed previously. The algorithm consists of three parts which include: a) mapping the alarm flag, b) configuring the sensor program, and c) treating the alarm signals.

1) Mapping the alarm flag: This refers to the software representation of the 3-bit data structure from Table 4.2. This process involves two steps as highlighted in Algorithm 4.1: a) initializing the flag data structure, and b) defining the bit-flag of each variable (e.g., METHANE, OXYGEN, and TEMP).

Algorithm 4.1 Mapping the alarm flag.

```
//Alarm flag declaration
uint8_t flagAlarm=0;  //init alarm flag structure
#define TEMP 1      //(0,0,1)
#define OXYGEN 2    //(0,1,0)
#define METHANE 4   //(1,0,0)
//Sensors thresholds
#define TEMPLEVEL 40      //Celsius
#define OXYGENLEVEL 21    //Percentage % in air
#define METHANELEVEL 17   //Percentage % in air
```

2) Configuring the sensor program: This involves: a) reading the sensors output voltages (Appendix B.3), b) configuring thresholds as illustrated in Algorithm 4.1.

3) Bit-flag triggers: This refers to triggering the flag-bits of the alarm structure from Table 4.2, which also aligns with the environmental risk model we previously analyzed. Algorithm 4.2 analyzes the triggering process of the methane, oxygen, and temperature sensors.

Algorithm 4.2 Bit triggering.

```
void check_flags(){
//Sets the bits ON or OFF.
  if(CH4 > METHANELEVEL ) flagAlarm |= METHANE; //if ON; flagAlarm=4
  if(O2 > OXYGENLEVEL) flagAlarm |= OXYGEN; //if ON; flagAlarm=2
  if(Temp > TEMPLEVEL) flagAlarm |= TEMP; //if ON; flagAlarm=1
}
```

4) Event handler classification: This process involves three functions highlighted in Algorithm 4.3 which include: a) constructing the event flag and b) determining the risk level based on the alarm flag.

Algorithm 4.3 Event handler classification.

```
//Event flag declaration
uint8_t flagEvent=0; //init event flag structure
#define HIGH_RISK 0
#define MEDIUM_RISK 1
#define NO_RISK 2
#define ERROR 3

void event_handler(){
  switch(flagAlarm) //Risk classification table
    {
      case 0:
        flagEvent |= NO_RISK; //sets no risk flag
        break;
      case 1: case:2 case 4:
        flagEvent |= MEDIUM_RISK; //sets medium risk flag
        break;
      case 3: case 5: case 6: case 7:
        flagEvent |= HIGH_RISK; //sets high risk flag
        break;
      default:
        flagEvent |= ERROR;
        break;
    }
}
```

5) Event handler functions: This process involves four main functions which include: a) low risk function, b) medium risk function, c) high risk function, and d) error function. Algorithm 4.4 specifies the method in which the event flag is checked against the type of risk determined by the alarm flag.

Algorithm 4.4 Event handler functions.

```
void event_functions(){
  switch(flagEvent) //Risk classification table
    {
      case 0: //(i.e, no risk)
        LowRisk(); //executes low risk functions.
        break;
      case 1: //(i.e, medium risk)
        MediumRisk(); //executes medium risk functions.
        break;
      case 2: //(i.e, high risk)
        HighRisk(); //executes high risk functions.
        break;
      case 3: //(i.e, error)
        ErrorFunction(); //execute the error functions.
      default:
        ErrorFunction(); //executes error functions.
        break;
    }
}
```

4.1.1.4 Alarm functions

As we have already mentioned, there are three cases which involves the types of alarm signals which are generated by the event handler; high, medium, and low risk alarms.

1) Low risk alarm: This alarm involves storing the sensor data in the μSD memory card on-board the node device, which is outlined in Algorithm 4.5. In low risk events, the node device saves its energy by switching off the main communication modules (GPRS, XBee, GPS); the data get retained in the memory card instead of wasting energy on their transmission.

Algorithm 4.5 Writing to SD memory.

```
char *CH4; char *O2; char *Temp; char data[100];//declare variables

void Write_To_SD() {
  sprintf(data,"CH4:%s|O2:%s|Temp:%s\r\n", CH4, O2, Temp);
  SD.ON();  //Powers the SD module
  SD.mkdir("Sensors");   //Creates folder "Sensors"
  SD.cd("Sensors");   //Go to folder Sensors
  SD.create("Gas_Sensors.txt");       //Creates the file
  SD.append("Gas_Sensors.txt", data);    //Writes data into file
}
```

2) Medium risk alarm: The transmission of this type of alarm is performed using the XBee radio module; the node device transmits the signal to a special gateway which responds (i.e., the gateway) by acknowledging the capture of the alarm. Therefore, this event handler function includes a special feature which locks the node in a retransmit-mode ($20s$ intervals) until the final acknowledgment is heard by the device.

3) High risk alarm: The transmission of this alarm is based on GPRS (primary) and XBee (secondary) communication. The primary alarms include initiating voice calls and SMS, also through twitter's live feed capabilities, and finally by e-mail communication. However, if our attempts to connect the node's GPRS module with an appropriate GSM network fail to proceed, then we utilize the back-up option which is through the XBee module (follows a similar procedure to medium alarm signaling).

4.1.2 Real-Time Processing

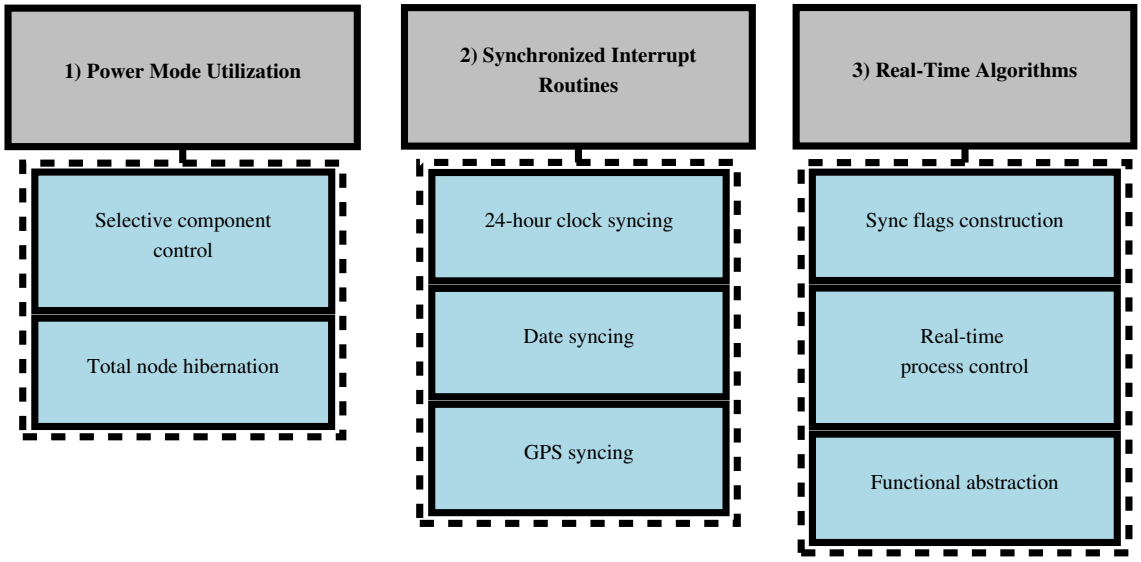


Figure 4.3: Real-time process model.

We propose the real-time processing model shown in Figure 4.3, which involves three main functions including:

* Power modes utilization: Which consist of three techniques including hibernation, deep-sleep, and regular sleep. Once we set the node into hibernation, its entire architecture including the components are completely disconnected from the power source; this method preseves most of the node's energy, therefore it is has the lowest power consumption ratio. In deep-sleep and regular-sleep modes, we retain a level of control over the node components, which enable us to selectively pick which components are set to be disconnected. These modes are completely dependant on the Real-Time-Clock (RTC) for managing their sleep and wake-up times.

* synchronised interruption routines: We demonstrate to the audience our method which utilizes the RTC in order to develop highly flexible and dynamic applications for WSNs. The main attribute of the RTC is its usefulness in real-time processing; it enables WSNs to synchronize their tasks according to time requirements and date constraints.

* Real-time algorithms: These algorithms provide the functional abstraction that defines the application structure of the RTC, and highlight its interactions with real-time events. We implement two types of algorithms based on the RTC: a) management algorithms which handle the power modes, and b) synchronizing algorithms which control the detection frequency in the node device.

4.1.2.1 Power modes

Internal Power Control Switches	
Type	*Description*
Sensor	This component controls the I/O switches on the sensor boards.
UART0	This data bus provides communications with the XBee radio modules.
UART1	This data bus provides communications with the GPRS and GPS modules.
Battery	This component controls the switch to the main battery.
RTC	This component controls the switch to the RTC module.

Table 4.3: Node components classification.

Node devices are classified by two major functions which control the power components (switches) inside the internal hardware, they include:

1) Selective component control: Which applies to the list of components from Table 4.3; In this function, we manually select the components which shutdown during the sleep period of the node device. The RTC retains its role of synchronizing the wake-up times in devices which are in a deep-sleep state (also regular sleep).

2) Total node shutdown: This function utilizes the hibernation mode to completely disconnect each component from the power supply (i.e., zero consumption in power). In this mode, the RTC maintains its syncing operations using an auxiliary battery as the main source of power.

4.1.2.2 Interrupts and subroutines

Interrupts are activated by the RTC module to signal an event (process), which consists of three types:

- Sensor event: Also referred to as the detection period, which is specified by the RTC. When this interrupt is active, node devices switch-on their sensor boards and take measurements from the surrounding environment.
- Sleep event: Which refers to the duration in which nodes are asleep, deep-sleep, or in hibernation. During these states, node components may still operate while others are completely switched off; mainly to preserve the energy in the battery source.
- Wake-up event: Which reactivates the node devices after their sleep periods lapse.

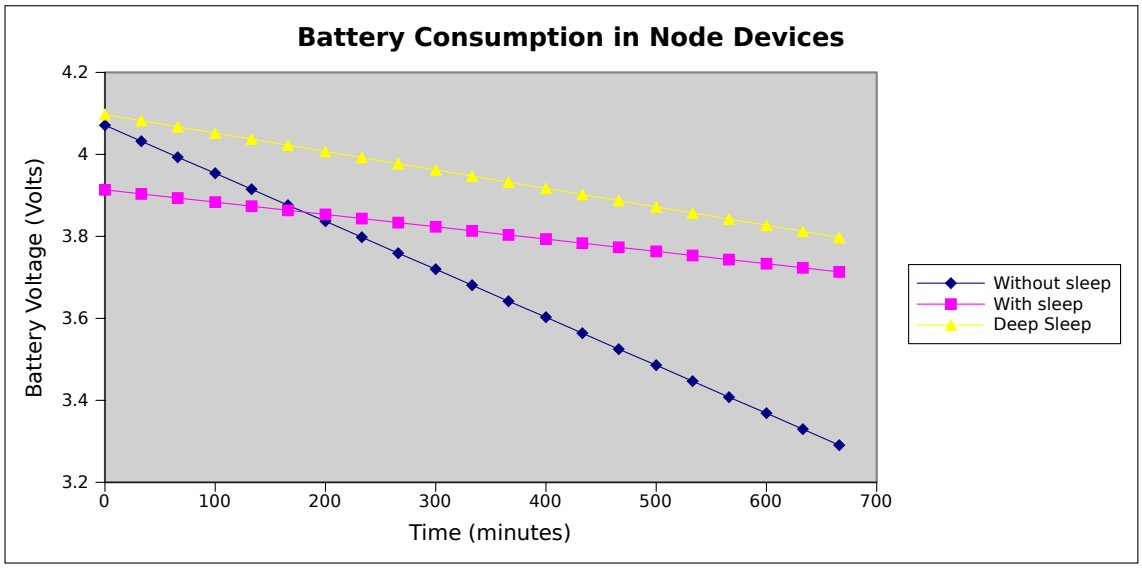


Figure 4.4: Power modes consumption analysis.

The RTC requires time syncing prior to its utilization (i.e., for managing events). We setup the RTC to sync its time with a GPS receiver, which is based on the NMEA standard; National Marine Electronics Association (NMEA) is a communication protocol used to specify the electrical requirements in GPS navigational equipment. The are two main techniques for syncing the RTC which include:

1) User-defined: This method of syncing is specified during the setup stage (programming) of the node device which is highlighted in Algorithm 3.10.

2) GPS sync: This method provides time in UTC (Universal Time Coordinated) format, then syncs the RTC as Algorithm 4.6 specifies; the synchronised data are based on the following parameters: hours, minutes, seconds, and the date.

4.1.2.3 Real-time algorithms

The real-time algorithm specifies four individual functions which activate during specific seasons (i.e., summer, autumn, winter, spring). In some applications (e.g., fire monitoring), the need for continuous monitoring is scaled down in winter, especially in outdoor areas where the sun is not

Algorithm 4.6 Syncing RTC with GPS.

```
void Start_GPS_Module(){
  GPS.ON(); //Powers the GPS module.
  while( !GPS.check() ) // waiting for satellite coverage
}
void Sync_to_RTC(){
  GPS.getPosition(); //gets GPS coordinates
  RTC.setTimeFromGPS(); //syncs to RTC
}
```

as powerful (compared to summer); as a result, hibernation mode is utilized (see Figure 4.4). The structure of real-time algorithms are as follow:

1) Event flag: This is a 4-bit flag which refers to the weather seasons as highlighted in Algorithm 4.7.

Algorithm 4.7 Event flag structure.

```
//Event flag declaration
uint8_t eventFlag=0;
#define January 1
#define February 2
#define March 3
#define April 4
#define May 5
#define June 6
#define July 7
#define August 8
#define September 9
#define October 10
#define November 11
#define December 12
```

2) Flag management: The RTC is synchronised by the GPS module as highlighted in Algorithm 4.6, then its output determines the type of operation which is required in a specific month/season. Algorithm 4.8 involves four main functions which include: a) summer, b) autumn, c) winter, and d) spring; and as we mentioned earlier, each program runs during different seasons.

3) Hibernation program: In this program, the node is put into hibernation during the winter season to preserve its energy levels; the node hibernates for nearly 3 months, however it still wakes up once a day to take measurements using its sensors, then transmits the data to a base-station. Algorithm 4.9 outlines the use of hibernation in the winter program, in which data consist of time/date, temperature, and battery measurements.

Algorithm 4.8 Flag management model.

```
void flag_mngt()  {
  switch(RTC.month)     {
    case October: case November: case December:
      SummerProgram(); //run the summer program
      break;
    case January: case February: case March:
      AutumnProgram(); //run the autumn program
      break;
    case April: case May: case June:
      WinterProgram(); //run the winter program
      break;
    case July: case August: case September:
      SpringProgram(); //run the spring program
      break;
    default:
      ErrorProgram(); //run the error program
      break;
  }
}
```

Algorithm 4.9 Hibernation mode configuration.

```
void setup(){
  PWR.ifHibernate(); //sets the hibernation flag.
}

void Hibernation_Program(){
  delay(8000); //waits for hibernation process to finish.
  if( intFlag & HIB_INT){ //checks the hib flag
    intFlag &= ~(HIB_INT);
    SensorGas.setBoardMode(SENS_ON);delay(1000);
    Temp = ((100*SensorGas.readValue(SENS_TEMPERATURE)) - 50) ;
    Utils.float2String(Temp, TempS, 2) ;
    Utils.float2String(PWR.getBatteryVolts(), Batt, 6);
```

4.1.3 Data Encryption

WSNs are exposed to many types of hijack attempts for compromising the network security. Advanced Encryption Standard (AES) is a security technique which is used to encrypt the data payload prior to transmission in a sensor network. There are two key components in the setup of a secured sensor network which include:

* Mutual link access key: Which is a 16 bytes field shared among all node devices within a sensor network; In Figure 4.5a, a compromising node is stationed to intercept the data from unsecured WSN, where as in Figure 4.5b the same node intercepts the data without being able to decode the message since we have encoded it with 128b encryption key.

* Application access key: In over the air programming (OTAP), we have implemented two types of access keys which include a) Network layer, and b) Application layer. The latter conforms with the application access code which allows access to the OTAP upgrade, while

P.nbr.	Time (ms)	Length	Payload	RSSI (dBm)	LQI	FCS
134	+7 =28108	73	*c*4****@i******@i**/**********@i** ****@i****R*#*****@i**NODE-A Access	-44	255	OK

(a) Unencrypted transmission.

P.nbr.	Time (ms)	Length	Payload	RSSI (dBm)	LQI	FCS
152	+5 =35030	79	*j*4****@i******@i**/**********@i***** *@i****9***A!*****;9W****P*********<m*	-44	255	OK

(b) Encrypted transmission.

Figure 4.5: Payload encryption in sensor network.

the network layer security refers to the AES encryption we explained previously.

4.2 Environmental Sensor Networks

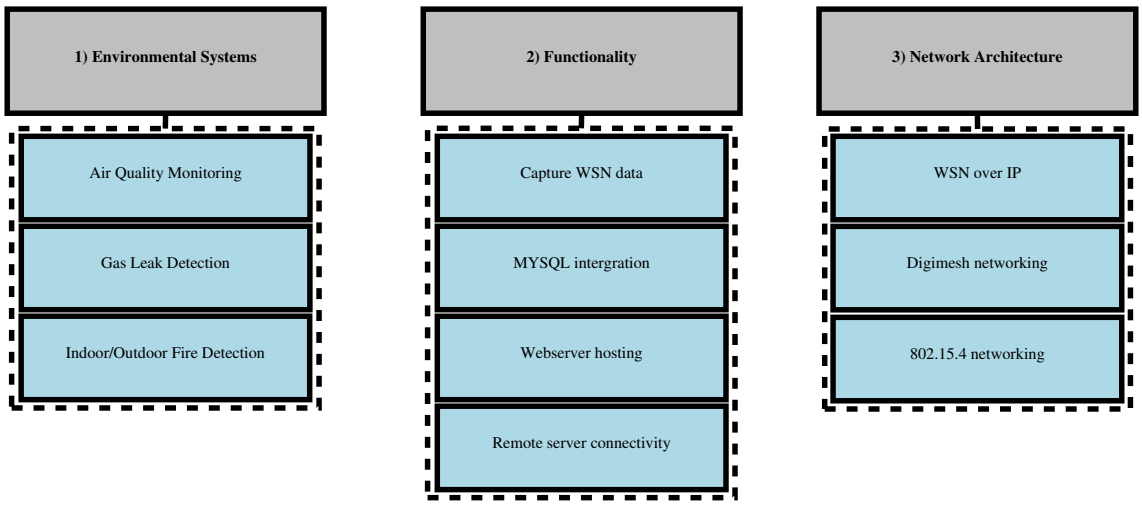


Figure 4.6: Environmental monitoring applications.

In this section, we propose the main environmental systems shown in Figure 4.6, which consist of three applications:

* Air-Quality monitoring: This application is intended for use in nursing-homes and retirement villages in order to monitor the air quality in the surrounding environment; It detects many types of air contaminants including NO_2, CO_2, NH_3, and H_2, then reports their concentrations to a base-station where they are processed into databases. Node devices utilize GPRS modules which generate SMS/voice-call signals to warn the residents of potential case (contamination,pollution), then contact the authorities whom initiate the evacuation process.

* Gas-Leak detection: This monitoring system is used to detect gas-leaks in pipelines that span across neighbourhoods into residential households or commercial buildings. This system is based on multihop transmission to ensure this project covers larger areas (no wired communication) without deterioration in the communications (signal loss, SNR); rather it relies on intermediate nodes to forward the data to the basestation points (quality of signal remains unchanged).

* Fire detection: This system is intended for use in indoor and outdoor environments in order to detect sparodic fire occurrences and reports them to suitable authorities (fire department, FESA, post them up on community websites, etc.). As the case with the air-pollution system, node devices utilize GPRS modules which connect to the Internet and posts real-time data on desginated servers that continuously track the progress of the system.

The three systems use multi-radio communications architecture for enabling wireless connectivity between the hardware devices which include the sensing platforms, routing nodes, and base-stations; these communications are based on IP addressing and data-forwarding techniques in order to initiate long-range links between numerous and independent WSNs. This type of networking is

also referred to as "hybrid communications" as it consists of numerous other protocols including IEEE-802.15.4, Digimesh, GPRS, and IP. In this section, we outline the functional interactions between these different communication protocols, then we concentrate on their utilization in WSN applications.

4.2.1 Communication Performance Analysis

First, we investigate the communication performance in various environments, also under different conditions which include:

1. Nursing-home simulation: The first test involves measuring the communication performance between node devices in a single-story compound which resembles that in nursing homes.

2. Multistory building simulation: The second test involves determining the number of nodes which are required to route the data across multiple-story building in order to minimize the number of lost packets (data); we focus on the communications between the nodes in connecting floors (i.e., floor-1 to floor-2, and so on.)

3. Outdoor forest simulation: The last test involves measuring the communication performance for an outdoor monitoring system; here we focus on the placement of nodes in order to achieve the highest possible radio reception ratio between the intermediate devices (forwarding nodes). This makes sure that each node has at least one symbolic link which routes its data to the base-station.

The tests are performed around various environmental obstacles including concrete walls, doors, trees, and other potentially disruptive objects. The aim of these tests is to measure the communication performances in each environment which we later use during the final stages of the our deployment.

4.2.1.1 Nursing-homes communication analysis

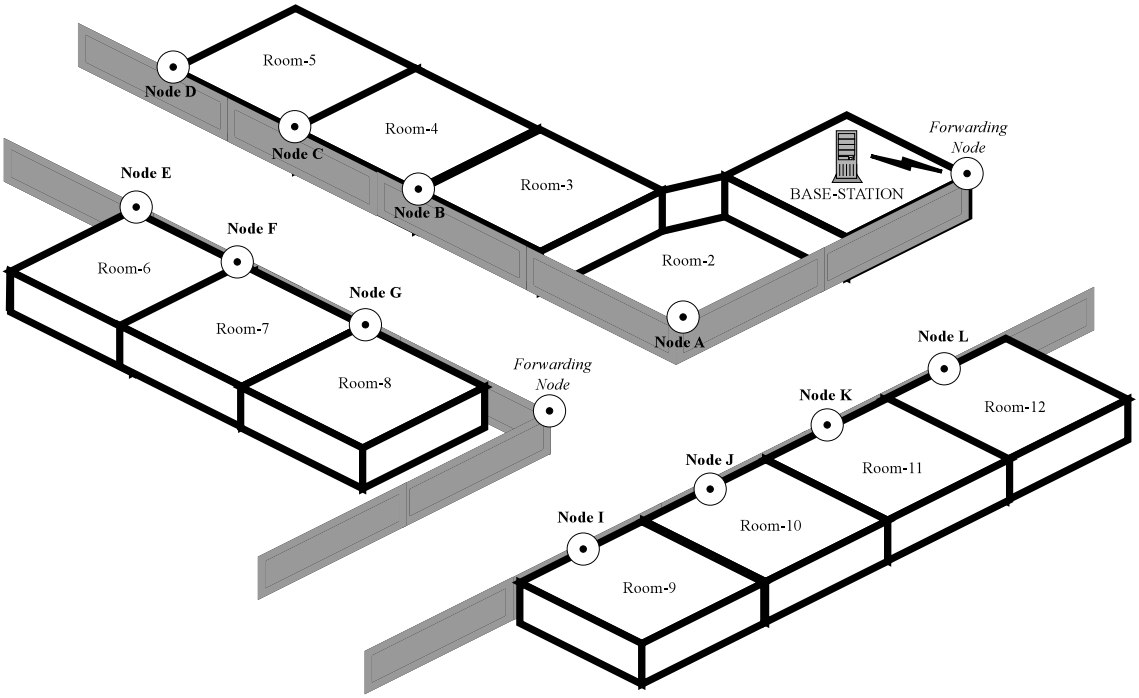


Figure 4.7: Nursing-home environment simulation.

In our first simulation, we setup a WSN to convene a similar architecture as those found in nursing-homes as shown in Figure 4.7. A total of two forwarding nodes were deployed across the pathways around the single-story complex in order to forward the packets from the surrounding nodes.

Barrier Analysis	
Physical Characteristic	*Number of Barriers*
Cement walls	$T_{walls} = 16$
Wooden doors	$T_{doors} = 36$
Physical Node Analysis	
Node Characteristic	*Equivalent Value*
Number of forwarding nodes	$T_{fowarding-nodes} = 3$
Maximum distance to base-station	$D_{max(Node-BS)} = 36\ meters$
Maximum number of hops (per node)	$T_{max(hop)} = 7$
Maximum number of network route request (per node)	$T_{max(RREQ)} = 3$
Maximum number of re-transmissions (per node)	$T_{max(re-transmit)} = 2$
Communication Performance Analysis	
Communication Criteria	*Rating*
Successful radio reception ratio	$RRT_{success} = 86\%$
Failed packet transmission ratio	$RRT_{fail} = 14\%$

Table 4.4: Communication performance analysis in nursing-home environment.

The environment was surrounded by obstacles such as wooden doors and concrete walls to simulate a real-life deployment; sensing nodes were deployed across a range of rooms which gathered

environmental data including temperature and air-quality parameters. The intermediate nodes act as forwarding nodes when the distances did not allow for point-to-point (p2p) communication between the source and the gateway (e.g., Node-D). Our tests ran for 2000 random packet transmissions, and recorded excellent radio reception ratios ($\geq$ 86%) throughout the communication system. The summary of the nursing-home simulation test is shown in Table 4.4, which includes the environmental conditions, node parameters, and the main results of this experiment.

4.2.1.2 Multistory communication analysis

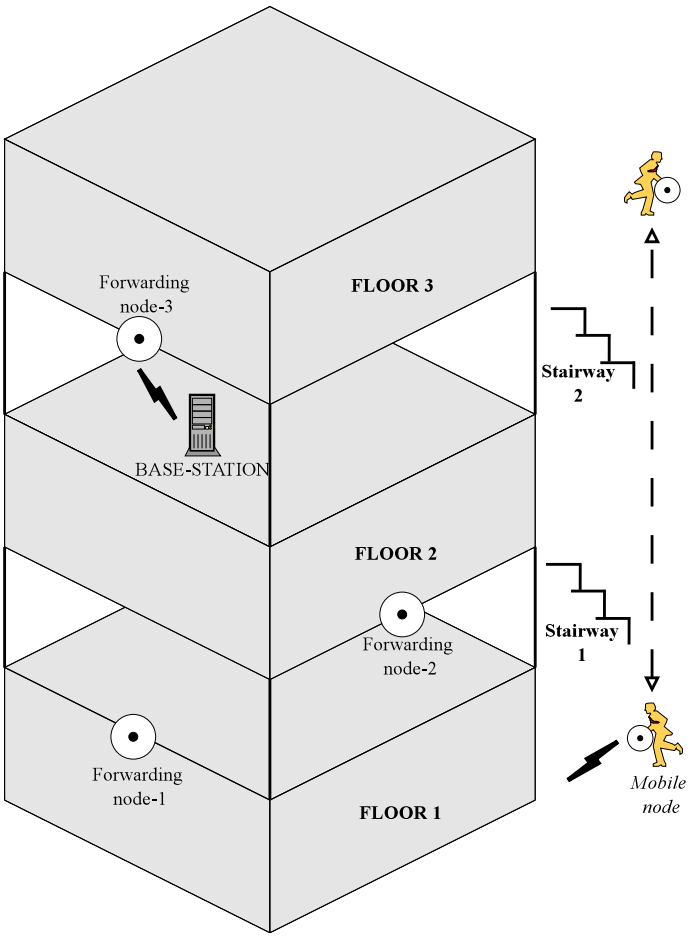


Figure 4.8: Multistory environment simulation.

In our second simulation, we investigate the different behaviors which are produced by introducing mobile nodes into a WSN in a multistory complex as shown in Figure 4.8. Our aim is to gain a better understanding of the characteristics in mobile WSNs, their routing effectiveness and transmission reliability using AODV. Furthermore, this experiment is used to model the communication performance in mobile patients whom use WSN nodes to monitor their locations in an indoor environment; deployment of forwarding nodes (stationary) are subject to two conditions: a) random deployment, and b) deployment using RSSI. Finally, we use two types of nodes which include stationary nodes (forwarding nodes) and mobile nodes (which are attached to the mobile person).

Random Localization			
Node Location	*Packets Forwarded*	*Random Localization*	*RSSI Localization*
Floor 1	$T_{total(forwarded)} = 101$	$RRT_{total} = 59\%$	$RRT_{total(success)} = 82\%$
Floor 2	$T_{total(forwarded)} = 42$	$RRT_{total} = 57\%$	$RRT_{total(success)} = 78\%$
Floor 3	$T_{total(forwarded)} = 42$	$RRT_{total} = 70\%$	$RRT_{total(success)} = 85\%$

Node Paramteres	
Node Characteristic	*Equivalent Value*
Average packet size	$Average_{packet} = 99\ bytes$
Maximum distance to base-station	$D_{max} = 35\ meters$

Table 4.5: Communication performance analysis in a multistory environment.

In this experiment, we set the transmission rate to occur randomly; therefore, there are no guarantees that during a specific transmission, the base-station is in coverage distance of the mobile node. The first part of this experiment, we randomly dispersed the nodes across the three floors; the results we obtained were not suitable in terms of radio reception throughput ($\leq 70\%$). However, with the introduction of RSSI mapping techniques, the RRT increased across all the forwarding nodes ($\geq 78\%$). The summary of this experiment is shown in Table 4.5 which includes the results of both RSSI and the random localization of nodes.

4.2.1.3 Outdoor communication analysis

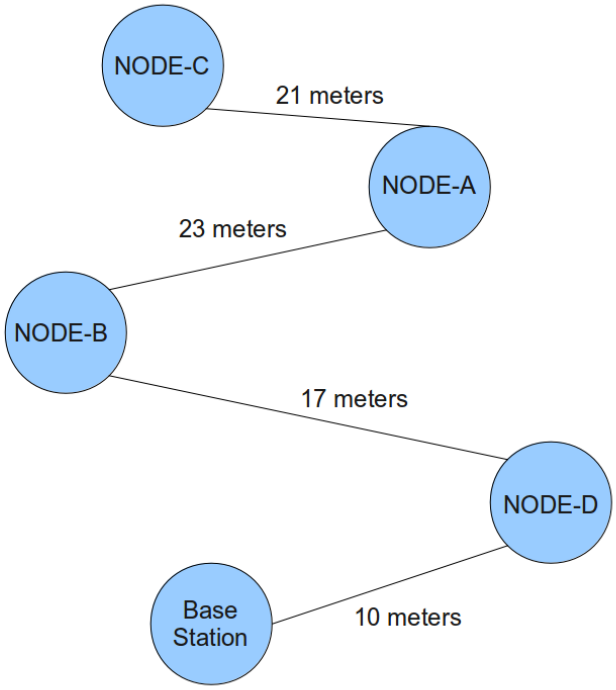


Figure 4.9: Outdoor environmental simulation.

This experiment is setup to investigate the communication performance in outdoor mesh networks. Node devices measure concentrations of various gases and temperature and transmit data

to a base-station every minute. The base-station receives data which consists of sensor and power calculations, then processes them into a database (MYSQL).

Experiment Parameters				
Criteria	*Description*			
Communication protocol	Digimesh			
Routing protocol	Ad-hoc On-demand Vector Distance (AODV)			
Radio frequency	2.4 GHz			
Sensors	CO_2, CO, CH_4, NO_2, O_2, temperature, humidity, and fuels			
Coveragte area	620 m^2			
Communication Performance Analysis				
Criteria	*Node-A*	*Node-B*	*Node-C*	*Node-D*
Distance to BS	50 meters	27 meters	71 meters	10 meters
Distance to nearest node	21 meters (C)	17 meters (D)	21 meters (A)	10 meters (BS)
Packets transmitted	61	56	60	62
Radio Reception Throughput	100%	98%	96%	96%
Initial voltage	4.14 V	4.11 V	4.09 V	4.12 V
Final voltage	4.03 V	4.03 V	4.01 V	4.02 V
No. of sensors	2	2	3	2

Table 4.6: Communication performance analysis in outdoor environment.

The experiment took place at a forest park located in Edith Cowan University (Appendix-C). A total of 5 nodes including the base-station are deployed in a line-of-sight (LOS) arrangement as shown by Figure 4.9; however, only a single node (Node D) was within a single hop from the base-station while others were out-of-bound. The off-limit nodes require multiple hops in order to transmit data to the base-station; which is taken care of by the AODV routing protocol. Furthermore, the results of this experiment are highlighted in Table 4.6 which suggest:

- Acceptable power consumptions considering the detection frequency and the size was moderately large; based on minute intervals with average packet size of 78 bytes.
- Radio reception throughput (RRT) was very high which suggest that the DM protocol meets all requirements for a real-life deployment of WSNs in dense vegetation or forest environments.

- It is possible to cover large areas with a small number of nodes; in our experiment, we only used 4 nodes and covered an area of 620 m^2.

4.2.2 Data Routing Analysis

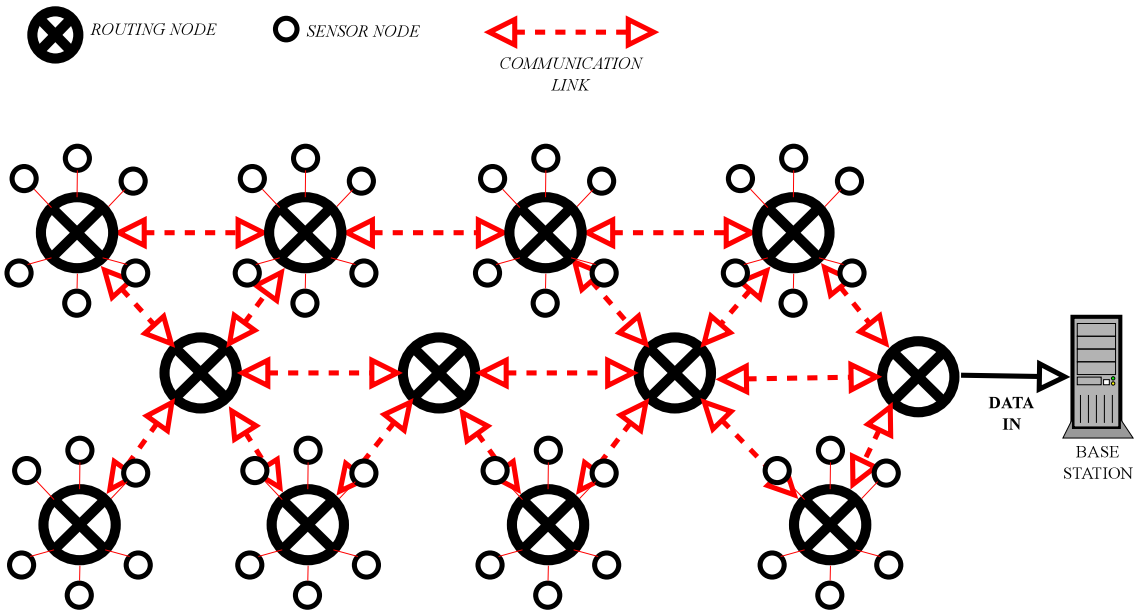


Figure 4.10: Stand-alone 802.15.4 routing protocol.

The IEEE-802.15.4 protocol does not specify any routing algorithms in its standard, as it only deals with PHY and the MAC layer. Hence, the use of this protocol is limited to p2p communications over short distance; denser p2p networks require a larger number of data-sinks in order to account for all the data. Due to these limitations, we developed our own protocol which enables routing in stand-alone 802.15.4 WSNs as shown in Figure 4.10. This protocol involves two types of node devices which include:

* Routing nodes: Which forward the data through predefined communication paths between source transmitters and the base-station.

* Sensing nodes: Which sense the environment, then transmit their data using ordinary p2p communication to specified routing nodes.

By referring to Figure 4.10, we notice the special arrangement of the routing nodes in the system; this factor imposes two important conditions that come into the realization of this protocol, and they are:

1) Routing node location: The placement of the routing nodes in the environment is absolutely crucial; sensor nodes have to be within coverage distance between at least two forwarding nodes (in case a path fails). A routing node sole responsibility is to receive data from the sensing node, then forwards it to an intermediate node until the data reaches the base-station.

2) Sensor node location: A group of sensor nodes form a multi-p2p topology with a single parent (routing node). A sensor node only transmits to a single routing node, hence it is important that communication between the two nodes are not disrupted (to prevent data loss).

Algorithm 4.10 Forwarding node setup in stand-alone 802.15.4 WSNs.

```
packetXBee* paq_sent;
void get_packets(){
  if( XBee.available() ){ //Checks data_buffer
    char* _data = (char*) calloc(1,sizeof(char));
    xbee802.treatData(); //reads API frame
    if( !xbee802.error_RX ){
        while(xbee802.pos>0){
         Extract_Data(); //executes the extraction program.
         free(xbee802.packet_finished[xbee802.pos-1]);
         xbee802.packet_finished[xbee802.pos-1]=NULL;
         xbee802.pos--;
        }
    }
  }
  SendToBS();
}
//Exctracting Data Function//
void Extract_Data(){
  for(int f=0;f<xbee802.packet_finished[xbee802.pos-1]->
      data_length;f++){
    XBee.print(xbee802.packet_finished[xbee802.pos-1]->data[f]
               ,BYTE);
    }
  strcat(_data,(xbee802.packet_finished[xbee802.pos-1]->data));
}
```

Algorithm 4.10 highlights the software configuration of the routing node in which data is extracted, then stored into an array, and finally transmitted to the intermediate routing node.

4.2.2.1 Router node discovery

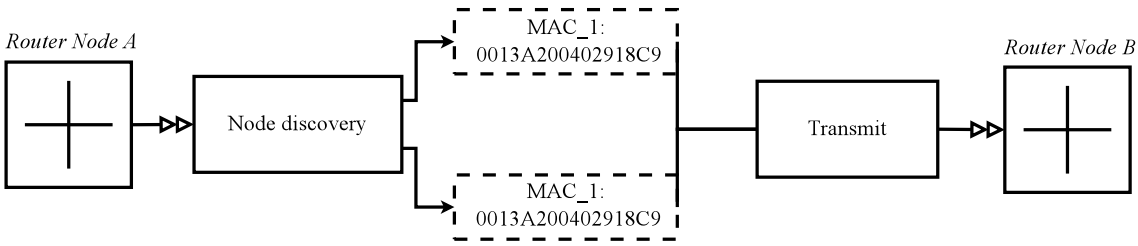


Figure 4.11: Node discovery in 802.15.4 routing.

Node discovery is a process which allow us to locate the intermediate routing node in a multi-hop system. This implies to the device-ID which falls within the communication path between the sensing node and the base-station. Figure 4.11 demonstrates the node discovery process between the router nodes; router-node-A issues a discovery request which returns the node addresses that

operate within a single hop from the host device (i.e., router node B). Finally, data gets extracted from the radio buffer and transmitted to the next destination, and so on.

4.2.2.2 AODV routing

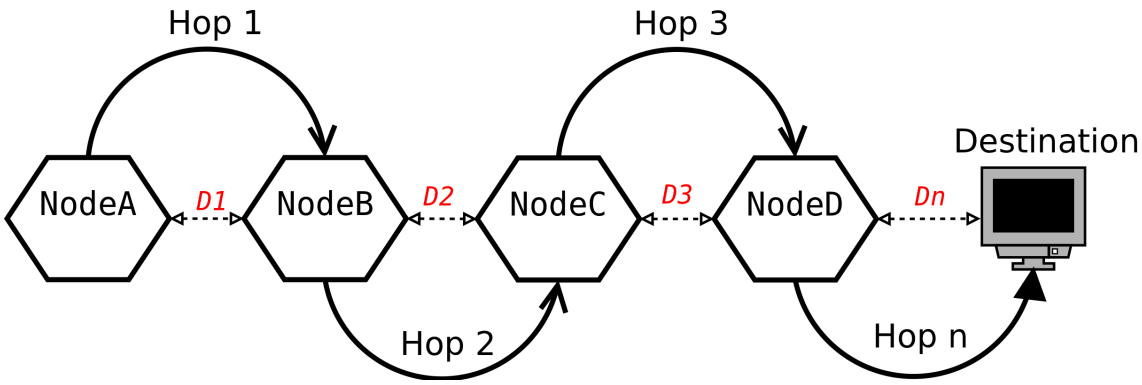


Figure 4.12: Multi-hop data network.

A multihop network has the ability to transmit data between nodes in a non-line-of-sight (NLOS) architecture using routing algorithms such as AODV (Ad-Hoc Distance Vector). In this routing technique, sensor nodes transmit their data using multiple hops as illustrated by Figure 4.12. The benefits associated with multihop networks include:

1. Allows long distance communication: Data packets flow through as many hops until they reach the base-station node, given that distance between the original node and the destination is not direct.

2. Decreases the power consumption in the WSN: Due to the reduction in distances between intermediate nodes which is determined by their RSSI (Received Signal Strength Indicator) values. Therefore, data packets are forwarded to the node which has the lowest transfer time.

4.2.3 Network Architecture

The general network model is based on WSNs and IP technology as shown in Figure 4.13, which consists of three main systems:

* IP base-station: Also referred to as Meshlium; this device integrates an FPGA board which runs an embedded Linux kernel (Voyage-Linux) aimed at low-end x86 platforms. It supports multiple radio communication modules which includes Wi-Fi (2.4 and 5 GHz bands), Zigbee/802.15.4/DM, and GPRS/GPS.

* WSN platforms: Which includes sensor devices and all their additional peripherals such as sensors, radio-units, external modules (memory, SIM-card, GPS, etc.). Environmental applications are built around these devices which act independently of the IP back-bone.

* Network components: They include the DSL modem, dual-band Wi-Fi router (2.4 GHz and 5 GHz), and workstations (desktop comupters, laptops, PDAs, etc.).

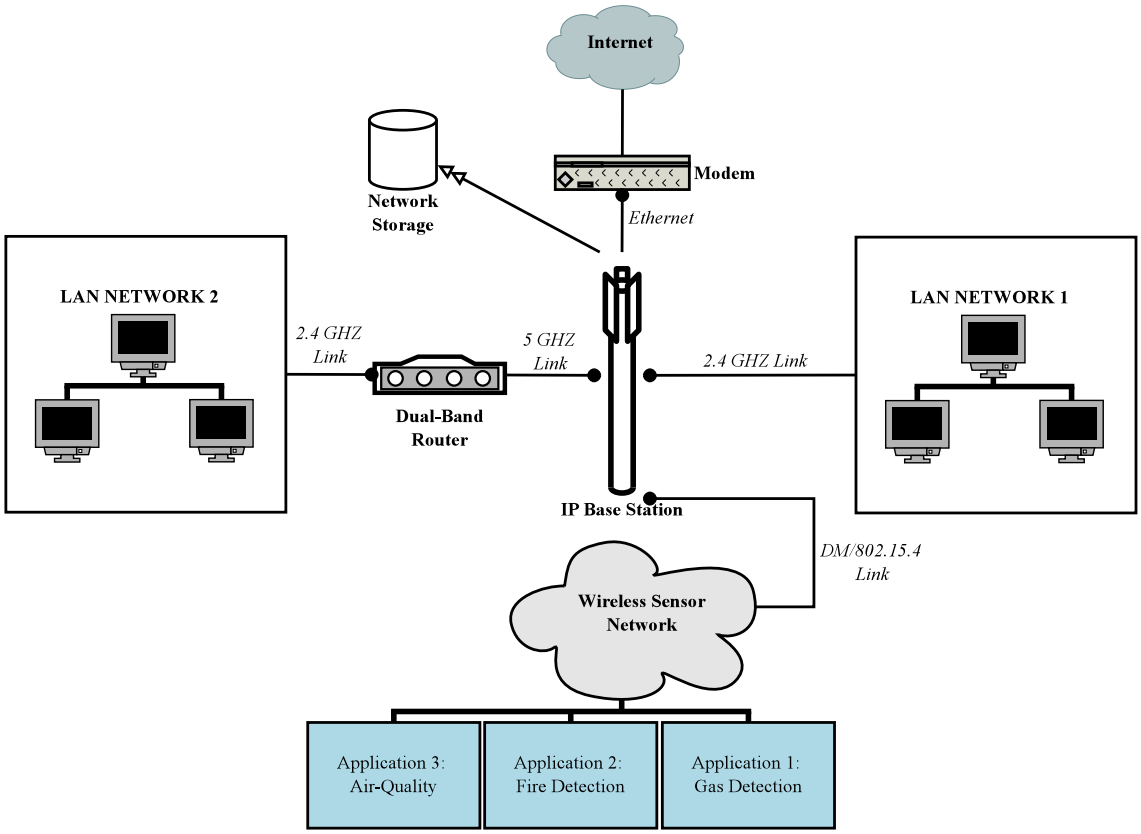


Figure 4.13: General network model.

The network model in Figure 4.13 is also referred to as a WSN-over-IP, because it provides additional functionalities that are generally associated only within IP networks. This hybrid architecture offers a lot more services than a traditional WSN including:

- WSN communication over the Digimesh or 802.15.4 protocol which is provided by a traditional WSN setup.
- Remote access capabilities using traditional IP software such as SSH and openVPN.
- Integration with Internet services including web-servers, MYSQL database, and online blogging (twitter).

The Mesh-Hood (M-Hood) system is a practical example of a WSN-over-IP which involves the architecture highlighted by Figure 4.13. Our main contributions are primarily in: a) the conceptual design, b) hardware configuration of the system, and c) implementation in an indoor air-quality monitoring application.

4.2.3.1 WSN-Over-IP (Mesh-Hood System)

The M-Hood concept applies towards all subscribed network entities including the workstations (desktop, laptops, mobile phones, tablets, etc.), access points, node devices, and sensor data, as

being meshed together so that access to the entire network body is independent from the host location (any specific workstation) or the source gateway (router, modem,etc.).

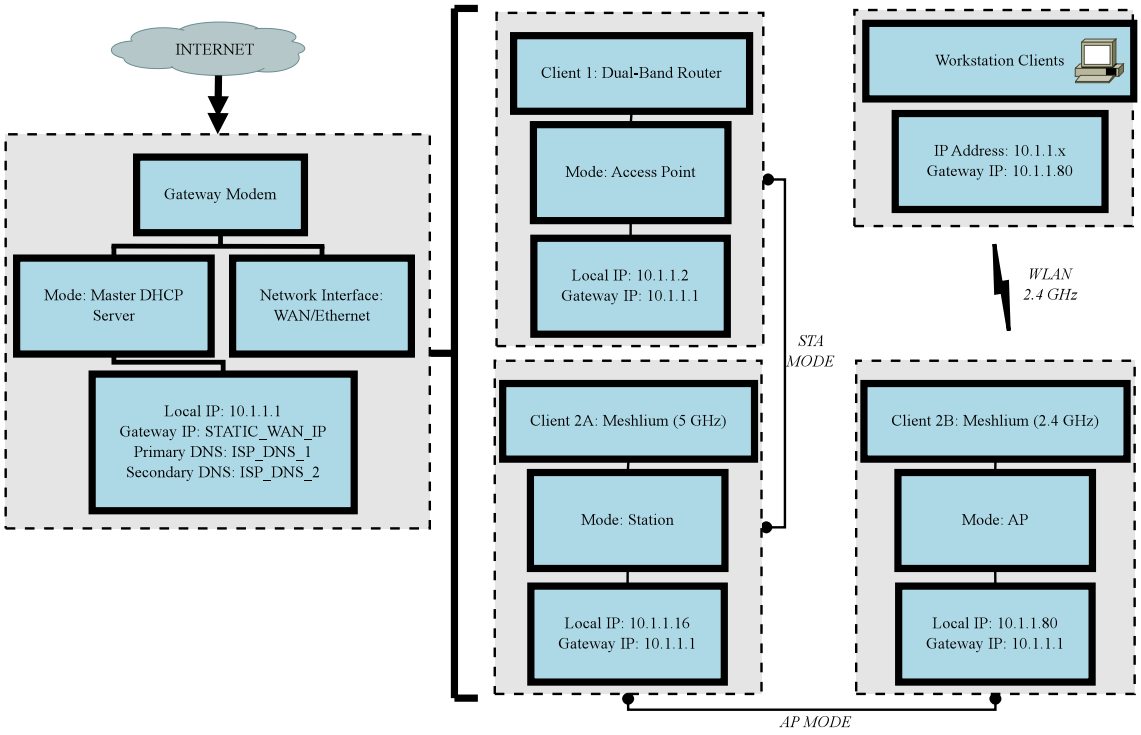


Figure 4.14: Mesh-Hood IP backbone.

The M-Hood network configuration is shown in Figure 4.14, which represents a real-life architecture of a WSN-over-IP system. Moreover, the radio architecture is based on virtual access points (VAPs); logical entities which exist within a physical access point. Each VAP is capable of advertising distinct SSIDs and capabilities set, hence they may appear as independent stations (to users), but the reality is they all belong to a single physical network entity (interface/card). Furthermore, there are two VAP modes used in the M-Hood system which include:

1. Station (STA): In this mode, the meshlium node connects to the dual-band router over the 5 GHz Wi-Fi band.

2. Access point (AP): In this mode, meshlium acts as a central point for workstations and other network devices in order to gain access to the net.

4.2.3.2 IP testing

IP Test Tools	
Tool	*Description*
Ping test	The ping test is performed to verify the correct operations of the mesh network, and to ensure that clients and servers are able to communicate to one another.
Route test	The traceroute operation tests the connections between different network interfaces in order to verify the correct flow of data communications (including the number of intermediate hops).
Iperf test	The iperf operation is used to measure the maximum TCP and UDP bandwidth performance in the network.

Table 4.7: IP testing tools.

IP offers convenient methods in order to test the communications between all networked systems; Table 4.7 describes the most widely used tools which include: a) ping test, b) trace-route, and c) iperf. In the case of the network depicted by Figure 4.14, communication performance testing is determined by the following conditions:

1. Workstation communications: To confirm that workstations can communicate between each other, and also with IP base-stations, routers, and gateways.

2. Internet connection: To verify Internet access is available through all operational radio bands which include 2.4 GHz and 5 GHz.

3. Network AP communications: To confirm that access points are correctly configured in the network; this applies to the method in which meshlium communicates with the router nodes and workstations, and vice versa.

4.2.3.3 Network Storage

There are two methods in which data is stored (in the M-Hood system):

* Local database: Which is hosted on the meshlium device itself; configured using phpmyadmin accessed through the local gateway interface (Meshlium Manager System).

* Dedicated network servers: Which are hosted on workstations within the same network infrastructure of the meshlium device (i.e., dedicated machines in a LAN). In this method, we setup an external server to store all sensor data in databases (MYSQL); using the lampp package to configure the apache and MYSQL servers on each workstation.

The type of data stored in the database system is shown in Table 4.8, which includes: type of gas, output voltage (V_{out}), sensor resistance (R_s), resistive gain (R_s/R_o), and most importantly its concentration in air (ppm).

Sample of Sensor Measurements				
Type of Sensor	V_{out} (Volts)	R_s ($k\Omega$)	R_s/R_O	ppm
CO	0.5677	78.0681	5.8697	21.500
NO_2	0.5096	7.1594	3.1373	0.0557
O_3	1.3045	4.2307	0.3846	15.5672

Table 4.8: Storage of sensor measurements in databases.

4.3 Applications

In this section, we propose the environmental applications using WSNs as shown in Figure 4.6, which includes gas-leak detection, air quality monitoring, and fire sensing. In particular, we focus on these three main issues:

* Operation theory: Which refers to the methods of operation in real-time environments including the types of sensors used, and network communication models.

* Obtention of results: We analyze the practical results obtained by the sensor network from each different environment.

* Real-time responses: Which refers to the actions generated by node devices during emergency events such as fire occurrences, or pollution risks. Finally, we analyze the tasks (node operations) which are associated with a specific emergency and application (i.e., fire sensing, air quality, etc.).

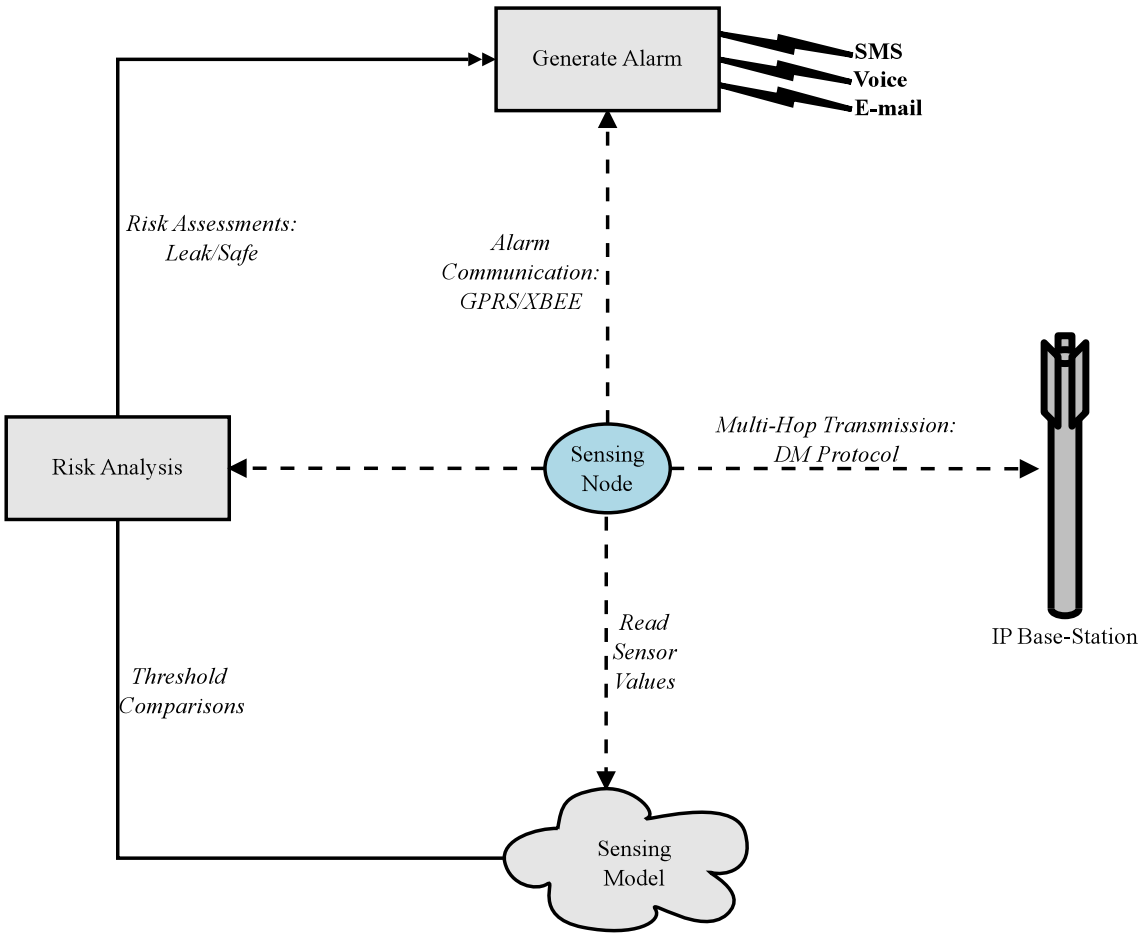


Figure 4.15: General application model.

The general application architecture is shown in Figure 4.15; in this model, there are three major parts involved, which include:

1. Data transmission: Which specifies the method in which data is transmitted from the sensing nodes towards the base-station.

2. Sensing elements: Which specifies the interactions between the sensors with their surrounding environment; i.e., the determination of the response based on the sensing model we described in Section 4.1.1.

3. Alarm notification: This implies the three alarm signals generated by each node device including SMS, voice-call, and e-mail alarms.

In the next discussion, we analyze the three applications and highlight some of the results obtained by each sensor network.

4.3.1 Gas-Leak Detection

The underlying operation of this system is to measure air quality levels in order to detect gas leaks of certain gases (mainly natural gas). Node devices periodically switch-on their peripherals (sensors, communication modules, etc.) and begin sensing the surrounding environment. As a result, we obtain a comprehensive overview of the characteristics and the nature of the encompassing environment. In Table 4.9, we provide a small sample of results that correspond with temperature, humidity, and natural gas (CH_4) concentrations for an indoor environment.

Gas-Leak Analysis (sample results)	
Temperature (C^0)	*Humidity (%)*
25.8064	21.6903
21.2903	20.8645
25.8064	20.3483
24.5161	20.658
21.6128	20.2451
27.4193	20.3483

Methane-Gas Concentrations			
V_{OUT}	R_s ($k\Omega$)	R_s/Ro	*ppm*
0.1258	17.4355	25.6404	14.2891
0.1161	18.9298	27.8380	15.0461
0.1032	21.3523	31.4004	16.2276
0.0903	24.4669	35.9807	17.6756
0.0838	26.3996	38.8230	18.5397
0.0774	28.6197	42.0878	19.5037
0.0741	29.9143	43.9917	20.0529

Table 4.9: Gas-leak sample results.

Take a note regarding the variation in methane concentrations as highlighted from Table 4.9; another property of the gas-leak system is the ability to generate an alarm at a specific concentration for any type of gas (e.g., CH_4). A user-defined threshold sets the boundary limits for each gas concentration in a polluted environment, once exceeded the node device responds with an alarm signal using one of its external radio modules (GPRS, XBee).

4.3.2 Air-Quality Monitoring

The underlying operation regarding the air-quality system involves monitoring the concentration of numerous air-containment substances from surrounding environments such as hospitals, schools, and home-care patients (ADLs). Node devices evaluate the risk of exposure to certain air pollutants and generate an alarm accordingly (depending on the risk involved). Furthermore, an IP gateway maintains a database which holds records of all the gas measurements as shown in Table 4.10.

Oxygen Levels in Normal Air	
V_{out}	*Relative Oxygen Levels(%)*
0.6193	21.772
0.6129	21.516
0.6096	21.384
0.6064	21.256
.	
.	
.	

General Air Quality Sensors (type 1 and 2)			
$S1_{out}$	$(R_s/R_o)_{S1}$	$S2_{out}$	$(R_s/R_o)_{S2}$
2.5000	1.0000	2.5000	1.0000
2.5000	1.0000	2.5000	1.0000
2.5000	1.0000	2.5000	1.0000
.			
.			
.			

Ammonia Concentrations			
V_{OUT}	R_s $(k\Omega)$	R_s/Ro	ppm
3.2774	4.2048	0.5256	1.2451
3.2064	4.4750	0.5594	1.2947
3.1612	4.6534	0.5817	1.3269
2.8258	6.1553	0.7694	1.5816
2.6548	7.0670	0.8834	1.7249

Table 4.10: Air-quality levels sample results.

The results shown in Table 4.10 correspond with three types of sensors: a) Oxygen, b) Air-contaminants, and c) Ammonia. First, the oxygen sensor provides its output in terms of the relative molecular concentration in air (21% in normal conditions). Second, the air-contaminant sensors provide a constant gain ($R_s/R_o = 1$) under normal air levels, hence the output voltage is equal to 2.5 Volts. Finally, the ammonia sensor, although uncalibrated, it still indicates the presence of ammonia gas in air from its decreasing voltage output (V_{out}).

The data which gets transmitted to the IP base-station can be accessed by numerous workstations which are connected through a LAN. Assuming the system is implemented throughout an environment containing patients suffering from certain cardiovascular illnesses, then data can be easily integrated into medical databases that doctors use to analyze the impacts of air pollutants on a patient's health. Similarly, home-carers can access live-data streams directly patients gateways using either SSH/MYSQL or through a web-browser.

4.3.3 Fire Sensing

The underlying operation of the fire sensing system is to detect breakouts of natural fires and also be able to observe the signs which lead to their occurrences in the surrounding environment (i.e., preventive measures). Bush-fire releases smoke which contains carbon monoxide (CO), carbon dioxide (CO_2), nitrogen oxides (e.g., NO_2), and dust particles. In our system, node devices are equipped with gas sensors which can identify the presence of those gases just mentioned; hence, any variations in air quality levels (i.e., rising pollution) would trigger an immediate response from the node device, followed by an alarm which gets transmitted to the fire authorities using the on-board GPRS modules (SMS, voice call).

Carbon Dioxide Concentrations in Normal Air			
V_{out}	$\triangle EMF$	*ppm*	Condition
0.00967	0.1903	335.6518	Normal
0.00645	0.1935	335.6914	Normal
0.07096	0.1290	334.8992	Normal
Oxygen Levels in Normal Air			
V_{out}	*Relative Oxygen Levels(%)*		Condition
0.6193	21.772		Normal
0.6129	21.516		Normal
0.6096	21.384		Normal
Carbon Monoxide Levels in Normal Air			
V_{out}	R_s/Ro	*ppm*	Condition
0.2677	13.2914	10.5723	Normal
0.2032	17.7491	8.2240	Normal
0.1193	30.7603	5.1012	Normal
	.		
	.		
	.		

Table 4.11: Fire sensing sample results.

In Table 4.11, we provide measurements taken from a normal environment of by-product fumes of fire smoke) and oxygen levels. In this type of environment (i.e., no pollution), carbon dioxide levels are approximately 330 *ppm*; the CO_2 sensor reads the output voltage (EMF) which is then subtracted from the voltage at 350 *ppm* ($V_{350ppm} = 220\ mV$) to give us the voltage variation in the electromotive force (ΔEMF). The concentration of the carbon dioxide gas is related to ΔEMF by Equation 4.1.

$$CO_{2(ppm)} = 10^{(\Delta EMF + 158.630)/62.877} \tag{4.1}$$

Similarly, the oxygen sensor reads the output in voltage, then gets converts into relative molecular concentration; i.e., the percentage of O_2 molecules in air. Table 4.11 contains the oxygen concentrations from a non-polluted environment which is around 21%. Finally, the carbon monoxide sensor also provides voltage output, which is then converted into parts-per-million (ppm) units using its corresponding equation from Table 3.7.

Conclusion

In this chapter, we discussed three main applications that are based on WSNs, which include: gas leak detection, air-quality monitoring, and finally a fire sensing system. Each application had its own characteristics, method of operation, and preferred working environment (indoor, outdoor). We have analyzed the many properties of these sensor networks, and conducted major performance tests based on the following parameters:

- Communication systems: We analyzed the communication performance in node devices based on the radio reception throughput (RRT); which is a measure which determines the transmission quality between numerous inter-connected objects (node devices, IP-basestation).
- Networking systems: We conducted tests on three deployment architectures including stationary, mobile, and outdoor sensor networks. Multi-hop data transmission was implemented throughout these experiments which allowed us to achieve longer communication distances and wireless coverage.
- WSN over IP architecture: We developed a network architecture of a WSN with the added the functionalities of IP systems including remote connectivity, seamless integration with hardware, and online services. The M-Hood system is a hybrid sensor network based on multiple communication methods, namely using 802.15.4/Digimesh and Wi-Fi (2.4 GHz and 5 GHz). The system is used for live data streaming, database integration, and remote connectivity throughout LANs and WANs.
- Sensor models: We developed sensor models used to measure the physical environment in order to determine pollution risks, fire occurrences, and possible gas-leaks from its surroundings. Consequently, an alarm is generated according the nature of risk; whether it is a high, medium, or a low risk, the sensing model specifies the exact operations and transmission methods for these different situations.
- Analysis of results: Which includes the measurements generated during the course of conducting numerous experiments and application procedures, such as environmental simulation (nursing home, forests, and multistory complex units), where we investigated the routing efficiency. Finally, we analyzed the results which coincided with the gas measurements from within the main applications including air-quality levels, fire smoke, and natural gas leaks.

Chapter 5

CONCLUSION AND FUTURE WORKS

WSNs technologies continue to evolve as a high-end application platforms designed to integrate seamlessly with their surrounding environments. Their applications would extend beyond comprehension once our realm (world) is dominated by wireless communications. Today, we are witnessing an evolution of WSNs towards the Internet of Things (IOT); the time when every electronic hardware becomes part of the vast machine, or as we would like to call it: the *"globalisation"* of the Internet. And WSNs will have a massive role into it. WSN technology is yet to peak (in terms of availability and social-awareness), and as long as research continues to embrace it; it will only get better, smarter, and more affordable for everyone. In this thesis, we showed how to efficiently configure a WSN in a simple but effective manner; free from all technical complications and tedious programming. The ultimate goal is for WSNs to become plug-n-play, but the technology isn't yet at a point to do just that. What we need is a universal standard which deals exclusively with the application layer protocol for WSNs. Consequently, we are able to develop portable systems and address the interoperability in WSN hardware.

The integration of the Internet Protocol (IP) into WSNs is a massive leap in terms of technological advancement. The Internet as commonly interpreted is a communication entity (global network) which allows devices to connect with one another and exchange data. In this thesis, we presented a WSN-over-IP architecture which uses the Internet as a communication tool utilized as outlined below:

- Data depository: Using online facilities (e.g., MYSQL or FTP servers) to archive captured data from WSNs (readily-available to users). We can think of the Internet as an unlimited virtual space (memory) that we use to store data such as sensor measurements from a WSN.

- Remote connectivity: WSNs are generally designed for short range communications, therefore one must be within proximate range to their facilities in order to establish a meaningful connection. The internet is suitable to create virtual ad-hoc links; a link from a PC to WSN

via an IP-gateway that provides access-routes to the WSN base-station (coordinator). This way, we control node devices from any remote location that has adequate telecommunication infrastructure (i.e., Internet connectivity).

- Social networking: Today, social networks play big roles in peoples lives; twitter and facebook have both struck a chord in our societies, and now everybody is seemingly connected online. In addition, with social networking comes awareness; the way events spread out across those platforms are as effective as an ant colony network. The impact of social networking have on WSNs lays within the safety and protection of society (e.g., fire monitoring).

Finally, we presented three environmental systems in which WSNs are principally utilized to gather environmental data from the surroundings. These systems were: **a.** Gas-leak detection, **b.** Air-quality monitoring, and **c.** Fire sensing. Strategic algorithms are instilled to govern sensing operations and models in which environmental risks were assessed upon specific parameters. In addition, the management of node components subsystems was addressed by strict protocols in which periphery functions (e.g., writing to μSD memory, radio transmission) are utilized under specific admission techniques (to increase battery efficiency). A summary which specifies the major features of this thesis is outlined below as follows:

- Low-power approach: In which techniques to reduce power consumption are utilized to extend the longevity of WSNs. Three power modes were specified: sleep, deep-sleep, and hibernate; the amount of energy consumed differs from one mode to another. Battery-low warnings are managed by internal circuitry (on the node device) which detect the relative capacity (i.e., percentage) of energy in the reservoir; an alarm is generated to signal depleting power levels when a user-defined threshold is exceeded.

- Hybrid network architecture: Data communications are specified according to multiple protocols including 802.11 (Wi-Fi), 802.15.4 (Digimesh), NMEA (GPS), and mobile-2G/3G (GPRS) that nodes use in combination to perform a variety of functions; e.g., sending SMS, initiating voice-calls, and uploading data using FTP.

- Multi-purpose sensors utilization: Observable phenomenas (gravity, speed, growth rate, etc.) are events that we see, observe, and comprehend; however, scientific instruments are required to measure, analyze, and record data concerning them. A gas sensor is concerned with parameters which constitute the micro-world (atoms, molecules, and so on); we use them to obtain equivalent quantitative parameters, ones that we are able to interpret physically (i.e., voltage, resistance, current) in order to describe certain properties of a gas (e.g., concentrations).

- Web integration: A real time operating system is utilized in order to provide network management functionalities which include an e-mail generating system, a data-processing MYSQL server, and a twitter blog containing up-to-date information obtained by the WSN.

5.1 Future Works

Some possible future extensions to the investigations made in this thesis include:

1. Additional Tests on the M-Hood System

The acquisition of a larger number of nodes would enable the real deployment of the Mesh-Hood system. Nursing homes, or retirement villages are ideal locations of deployment, and for testing/implementing the proposed system. This thesis covers the architecture of the mesh-network which allows sharing of the Internet among numerous workstations and users. It also describes the technologies used to store the sensor network data on the internet, which enables live-monitoring capabilities required by many applications of WSN. Our idea was to deploy the M-Hood system in retirement villages, because these locations are suited to short range communications, which is typical in WSNs. Furthermore, integrating the M-Hood system provides benefits such as shared network infrastructure, live monitoring capabilities, and centralized network operations.

2. Real Life Testing of the Fire Monitoring System

Our tests have confirmed the functionality of the fire monitoring system under simulated/virtual environment. We have successfully tested the fundamental functions of this system which includes sensing algorithms, response to emergencies, node communications, and data transmission. The next stage involves a real-life deployment in the environment in order to generate a more comprehensive analysis on its operations in real-time.

3. Calibration of Gas Sensors

The level of calibration we achieved in our gas sensors is sufficient for monitoring purposes, but not for determining the exact concentrations of the gases in the surrounding environment. This capability often requires a more sensitive approach towards the calibration of the sensors, and also very important in the control and automation application domain. Therefore, we suggest that by re-calibrating the gas sensors at very high and sensitive levels result in improved sensor readings of the real concentrations in the environment. This would specifically benefit monitoring the air pollution levels, which rely on sensor networks for precise and accurate detection capabilities.

Bibliography

[1] K. Janani, V. Dhulipala, and R. Chandrasekaran, "A wsn based framework for human health monitoring," in *Devices and Communications (ICDeCom), 2011 International Conference on*, 2011.

[2] M. Jobs, F. Lantz, B. Lewin, E. Jansson, J. Antoni, K. Brunberg, P. Hallbjorner, and A. Rydberg, "Wban mass: A wban-based monitoring application system," in *Antennas and Propagation for Body-Centric Wireless Communications, 2009 2nd IET Seminar on*, 2009.

[3] E. Jovanov, N. Hanish, V. Courson, J. Stidham, H. Stinson, C. Webb, and K. Denny, "Avatar-a multi-sensory system for real time body position monitoring," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, 2009.

[4] G. de Lima, L. e Silva, and P. Neto, "Wsn as a tool for supporting agriculture in the precision irrigation," in *Networking and Services (ICNS), 2010 Sixth International Conference on*, 2010.

[5] W. Cao, G. Xu, E. Yaprak, R. Lockhart, T. Yang, and Y. Gao, "Using wireless sensor networking (wsn) to manage micro-climate in greenhouse," in *Mechtronic and Embedded Systems and Applications, 2008. MESA 2008. IEEE/ASME International Conference on*, 2008.

[6] J. Wilson, *Sensor Technology Handbook*, vol. 1. Newnes, 2005.

[7] R. Tan, G. Xing, J. Chen, W.-Z. Song, and R. Huang, "Quality-driven volcanic earthquake detection using wireless sensor networks," in *Real-Time Systems Symposium (RTSS), 2010 IEEE 31st*, 2010.

[8] L. Cheng and S. Pakzad, "Agility of wireless sensor networks for earthquake monitoring of bridges," in *Networked Sensing Systems (INSS), 2009 Sixth International Conference on*, 2009.

[9] "Part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans)."

[10] Libelium, "Digimesh: Networking guide," tech. rep., University of Zaragoza, Spain, 2011. This source provides technical details on the DigiMesh networking.

[11] L. Klein-Berndt, "A quick guide to aodv routing," tech. rep., National Institute of Standards and Technology, 2008.

[12] K. Lahiri and A. R. C. Park, "Battery discharge characteristics of wireless sensor nodes: An experimental analysis," in *In Proceedings of the IEEE Conf. on Sensor and Ad-hoc Communications and Networks (SECON)*, 2005.

[13] C. Kompis and P. Sureka, "Power management technologies to enable remote and wireless sensing," tech. rep., ESP Central Ltd (the Electronics, Sensors, and Photonics KTN), 2010.

[14] B. G. Adam Dunkels and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, 2004.

[15] A. Dunkels, "Making sensor networks ipv6 ready," in *Networked Embedded Sensor Systems (ACM SenSys)*, 2008.

[16] A. Dunkels, "Rime — a lightweight layered communication stack for sensor networks," in *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, 2007.

[17] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Demo abstract: Software-based sensor node energy estimation," in *Proceedings of the Fifth ACM Conference on Networked Embedded Sensor Systems (SenSys 2007)*, 2007.

[18] F. Osterlind and A. Dunkels, "Throughput, approaching the maximum 802.15.4 multi-hop," in *in Proceedings of the Fifth ACM Workshop on Embedded Networked Sensors*, 2008.

[19] A. Dunkels, L. Mottola, N. Tsiftes, Fredrik Österlind, J. Eriksson, and N. Finne, "The announcement layer: Beacon coordination for the sensornet stack," in *Proceedings of EWSN 2011*, 2011.

[20] A. Dunkels, "The contikimac radio duty cycling protocol," tech. rep., Swedish Institute of Computer Science, 2011.

[21] P. Kumar, M. Gu andnes and, Q. Mushtaq, and Schiller, "Optimizing duty-cycle for delay and energy bound wsn applications," in *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, 2010.

[22] M. Arshad, N. Kamel, N. Saad, and N. Armi, "Performance enhancement of wireless sensor nodes for environmental applications," in *Intelligent and Advanced Systems (ICIAS), 2010 International Conference on*, 2010.

[23] L. Yong-Min, W. Shu-Ci, and N. Xiao-Hong, "The architecture and characteristics of wireless sensor network," in *Computer Technology and Development, 2009. ICCTD '09. International Conference on*, 2009.

[24] N. Suryadevara, A. Gaddam, S. Mukhopadhyay, and R. Rayudu, "Wellness determination of inhabitant based on daily activity behaviour in real-time monitoring using sensor networks," in *Sensing Technology (ICST), 2011 Fifth International Conference on*, 2011.

[25] R. V. P. Yerra, A. K. Bharathi, P. Rajalakshmi, and U. B. Desai, "Wsn based power monitoring in smart grids," in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2011 Seventh International Conference on*, 2011.

[26] L. Shixing, X. Wujun, and Z. Yongming, "Research and implementation of wsn in fire safety applications," in *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, 2010.

[27] A. V. Andras Nasa, Xenofon Koutsoukos, "Air quality monitoring with sensormap," in *International Conference on Information Processing in Sensor Networks*, 2008.

[28] R. North, "A mobile environmental sensing system to manage transportation and urban air quality," in *IEEE International Symposium on Circuits and Systems*, 2008.

[29] O. Postolache, "Indoor monitoring of respiratory distress triggering factors using a wireless sensing network and a smart phone," in *Instrumentation and Measurement Technology Conference*, 2009.

[30] L. J. Wu Zhengzhong, Liu Zilin and H. Xiaowei, "Wireless sensor networks for living environment monitoring," in *WCSE*, 2009.

[31] Q. W. Wei Tan, "Mine fire detection system based on wireless sensor network," in *International Conference on Information Acquisition*, 2007.

[32] F. Tian and X. Xu, "Design of wireless sensor networks node in coalmine," in *Second International Conference on Intelligent Computation Technology and Automation*, 2009.

[33] S. C. Mohammad reza Akhondi, Alex Talevski and S. Petersen, "Applications of wireless sensor networks in the oil,gas and resources industries," in *24th IEEE International Conference on Advanced Information Networking and Applications*, 2010.

[34] L. Chen, S. Yang, and Y. Xi, "Based on zigbee wireless sensor network the monitoring system design for chemical production process toxic and harmful gas," in *Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on*, 2010.

[35] D. Tianmin and S. Yao-yao, "Design of the intelligent public transportation monitoring system based on wsn," in *Consumer Electronics, Communications and Networks (CECNet), 2011 International Conference on*, 2011.

[36] T. Defeng, L. Shixing, X. Wujun, and Z. Yongming, "A fire monitoring system in zigbee wireless network," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2010 International Conference on*, 2010.

[37] L. Liu, "Research on environment-adaptive architecture model of wireless sensor networks," in *Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on*, 2010.

[38] R. Cheour, K. Lahmar, and M. Abid, "Evolution of wireless sensor networks and necessity of power management technique," in *Faible Tension Faible Consommation (FTFC), 2011*, 2011.

[39] H. Zhang, *Efficient data transport in wireless sensor networks*. PhD thesis, University of Adelaide, School of Computer Science, 2009.

[40] Libelium, "Waspmote," tech. rep., University of Zaragoza, Spain, March 2011.

[41] D. Gascon, "12km zigbee link with waspmote." Research Article, 2010.

[42] Libelium, "Meshlium," tech. rep., University of Zaragoza, Spain, 2011.

[43] Microchip, "Mcp9700/9700a/mcp9701/9701a," March 2009. MCP9700A Temperature Sensor Datasheet.

[44] L. Sencera Co., "808h5v5 humidity transmitter." 808H5V5 Humidity Sensor Datasheet.

[45] Figaro, "Figaro oxygen sensor sk-25f," October 2009. Figaro SK-25 Oxygen sensor datasheet.

[46] e2v Technologies, "Mics-2710 no2 sensor," July 2008. MiCS-2710 NO2 Sensor Datasheet.

[47] Figaro, "Tgs2600 air contaminants sensor," December 2005. TGS2600 technical datasheet.

[48] Figaro, "Tgs2602 air contaminants sensor," 2005 December. TGS2602 technical datasheet.

[49] e2v Technologies, "Mics-2610 o3 sensor," July 2008. MiCS-2610 Ozone gas sensor technical datasheet.

[50] Figaro, "Detection of carbon dioxide," December 2005. TGS 4161 Carbon Dioxide Sensor Datasheet.

[51] Figaro, "Detection of carbon monoxide," July 2007. TGS2442 Carbon Monoxide datasheet.

[52] Figaro, "Detection of ammonia," May 2007. TGS2444 Ammonia sensor datasheet.

[53] Figaro, "Detection of methane," February 2005. TGS2611 Methane sensor datasheet.

Appendix A

Thesis Contributions

A.1 Research Publications

1. **Amro Qandour**, D.Habibi, Senior Member, IEEE, and I.Ahmad, IEEE, Member, *"Application Framework for Wireless Sensor Networks "* (accepted), to appear in IEEE Conference Proceedings on Networking, Sensing and Control (ICNSC12)

2. **Amro Qandour**, D.Habibi, Senior Member, IEEE, and I.Ahmad, IEEE, Member*, "Applied Application of Sensor Networks in Underground Mines"* (accepted), to appear in IEEE Conference Proceedings on Networking, Sensing and Control (ICNSC12)

3. **Amro Qandour**, D.Habibi, Senior Member, IEEE, and I.Ahmad, IEEE, Member, *"Wireless Sensor Networks for Fire Emergency and Gas Detection "* (accepted), to appear in IEEE Conference Proceedings on Networking, Sensing and Control (ICNSC12)

A.2 Social Media

This appendix contains live-video demonstrations on how to use various functions of the Waspmote. I continuously maintain this support channel in order to provide educational videos for the Waspmote community.

Name	Description	Link
Sending RTC and ACC data to Waspmote	To demonstrate how to use the RTC and accelerometer modules in the Waspmote	http://www.youtube.com/watch?v=srZOiBC1fu4
GPS Demonstration using Waspmote	To demonstrate how to use the GPS module in the Waspmote to retrieve coordinates and NMEA sentence definitions	http://www.youtube.com/watch?v=rfNVJUn29EA
Waspmote Battery Interruption under a simulated power source	To demonstrate how to use the battery interrupt function of the Waspmote	http://www.youtube.com/watch?v=_Irk2LFUvfo
Running OTAP with Meshlium	To demonstrate how to use OTAP in Meshlium	http://www.youtube.com/watch?v=oBdjKXxuy44
Waspmote OTAP in Ubuntu 11.04	To demonstrate how to use OTAP in the Ubuntu OS	http://www.youtube.com/watch?v=yZe9cXJXxzY
RTC Waspmote	To demonstrate how to use the RTC module of the Waspmote	http://www.youtube.com/watch?v=U_o2xS5tpRY
Waspmote Over the Air Programming	To demonstrate how to use OTAP functions in the Waspmote-IDE	http://www.youtube.com/watch?v=XdSMQwJAkjE

Table A.1: Video demos

A.3 Media Coverage

Wifi sensors to tackle bushfires

Tuesday, 21 February 2012

A network of inexpensive fire monitoring sensors positioned throughout bushland could be the latest early-warning system in the fight against fires.

Researchers from ECU's Centre for Communications Engineering Research (CCER) have taken sensors and equipped them with wireless communication technology. Placed throughout an area of bush, the sensors take readings of temperature, oxygen, carbon dioxide levels as well as air contaminants.

When smoke from a fire is detected, the sensors can instantly alert fire authorities via wireless internet connection or a text message to a mobile phone, providing accurate GPS information.

Dr Iftekhar Ahmad and Mr Amro Qandour with Wifi bushfire monitor

Professor Daryoush Habibi, Dr Iftekhar Ahmad and Mr Amro Qandour developed the sensor network as part of the research they have been conducting into engineering applications that serve the local community.

"Bushfire is a major problem right across the nation. We wanted to serve our communities by finding a solution to the bushfire detection problem," Dr Ahmad said.

"The sensors have the capability to transmit data over long distances, allowing for large geographical areas or remote regions to be covered."

CCER are in the final phase of testing the devices and hope to team up with an industry partner to make the monitoring system available to fire authorities.

"Our priority is to have this system available for authorities. We hope to work with them in the near future to help protect our environment and human life from what can be devastating, and unfortunately, all too frequent occurrences," Dr Ahmad said.

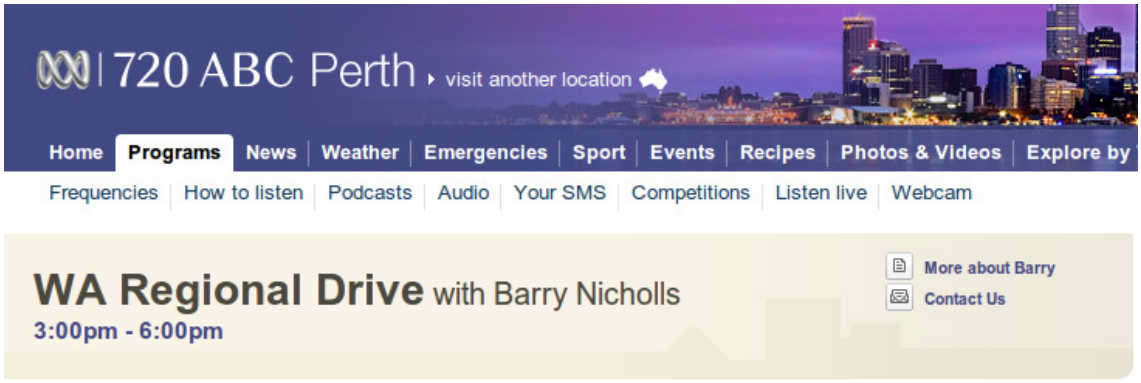
The CCER team have developed two other environmental monitoring systems that they also hope to make commercially available.

One monitors air quality within the home and identifies high levels of potentially fatal gases such as nitrous oxide, nitrogen dioxide, carbon monoxide, carbon dioxide and natural gas. Similar to the bushfire monitoring system, an alarm can be triggered or a text message sent.

The CCER team envisage that retirement villages could use the system to safeguard elderly residents against faulty heaters or gas stoves.

Another monitoring system measures the ultra violet (UV) index. This system could be utilised by surf lifesaving clubs across Australia, who could provide real-time UV readings on display screens or send the information as an alert to the mobile phones of beach goers.

"Feature article in ECU media release edition"



"Feature article on ABC news website (also live phone interview)"

ECU Engineering fighting regional fires, wirelessly

Imagine a summer morning in the near future, when you have decided to head for the beach. A few years ago, if you lived in an area of near-urban bushland, the threat of bushfires might have made this a risky decision. Today, thanks to research and development at ECU in 2011, you would have few concerns.

The technology now exists to establish a network of wireless-enabled monitors throughout fire-prone bush in semi-urban regions. Connected to the bushfire brigade and emergency services, the solar-powered monitors sample the air every 10 minutes and pick up any hint of bushfire smoke. Each monitor transmits its findings to a central database and, because they are all GPS identified, bushfire authorities can quickly spot and pin down fires before they spread.

This monitor is currently in the final stages of development at ECU's Centre for Communications Engineering Research (CCER) under the direction of Dr Iftekhar Ahmad and Professor Daryoush Habibi, and is just one of a number of wireless networked environmental sensors that will help to protect our environment and also protect humans from environmental dangers.

"Sensors are relatively cheap nowadays," Dr Ahmad says. "Most of the components are readily available, but they have to be combined in an optimum manner, calibrated for the intended use, and the communication protocols developed. Our intention is to provide society with cheap, reliable sensors that will monitor vital environmental factors in real time and make the information available wherever it's needed."

On the way to the beach, you might pass the house where your aged parents still live. You used to worry about them always forgetting to turn off their unflued gas heater. If your mother suffered from asthma, it could be exacerbated by exposure to nitrous dioxide from the heater. Or perhaps she had emphysema, and you worried that carbon monoxide could affect her respiration.

Now, however, they have installed a detector which measures both these gases in their home, as well as carbon dioxide and oxygen. The readings are taken every half hour, transmitted to the modem that also services the home computer and, if pre-set levels are exceeded, an alarm text will be sent automatically to your mobile phone.

This system also came out of ECU's CCER. "We can monitor almost any gas we want," Dr Ahmad says. "We intend to test our prototype in people's homes, perhaps in partnership with a local council, to ensure it gives us the results that warrant commercial production."

When you reach the beach, you are sensibly aware of the cumulative damage of ultra-violet light exposure and the risk of melanoma. Fortunately, another environmental monitor developed at ECU provides you with the answer.

A large digital clock dial on the lifesaving club's watchtower is connected to a real-time continuous UV monitor, so you know that right now, right here, the UV reading is still yellow and, so long as you wear sunglasses and sunscreen, you can enjoy the beach safely.

"There's a strong psychological element to having a real-time reading right on the beach," Dr Ahmad says. "As people see the UV index rising towards solar noon, they are more likely to take protective action than if they just checked a chart in the newspaper that morning. With Bluetooth capability the monitor could synch to mobiles within range, and beachgoers would get a warning even if they couldn't see the dial."

"Feature article in COHESION magazine (January 2012 edition)"

Appendix B

Source Code Library

B.1 Power Dissipation in Controlled Regulation

Setup Loop

```
void setup() {
  RTC.ON();
  RTC.setTime("01:01:01:01:01:00:00");
  RTC.setAlarm2("01:00:01", RTC_OFFSET, RTC_ALM2_MODE4);
  }
```

Main Loop

```
void loop() {
  PWR.sleep(ALL_OFF);
  if( intFlag & RTC_INT )    {
    USB.begin();

    SensorGas.setBoardMode(SENS_ON);delay(1000);
    getTemp(); delay(1000);
    getOxygen(); delay(1000);
    getCarbonD(); delay(1000);
    getFuel(); delay(1000);
    getNitrogenD(); delay(1000);
    getCarbon(); delay(1000);
    getHumidity(); delay(1000);
    getMethane(); delay(1000);

    Utils.float2String(PWR.getBatteryVolts(), Batt, 6);
```

```
    sprintf(data,"%s|%s%c%c", RTC.getTime(), Batt, '\r', '\n');
    USB.println(data); delay(1000);

    SD.ON(); delay(1000);
    writestate=SD.appendln("PWR.txt", data);
    SD.OFF();
    USB.println(writestate);

    xbee802.init(XBEE_802_15_4,FREQ2_4G,NORMAL);
    // Powers XBee
    xbee802.ON(); delay(1000);
    SendBS();

    xbee802.OFF();
    SensorGas.setBoardMode(SENS_OFF);
    intFlag &= ~(RTC_INT);
    RTC.clearAlarmFlag();
    RTC.setAlarm2("01:00:01", RTC_OFFSET, RTC_ALM2_MODE4);
  }
}
```

B.2 Power Dissipation in Non-Controlled Regulation

Variable Deceleration

```
//------------VARIABLES-----------
packetXBee* paq_sent;
float CH3 = 0;
float CO = 0;
float O2 = 0;
float CO2 = 0;
float NO2 = 0;
float Air1 = 0;
float Temp = 0;
float Hum = 0;
char data[100];
char Batt[50];
uint8_t writestate=0;
char gateway_data[100];
```

```
uint8_t saveSD=0;
```

Setup Loop

```
void setup() {
  USB.begin();
  xbee802.init(XBEE_802_15_4,FREQ2_4G,NORMAL);
  // Powers XBee
  xbee802.ON(); delay(1000);
  SD.ON(); delay(500);
  SD.create("PWR.txt");
  USB.println(SD.flag, DEC);
  SensorGas.setBoardMode(SENS_ON);
  RTC.ON();
  RTC.setTime("01:01:01:01:01:00:00");
  RTC.setAlarm2("01:00:01", RTC_OFFSET, RTC_ALM2_MODE4);
}
```

Main Loop

```
void loop() {
  if( intFlag & RTC_INT )    {
    getTemp(); delay(1000);
    getOxygen(); delay(1000);
    getCarbonD(); delay(1000);
    getFuel(); delay(1000);
    getNitrogenD(); delay(1000);
    getCarbon(); delay(1000);
    getHumidity(); delay(1000);
    getMethane(); delay(1000);
    Utils.float2String(PWR.getBatteryVolts(), Batt, 6);
    sprintf(data,"%s|%s%c%c", RTC.getTime(), Batt, '\r', '\n');
    USB.println(data); delay(1000);
    writestate=SD.appendln("PWR.txt", data);
    USB.println(writestate);
    SendBS();
    intFlag &= ~(RTC_INT);
    RTC.clearAlarmFlag();
    RTC.setAlarm2("01:00:01", RTC_OFFSET, RTC_ALM2_MODE4);
  }
}
```

B.3 Gas Sensors Functions

Temperature

```
void getTemp() {
  Temp = ((100*SensorGas.readValue(SENS_TEMPERATURE)) - 50);
  USB.print("Temperature="); USB.println(Temp);
  }
```

Oxygen Sensor

```
void getOxygen() {
  SensorGas.configureSensor(SENS_O2, 100);
  SensorGas.setSensorMode(SENS_ON, SENS_O2);
  delay(10000);
  O2 = SensorGas.readValue(SENS_O2);
  USB.print("O2= "); USB.println(O2);
  SensorGas.setSensorMode(SENS_OFF, SENS_O2);
  }
```

Carbon Monoxide Sensor

```
void getCarbonD() {
  SensorGas.configureSensor(SENS_CO2,1);
  SensorGas.setSensorMode(SENS_ON, SENS_CO2);
  delay(10000);
  CO2 = SensorGas.readValue(SENS_CO2) ;
  USB.print("CO2= "); USB.println(CO2);
  SensorGas.setSensorMode(SENS_OFF, SENS_CO2);
  }
```

Fuel Sensor Type-1 Sensor

```
void getFuel() {
  SensorGas.configureSensor(SENS_SOCKET2A,1 ,1);
  SensorGas.setSensorMode(SENS_ON, SENS_SOCKET2A);
  delay(10000);
  Air1 = SensorGas.readValue(SENS_SOCKET2A) ;
  USB.print("Air = "); USB.println(Air1);
  SensorGas.setSensorMode(SENS_OFF, SENS_SOCKET2A);
  }
```

Nitrogen Dioxide Sensor

```
void getNitrogenD() {
  SensorGas.configureSensor(SENS_SOCKET2B,1,11);
  SensorGas.setSensorMode(SENS_ON, SENS_SOCKET2B);
  delay(10000);
  NO2 = SensorGas.readValue(SENS_SOCKET2A) ;
  USB.print("NO2 = "); USB.println(NO2);
  SensorGas.setSensorMode(SENS_OFF, SENS_SOCKET2B);
  }
```

Humidity Sensor

```
void getHumidity() {
  Hum=SensorGas.readValue(SENS_HUMIDITY);
  USB.print("HUMIDITY = "); USB.println(NO2);
  }
```

Methane Sensor

```
void getMethane() {
  SensorGas.configureSensor(SENS_SOCKET4A,1,5);
  SensorGas.setSensorMode(SENS_ON, SENS_SOCKET4A);
  delay(10000);
  CH3 = SensorGas.readValue(SENS_SOCKET4A) ;
  USB.print("Methane = "); USB.println(CH3);
  SensorGas.setSensorMode(SENS_OFF, SENS_SOCKET4A);
  }
```

Carbon Monoxide Sensor

```
void getCarbon() {
  SensorGas.configureSensor(SENS_SOCKET3B, 1, 20);
  SensorGas.setSensorMode(SENS_ON, SENS_SOCKET3B);
  delay(10000);
  CO = SensorGas.readValue(SENS_SOCKET3B) ;
  USB.print("CO= "); USB.println(CO);
  SensorGas.setSensorMode(SENS_OFF, SENS_SOCKET3B);
  }
```

Appendix C

Hardware Devices

Figure C.1: Sensing/Routing node.

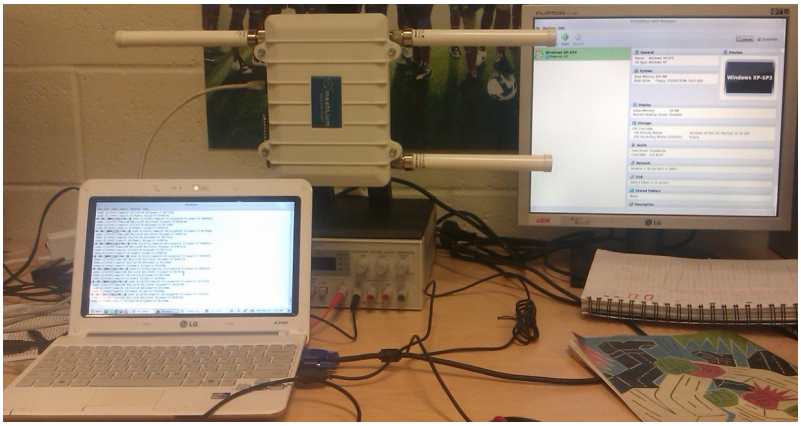


Figure C.2: The IP-basestation

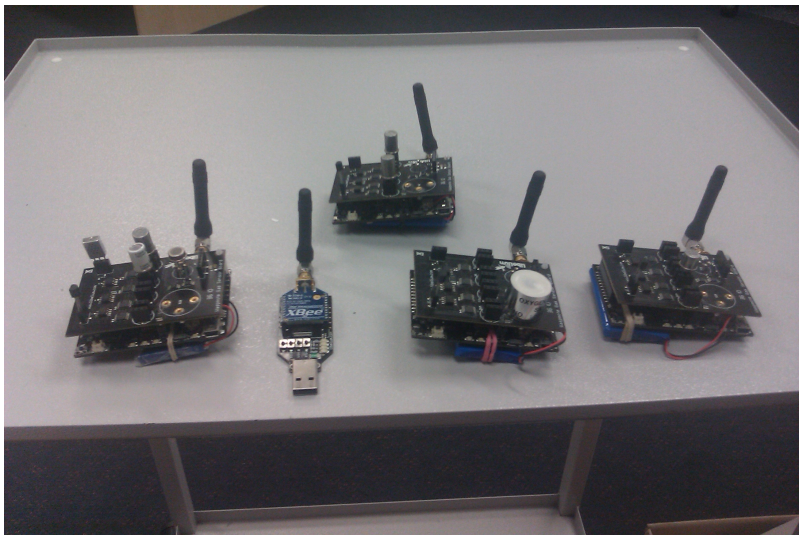


Figure C.3: Nodes with various gas sensors.

Figure C.4: Forest floor environment.

Figure C.5: Outdoor sensing/router node

Figure C.6: Sagemcom HiLo GPRS module.

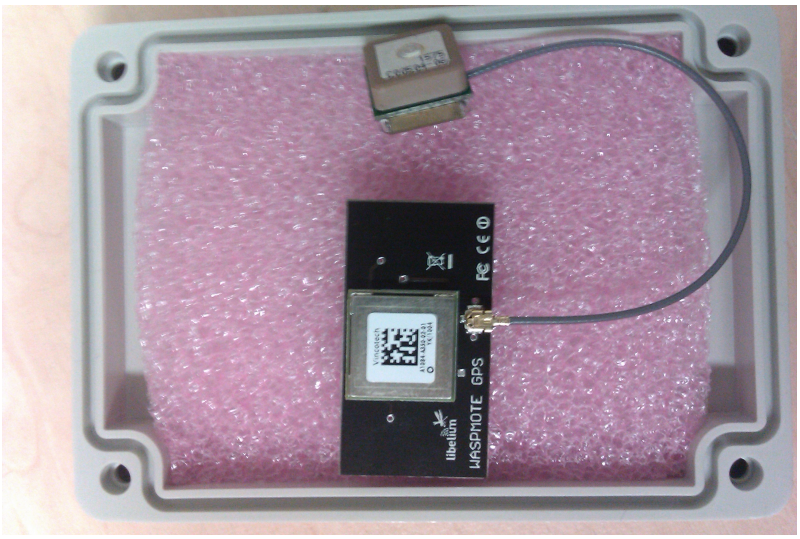


Figure C.7: Vicnotech A1048 GPS Receiver.

Appendix D

OTAP Troubleshooting

D.1 OTAP Setup

Sometimes using OTAP can be really frustrating. If you have problems using OTAP-shell, then this section can help you fix some of the issues of OTAP. These are the problems that you might encounter:

- When using the function scan_nodes and you don't receive response.
- Encountering a problem regarding the use of OTA with some functions from WASPSD.cpp library.
- Devices don't respond to OTA commands.

I have spent sometime on the OTA in Waspmote and I found a number of bugs in its operations. I found a number of solutions to address these problems of OTAP and it doesn't require any special technical abilities. Here is how you need to setup your nodes to use OTAP:

1. Waspmote Gateway: Ensure API-mode:1 is enabled.
2. Waspmote Node: It must use API-mode:2.
3. 2-GB micro-SD card must have a fat16 file-system (use gparted for windows vista plus).

Here are some common error messages and tips for troubleshooting these encounters:

1. "PROGRAM RECEIVED ERROR":
   (a) Initialize the SD card in the setup() loop of the OTA program and try again.
   (b) Remove all batteries including the smaller 3V battery at the back of the node and try again.
   (c) Reformat the SD-card and try again.
2. "Device doesn't respond":

(a) Ensure that your Waspmote is API2 and gateway is API1

(b) Again try the steps in 1.

Generally, you should try OTA with one Waspmote at a time, and sending smaller programs. When you are confident with its commands then you may start transmitting bigger programs.