WIRELESS SENSOR NETWORKS FOR ENVIRONMENTAL MONITORING APPLICATIONS

A thesis presented to the Department of

Computer Science

African University of Science and Technology

In Partial Fulfilment of the Requirements for the Degree of

Master of Science

Submitted by

Odo, Christian Maduabuchi

Abuja, Nigeria

August, 2016

WIRELESS SENSOR NETWORKS FOR ENVIRONMENTAL MONITORING APPLICATIONS

By

Odo, Christian Maduabuchi

A THESIS APPROVED BY THE COMPUTER SCIENCE DEPARTMENT

RECOMMENDED: _____

Prof. Ousmane Thiare

_____

Head, Department of Computer Science

APPROVED: _____

Chief Academic Officer

DECLARATION

This thesis has not been previously submitted for a degree in African University of Science and Technology and any other University. The research report contained in this thesis has been conducted by the author unless stated otherwise.

Signed: _____

ABSTRACT

After many years of rigorous research and development in wireless sensor network (WSN) technology with numerous responses to innovative applications, WSNs still have some interesting unanswered questions. In this thesis we explain the challenges of the state of art in WSN for environmental monitoring applications using open-source hardware platforms, Arduino UNO, DHT11 temperature-humidity sensor, XBee and Raspberry Pi. The system is not only low cost but scalable enough to accept multiple sensor nodes associated with environmental monitoring.

Leveraging on WSN & cloud technologies, we present the design and implement a cloud-based online real-time environmental data (temperature-humidity) collection platform to avoid memory conflicts of the BS that already has small storage capacity.

We implement a robust sensor node failure detector to enable user know the status of the network at every point in time under an online real-time basis. In this thesis we have detailed the overall construction architecture of the hardware and software design.

Some samples of the deployment and measurement data obtained are presented using various charts to validate the practicality of the system.

We emphasize the importance of having a generic system in which its ZigBee network configuration function set (router and coordinator) mode of data transmission will be implemented in API mode to accommodate any sensor node for better packet reception (RX) and transmission (TX).

ACKNOWLEDGMENT

I consider myself very lucky to have worked under Prof. Ousmane Thiare as my supervisor. It's quite a privilege to have my work thoroughly scrutinized with such an erudite scholar. I express my unreserved gratitude to him for being patient enough to read through my work, guide, encourage and willingness to discuss my problems.

I wish to thank Prof. Traore, HoD Computer Science, whose continuous seminar presentations has exposed me into the real world of research. Kudos to him.

I thank the management of National Institute for Legislative Studies, National Assembly, for sponsoring my tuition fee for this MSc. Program. And most especially the Director-General of the Institute, Dr. Ladi Hamalai for her unreserved kindness and permission to do this program. I appreciate sincerely Donald Mkpanam for his intellectual support.

I sincerely appreciate my dear wife Angela Adanne Maduabauchi for her moral support, prayers and encouragement. I thank Leaders of DHU ST. James Anglican Church, Gbazango Kubwa, Abuja, My mother Susan Ebam Ijeoma Odo and my siblings for their prayers and good will.

DEDICATION

I dedicate this work to God Almighty for showing me His mighty hand even when I thought that I may not be able to graduate.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

1.0 INTRODUCTION

The quest for a healthy environment and the survival of the human race has led researchers to undertake a fight against global warming and desert encroachment. This became imperative because they have a high negative impact on ecological life and the global economy. The resultant effect may be difficult to control if there are no perfect ways to determine their realistic rate of increase and moderation. In line with contemporary challenges in monitoring biodiversity and climate change, it has become imperative to mitigate the prevailing pressure on ecological systems. Due to the interwoven relationships between ecosystems and human activities, it is very significant to improve quality of life by using existing technologies to monitor biodiversity activities. In addressing the challenges that it imposes to ecological life, a very sensitive and reliable technology such as sensors is needed to obtain real-time data situation of a given environment. However, such dynamic sensors for data collection have to be linked up in a network for easy communication to a central data repository (base station, or BS) in order to check for redundancy and aggregation based on similarities to void error reading analysis. WSN can be used as platform to collect data and study the behaviour of ecological data.

WSN has a vital application like remote monitoring and target tracking [1]. It is usually characterized by a dense deployment and is large scale in environments limited in terms of resources [2]. The systems are powered with batteries and configured to carry out a processing capabilities like collection and storage of data and energy sensor optimization.

Energy limitation is a serious concern in the design of WSNs as it largely determines the success of network operations. Researchers over the years have applied WSNs in various fields, like military surveillance, energy management, earthquake detection and ecological data mining. But for effective environmental monitoring, guaranteed quality of service (QoS) and fast accessibility of real-time online data in WSNs are still open questions.

1.1 STATEMENT OF PROBLEM

The need to access real-time ecological data remotely cannot be overemphasized. Consequently, considering the limited storage space of nodes in WSNs, the traditional ways of gathering high volumes of biodiversity data and storage of such data are ineffective. The current state of the art in real-time biodiversity data collection methods fails to incorporate real-time synchronization of data to a cloud service. Due to the limited storage capacity over time as data grows, the mode of data visualization, management and analysis of data become complex and easily prone to errors. Failure detection and fault tolerance are currently not implemented and information on failed sensor nodes is not available remotely.

1.2 GOAL

The aim of this project is to set up a robust WSN for Environmental Monitoring Application of ecological data (biodiversity).

1.3 SPECIFIC OBJECTIVES

i. Set up a cloud storage service for collection of real-time biodiversity data (temperature and humidity) and synchronize the data between the BS and cloud infrastructure.
ii. Design and implement a communication protocol for collection of tracking information.
iii. To explore the techniques of data collection and aggregation based on similarity and redundancy of information.
iv. To implement a remote web application service for monitoring of real-time sensor node failure.

CHAPTER 2

CONCEPTUAL ANALYSIS

2.1 WIRELESS SENSOR NETWORK

WSN application has grown widely in the last decade. Many industries, researchers and engineers work on this novel technology by applying it in civilian and military sectors that use many nodes. Many applications have been developed and deployed for the purpose of solving numerous problems in data acquisition and environment control. These node networks must have the capacity, via universal wireless, of sensing, processing and communicating physical parameters such as pressure and temperature [3].

A WSN is described as networked nodes that jointly sense and manipulate data, enabling interaction between humans or computers and the immediate environment (Bröring in [4]). WSNs today include sensor nodes, actuator nodes, gateways (Internet connectivity) and clients. It also entails a large number of sensor nodes randomly deployed indoors or near the monitoring area (sensor field), which form networks through self-organization. Operationally, sensor nodes observe the data collected and transmit to other sensor nodes by hopping. For the period of transmission process, sensor data may be moved by multiple nodes to get to a gateway node after multi-hop routing, and eventually end with the management node through the Internet or satellite as the case may be. The user configures and manages the WSN with the management node, broadcasts monitoring jobs and collects the monitored data [5].

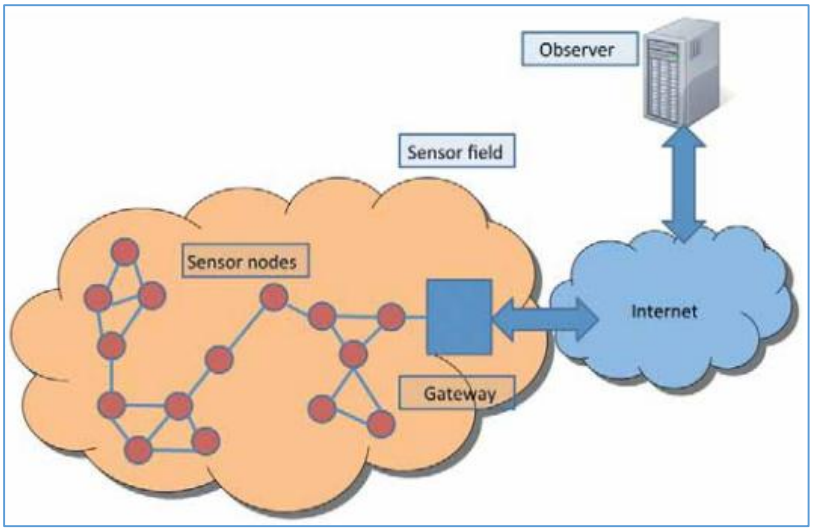


Figure 2.1: Wireless Sensor Network Mesh Network

Image source: http://www.iec.ch/whitepaper/pdf/iecWP-internetofthings-LR-en.pdf

WSNs are critically resource inhibited by limited power supply, memory, performance and communication bandwidth [6]. While initial sensor nodes were resource-constrained with limited capabilities, recent improvements in sensor hardware technology have made it possible to have better processing power sensors, memory and protracted battery life [7]. But Mohammad in [8] cites Heinzelman and states that the energy consumption of radio communication largely depends on the number of bits of data that can be transmitted within the network.

Fischione in [9] states that WSNs make the Internet of Things (IoT) possible and defines it as a networked wireless system for computing, transferring and receiving nodes meant for interaction, controlling, sense and activation.

He further stated the following as characteristics of WSNs:

- Battery-operated nodes;
- Short range wireless communication;
- Mobility of nodes; and
- No/limited central manager.

2.2 TYPES OF WSN

Ado et al. in [10] list diverse types of WSN as follows:

2.2.1 TERRESTRIAL

This is largely used in the field of environmental monitoring and poses a challenge to the sustainability of the network in terms of energy management [11].

2.2.2 UNDERGROUND

Sensors are buried underground using wireless technology to enable them to communicate. It is used for agricultural purposes to monitor conditions in the soil. It has a land node to transfer sensed information from the underground nodes to the BS [12]- [13].

2.2.3 UNDERWATER

Ado et al. cited by Potdar et al. [10] explain that this type of WSN still imposes serious research challenges due to the hostility of the deployed environment to the nodes which are usually meant for exploration. Here there is recurrent loss of signal, propagation delays and synchronization problems are high.

2.2.4 MULTIMEDIA

This monitors real-time data like images, videos and audio. The sensors use camera and microphones. The problem here is high energy consumption according to Misra in Ado et al. [10].

2.2.4 MOBILE

The mobile nodes have the ability to autonomously reorganize the network and communicate with the physical environment. This network is more flexible than the static sensor networks because it has the ability to improve coverage, energy efficiency, superior channel capacity, and so on [14].

2.3 MOBILE AD HOC NETWORK

A mobile ad hoc network (MANET) is set of wireless mobile systems where nodes act as a team by forwarding packets for each other to ensure communication beyond the wireless transmission field. A MANET allows wireless devices to inter-communicate autonomously. It does not rely on the BS to channel messages to nodes in the network. It has the problem of maintaining data packets' traffic route [15].

2.4 WSN VERSUS MANET

WSNs are similar to MANET in a number of ways. WSNs create networks with interconnected sensor nodes, in an ad hoc form and no formal infrastructural set up is required for either. WSNs collect data through the nodes while MANET does not always use sensor nodes [16]. The role of MANET is purely for communication and not for monitoring and data collection. The mode of communication is via radio waves. The nodes communicate directly to those nodes that are in radio range (by electromagnetic wave). However, other nodes require assistance of intermediate nodes to route their packets [15].

MANET is a kind of ad hoc network with peculiar features like open boundary network, distributed network, changing topology, speedy implementation and hop-by-hop communications. The above characteristics of MANET make it widespread, especially in military and disaster management applications [17].

2.5 SENSOR NODES

A sensor network is a system of nodes where each sensor node is equipped with a radio transceiver along with an antenna, a microcontroller, an interfacing electronic circuit and an energy source (battery). The sensor node is one of the main parts of a WSN. Each sensor node forms a small computer consisting of a processing unit and a limited amount of computational power and memory [16]. Ado et al. see a wireless sensor as a true embedded system with a wireless communication function that can collect physical quantities like heat, humidity,

temperature, vibration, radiation, sound, light and movement and convert them into digital values which are sent as sensed data to a BS [10].

2.5.1 SENSOR NODES MECHANISM

The hardware of a sensor node normally comprises four parts: the power and power management module, a sensor, a microcontroller and a wireless transceiver. The power module supplies the reliable power required for the system. The sensor is the core of a WSN node which can obtain the environmental and equipment status. Sensors are in charge of collecting and transforming the signals, like light, vibration and chemical signals, into electrical signals and then transferring them to the microcontroller. The microcontroller collects the data from the sensor and processes the data as required. The wireless transceiver (RF module) then transfers the data, so that the physical realization of communication can be achieved [16].

Ado et al. in [10] classify two types of sensors: universal sensors and gateway sensors. A generic sensor has a role of collecting measures from the deployment area while the gateway sensor has further capacity in terms of computing resources, storage and transmission.

2.6 TCP/IP

TCP/IP is the suite of communications protocols used to join hosts on the Internet. TCP/IP uses numerous protocols, the two main ones being TCP and IP. TCP/IP is the de facto standard for communicating data over networks. It was built into the UNIX operating system and is used by the Internet, though other network operating systems that have their own protocols, like Netware, which also support TCP/IP [18].

The Transmission Control Protocol (TCP) is one of the core protocols of the Internet Protocol Suite. TCP is one of the two novel elements of the suite, perfecting the internet Protocol (IP), and so the whole suite is commonly referred to as TCP/IP. TCP provides stable, ordered transmission of a stream of octets from different programs on diverse computers. It is the protocol used by main Internet applications like the WWW (World Wide Web), email, remote administration and file transmission [19].

In furtherance of its capability as cited in [20] TCP/IP is an industry standard suite of protocols that provides communication in a heterogeneous environment. It offers a routable, enterprise networking protocol and access to the Internet and its resources. In general view according to [21], TCP has following features: connection-oriented, error detection and correction (reliable), full-duplex connection, provides a 'byte pipe', sliding window protocol etc.

2.7 DATA AGGREGATION

Data aggregation is a way of incorporating multiple copies of information into one copy that is effective and able to meet user requirements in middle sensor nodes. It saves energy and obtains accurate information. In sensor networks energy consumption is higher in transmitting data than in processing the data. Therefore, node computing and storage capacity in data aggregating operations helps to reduce large quantities of redundant information, and minimize the rate of data transmission to save energy [16].

The main concern of data aggregation is to increase the network lifetime by reducing the resource consumption of sensor nodes (like battery energy and bandwidth). In a further assertion [22] data aggregation was explained as a system that usually involves the fusion of data from multiple sensors at intermediate nodes and transmission of the aggregated data to the BS (sink). Compared to conventional networks, sensor networks have higher data loss rates. Data aggregation could reduce data redundancy but has a high risk of more loss of information, which reduces the robustness of the sensor network [16]. But Sumedha and Samarth argue in [23] that redundancy sustains reliability and state the need to maintain redundancy, but it should be at an adequate level.

In a related development, data aggregation is usually achieved by a cluster head and consists of collecting data from all cluster members, applying an aggregation function to all collected data and transmitting a single value as measured data [10].

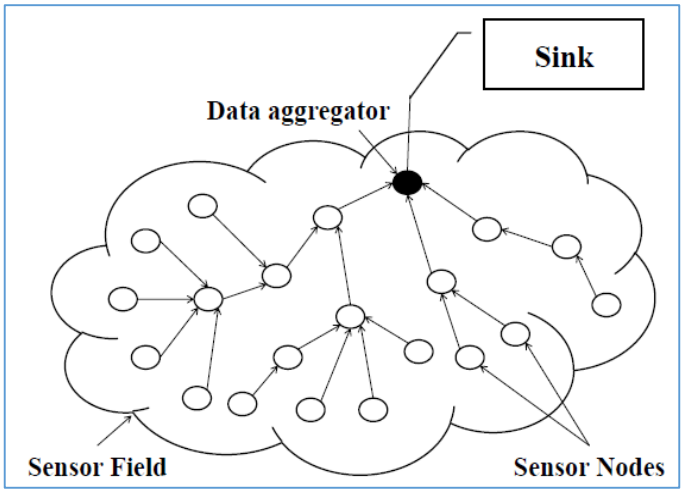


Figure 2.2: Data Aggregation

Image source: [22].

2.8 DATA AGGREGATION TECHNIQUES

There are many techniques of data aggregation for the purpose of handling data redundancy and energy management. The essence of aggregative knowledge is to transfer computed data to the sink node by choosing the most economical path. In [24] the techniques are categorized as Centralized Approach, In-Network Aggregation, Tree-Based Approach and Cluster-Based Approach.

2.9 CLUSTERING

Clustering is a technique for decreasing data collection costs in WSNs. Here, sensor nodes are gathered into separate sets, with each set managed by a designated cluster head (CH), chosen from among the sensor nodes. The cluster members send their collected observations (highly correlated) to their CH. The CH overcomes the local redundancies and transfers the compressed data to the sink by means of multi-hop transmission [22]. Clustering is a standard approach for reducing energy consumption and traffic load. Normally a cluster member keeps complete state information about all other members in its cluster [25]. Below are some of the techniques of data aggregation.

2.9.1 CENTRALIZED APPROACH

Here each node sends a packet to a central node via the shortest available route. All the sensor nodes send data packets to a node regarded as the most powerful, otherwise known as a header. The header does the aggregation and sends the single packet to a BS or a remote base [23].

2.9.2 IN-AGGREGATION

In a multi-hop network this is a global process of data gathering and routing of information that computes data at intermediate nodes for the purpose of reducing power consumption. Sumedha and Samarth in [23] explained that the process are in two ways such as **With Reduced Size**, where the packets received are compressed and merged as a single packet before transferring to base state and **Without Reduced Size,** which is collection of all data packets and merging them as they are received without reduction of the packet size.

2.9.3 TREE-BASED APPROACH

In [23] it was discovered that at first, Data Aggregation Tree (DAT) is formed to easily data collect flow. And for each data aggregation, a minimum spanning tree is created as a network. Ameya et al. describe how the flow of information starts from leaves nodes up to the sink and thus, the aggregation is performed by parent nodes [24].

2.10 802.15.4 IEEE STANDARD

IEEE 802.15.4 is a standard for wireless communication. The standard was designed for point-to-point and star communication at over-the-air baud rate [26]. The standard defines the "physical layer" and the "medium access layer". The specification for the physical layer, or PHY, defines a low-power spread spectrum radio operating at 2.4 GHz with a basic bit rate of 250 kbit/s [27].

2.11 ZIGBEE SUITE

ZigBee technology is built on IEEE standards with specification of 802.15. It functions on IEEE 802.15.4 physical radio with unlicensed radio frequency bands involving 2.4 GHz, 900 MHz and 868 MHz. Three ZigBee specifications include ZigBee, ZigBee IP and ZigBee RF4CE. The ZigBee IP

operates on IPV6 standards for full mesh networks while ZigBee RF4CE uses partial mesh networks [28]. The technology was built to ensure low data rate, low energy usage, minimal cost. The wireless networking protocol was meant for application of remote monitoring and automation. The commercial classification of ZigBee is known as wireless control. Data broadcast rates is between 20 to 900 kbps [29]. The differences between ZigBee technologies include but are not limited to XBee and XBee Pro module. ZigBee is mostly applied in power plant operation and industrial automation and is usually linked with IoT and machine-machine (M2M) communication [30].

Although many wireless standards exist, ZigBee has a unique focus. The technology was not meant to compete with 802.11 Bluetooth standards but enhanced to ensure sensing applications and remote control [31].

2.12 XBEE

XBee is compatible radio modules from Digi International. The first XBee radios were introduced under the MaxStream brand in 2005 [32] and were based on the IEEE 802.15.4-2003 but the publication of IEEE 802.15.4-2006 supersedes the earlier. It is worth noting that ZigBee as a protocol uses the 802.15.4 standard as a baseline and enhances additional routing and networking functionality. The protocol was established by the ZigBee Alliance. The companies worked in cooperation to develop a network protocol that can be used in a diversity of commercial and industrial low data rate applications. The design added mesh networking to the underlying 802.15.4 radio [28].

Two models were initially introduced – a lower-cost 1 mW XBee and the higher-power 100 mW XBee-Pro [26]. Since the initial introduction, a number of new XBee radios have been introduced and all XBees are now marketed and sold under the Digi brand. The low-power, reliable wireless network produced by ZigBee standard allows sensor data to be delivered reliably and constantly to an Arduino system. The current state of art of wireless connection uses ZigBee instead of Bluetooth because of its long covering range, low cost, mobility and reliability.

2.12.1 XBEE MODULE

This has the tendency of making a complex mesh network structure independently without involvement of user application program that runs on the microcontroller or microprocessor platform. The technology has significantly reduced the complexity of WSN system development. It forms mesh network topology using ZigBee networking protocols and easily integrated into Arduino and Raspberry Pi via the USB communication interface.

The modules are configured into three types of devices:

i. Coordinator;
ii. Router; and
iii. End device.

The XBee module on the BS works as coordinator and the XBee modules on the sensor nodes as routers [33]. However, for the purpose of this research XBee was adopted as a wireless platform to transmit and receive data between Arduino and Raspberry Pi as the BS.

2.13 ARDUINO

Brad in [34] states that Arduino boards are not full computers but micro-controllers. He states that this board does not run a full operating system but executes written code as their firmware interprets it. However, Sheikh and Xinrong in [33] explain that Arduino is a widely used open-source single-board microcontroller development platform with flexible, easy-to-use hardware and software components. Arduino UNO R3 is based on an Atmel Atmega328 microcontroller and has a clock speed of 16 MHz. It has six analog inputs and 14 digital I/O pins, so it is capable of connecting some sensors to a single Arduino board. The shield can directly be plugged into the standardized pin-header of the Arduino UNO board if developed.

2.14 RASPBERRY PI B

Raspberry Pi Model is low-power credit-card-sized single-board computer. The CPU is an ARM processor with 700 MHz clock speed. CPU performance is comparable to a Pentium II

300 MHz processor and the GPU performance works in line with the original Xbox. It has a variety of peripheral interfaces like a USB port, HDMI port, 512 MB RAM, secure digital (SD) card storage and 8 GPIO ports for expansion. Raspberry Pi is as good as a desktop computer that uses HDMI and USB connectors to connect IO devices like a monitor and keyboard. It supports operating systems like Debian-based Linux distro, Raspbian. Raspberry Pi has an Ethernet port and can connect to a local network via cable or USB Wi-Fi adapter, and can also be accessed via SSH remote login [33].

2.15 FAULT TOLERANCE

Sushel cites Vallée et and describes fault tolerance as one of the metrics which is considered to be most significant since the resource failure affects throughput, response time, job implementation and system performance. He adds that a fault tolerance policy is necessary to detect failures, resolve failures and also recover performance metrics. In a further claim the papers state that fault tolerance as a major concern, which guarantees availability of critical services and application execution. These are measures to avoid or recover from failures.

- **Reactive fault tolerance**: Reactive fault tolerance policies minimize the effect of failures on application execution when it occurs.
- **Proactive fault tolerance**: Proactive fault tolerance policies try to pre-empt recovery from faults, errors and failures by using predicting techniques and proactively replacing the suspected mechanisms with other working components [35].

Sani and Jeong [36] explain various reasons for sensor node failures as radio interference, battery failure and synchronous and asynchronous communication mechanisms. These failures mostly occur due to software or hardware faults, malicious code, environmental changes and timing closure. These result in an event where a sensor node turns out to be inaccessible or disrupts certain operating settings that are critical for providing the required service. Communication in sensor networks is highly critical because the links in a network are prone to faults and errors. For example, the collision of network packets may result in a failure from source to destination.

Such a failure in a routing protocol leads to sizeable network packet drops resulting in excessive network delays and consumption of a considerable amount of energy [36].

2.16 ECOLOGICAL SYSTEMS/BIODIVERSITY

Biodiversity is a complicated concept that largely refers to the multiplicity and variability of living organisms and the ecological complexes in which they happen. The concept consists of, but is not limited to, microscopic life, fungi, plants and animals. Biodiversity is also indispensable for living as it affects virtually every aspect of human existence. The importance of biological diversity extends to healthy, functioning natural systems, and fulfilling fundamental aesthetic values [37].

Ecosystem diversity is described as the variety of ecosystems in a given area. Comprehensively it is total interactions that involve the physical, biological, chemical environment and the resulting ecological practices [37]. However, this research is focused on bio-physical interaction (temperature and humidity) in an ecosystem for effective environmental monitoring using WSN.

2.17 REMOTE MONITORING

The terms remote monitoring (RM) and remote interrogation (RI) are often used interchangeably; however, RM is the colloquially accepted term for both. RM refers to the automated transmission of data based on pre-defined signals related to device functionality [38]. RM is project specific as different things could be monitored.

More broadly, Ahmad et al. cited Nhivekar et al. in [39] describe real-time data monitoring as an important support application which could be applied to observe supported electrical device conditions and other environmental biodiversity such as temperature, humidity, voltage, current and wind conditions. Internet-embedded technology makes data transfer and accessibility globally possible using devices to communicate with computers in performing its operation [40]. The idea of wireless data transmission is to provide device simplicity instead of wired systems and lower cost for long range communication [41]. Frequent human site visits are not advisable due some factors like safety, rough environment, high cost, harsh weather and wildlife risk. To avoid this unwarranted risk, reliable long range wireless monitoring systems, with simple

maintenance are required [42]. The IoT era, has made it possible for sensor data sampling to be real-time, online and accessed anywhere with an Internet connection [43]. Current technology makes it possible to establish unmanned monitoring in order to overcome the above difficulties. The real-time monitoring system helps to minimize frequent human site visits for configuration and system maintenance [39].

2.18 CLOUD COMPUTING

The concept of cloud computing came into existence in the 1950s with implementation of mainframe computers, accessible via thin/static clients. Since then, cloud computing has evolved from static clients to dynamic ones from software to services.

Cloud computing does not have a common accepted definition [44]. Cloud is a platform that makes it possible for information to be accessed from any location at any time. Traditionally to access data in a computer system, the user is required be in the same location as the data storage device, but the introduction of cloud services has eliminated this [45].

Bodkh et al. [46] define cloud computing as a centralized pool of resources and computing outsourcing tools for different computing services to different people as utility-based systems [46]. NIST elaborated the scope and explains it as a model for enabling universal, convenient, on-demand network access like servers, storage, applications, and services [47].

2.18.1 SERVICE MODELS

Cloud computing architecture consists of three layers [48]:

- Software as a Service (SaaS): In this model, a complete application is offered to the customer, as a service on demand. A single instance of the service runs on the cloud and many end users are serviced. Users are not expected to locally buy servers or software licences but pay for the services whose costs are mostly negligible. Alex and James in [45] explain that with SaaS it is unnecessary to have a physical copy of software to install on your devices.

- Platform as a Service (PaaS): A development environment is offered as a service. The subscriber can build customized applications that will run on the provider's infrastructure. To ensure manageability and scalability requirements of the applications, some OS and application servers like LAMP (Linux, Apache, MySQL and PHP), restricted J2EE and Ruby are needed.

In [49] the provider gives end users access to the components that are required to develop and operate applications via the Internet.

- Infrastructure as a Service (IaaS): This is the most rudimentary level of service. Here each of the service models comply with the security and management mechanism from the underlying model, as shown in the following diagram [50]:

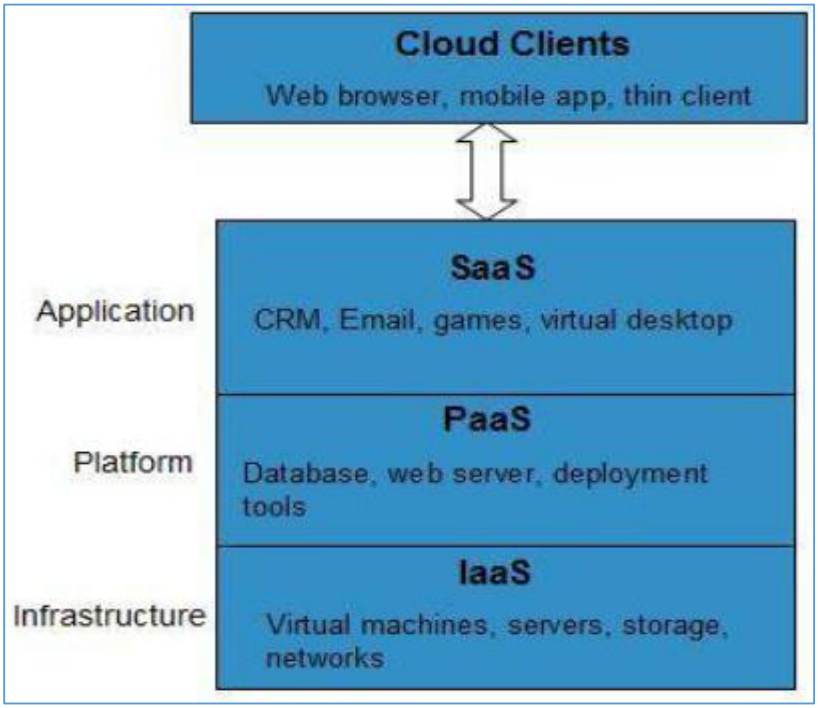


Figure 2.3: Cloud Services

Harris and Terry in [48] emphasize that IaaS delivers basic storage and computational capabilities as homogeneous services over the network. Servers, storage systems, networking equipment,

data centre space etc. are provided to handle workloads. Examples of these providers include Amazon, GoGrid, 3 Tera, etc.

2.19 EMPIRICAL STUDIES

This section encompasses difference approaches (solutions) researchers have adopted to solve issues of environmental monitoring using WSNs and future focus to combat the current challenges.

2.20 ENVIRONMENTAL MONITORING

Environmental monitoring is one of the most important applications of WSNs. It ordinarily requires a lifetime of several months, or even years. In line with this, the inherent constraint of battery energy of sensor nodes makes it very difficult to have an adequate network lifetime. In reality there turns out to be a bottleneck in scale of any application in WSNs [51]. Mou et al. [51] assert that contemporary climate change and natural disasters globally have demonstrated the importance of environmental monitoring, which has progressively developed as a foremost application of WSNs. For example, within a WSN sensor nodes today can be used in a harsh environment to sporadically measure meteorological and hydrological parameters in their surroundings, like light, temperature, humidity, wind speed and direction. These are done by using advanced wireless communication and sensor technology.

2.21 DATA PREDICTION, COMPRESSION AND RECOVERY IN CLUSTERED WSN.

Mou et al. in [51] applied the OSSLMS and Principal Component Analysis (PCA) data prediction techniques for collection of environmental data and at the same time managed energy of the sensors by avoiding overhead high data communication computations. In their research, the focus was on minimizing the amount of continuous transmitted data from the sensor nodes to the BS during the monitoring process. The fundamental issue in their research work was to ensure the accuracy of the prediction within a user-given error range. Considering the periodical data sensing that must be collected in the environmental monitoring, constant observation of a sensor node was temporally correlated to a certain point. To avoid the high cost of the data

communication process, the above prediction model was dually applied, and temporal correlation was adopted to perform prediction of data instead of periodical collection of real-time data that would have increased energy use and reduced sensor battery life. These were considered and applied to enhance efficiency and accuracy of data for monitoring the application based on the accepted error tolerance.

The outcome of applying this correlation-based approach of dual prediction protocol that has an outstanding effect was to reduce the frequency of data transmissions in such a way to guarantee the prediction accuracy. Another efficient method in handling reduction was by applying compression techniques [52]- [53] that causes a reduction in the volume of transmitted data by reducing the size of the original data.

In their work, data compression schemes were classified into two categories: Lossless and lossy compression.

Lossless data compression perfectly reconstructs original data from the compressed data while lossy data compression gives room for some of the original features to be lost after decompression operation. However, research has shown that lossless algorithms are not always necessary despite the fact that they have better performance in recovery of data in a highly constrained resource in a WSN. They found that in terms of transmission of data in a WSN lossy compression performs better in reducing size of data to be sent via the network. In their argument, to judge the quality of compression algorithms the amount of data compression and the construction error are very significant criteria to consider.

Mou et al. in [51] proved successfully that using the PCA method to compress the original data gives satisfactory results in two ways. Comparative to the prediction error, which ensures the user's accept total error bound, the work explains that the error generated by the PCA compression is negligible. Since the research also focused on energy efficiency scheme for uninterrupted environmental monitoring, they developed a novel framework with a delicate combination of data prediction, compression and recovery in cluster-based WSNs.

The foremost idea of their framework was to minimize the overhead communication cost via data prediction and compression techniques, at the same time guaranteeing accuracy.

They grouped sensor nodes for collecting environmental parameters into multiple clusters based on their physical locations. Due to the limitation of a single algorithm to implement this, a dual prediction mechanism was applied using an LMS prediction algorithm with optimal step size at sensor nodes and their respective CHs, the research proved that prediction accuracy was highly satisfactory and there was also faster convergence speed. Here the CHs extracted the principal component of composed data by the PCA techniques after a sampling period, and redundant data was prevented.

In conclusion, data was successfully recovered at the BS. Through the entire process, all errors were well-regulated and kept within tolerable bounds. When data reduction was achieved, the size of recovered data at the BS was equal to the original sensory data collected by all nodes. The work showed how beneficial it was for the BS to gain a more comprehensive understanding of environmental parameters. The results validate that the use of LMS prediction algorithm and PCA technique is very efficient in energy management for environmental monitoring applications in cluster-based WSNs.

2.21.1 CHALLENGES

There is need to integrated the scheme closely with clustering algorithms to further minimize the communication overhead and increase the error accuracy of data prediction and recovery. More so, the work did not take cognizance of the outliers' detection mechanism to oversee the unexpected change in local environment and unreliable measurement. Issues of non-synchronization, failure and data loss were not solved.

2.22 BIODIVERSITY FOR ENVIRONMENTAL MONITORING

Sheikh and Xinrong in [33] implemented a WSN for Environmental Monitoring Application to collect temperature and humidity data using open-system hardware/software: Raspberry Pi, Arduino UNO R3, ZigBee Module, XBee Pro S2B and RTH03 temperature and humidity sensor.

For this research, their work happens to be the latest biodiversity research that have implemented temperature and humidity data collection using the above cheapest technology with remote access. However, the system lacks the capacity to implement fault tolerance (sensor node failure detection) and a cloud-based synchronized storage system (second database server) for long-time data gathering, owing to the limited memory of Raspberry Pi.

They were able to achieve the following: low cost, resource compatibility, scalability, and ease of customization, deployment and maintenance. One major improvement of the design lies in the integration of the gateway node of WSN, database server and web server into one single compact, low-power, credit-card-sized computer Raspberry Pi, which made it easy to be configured to run headless (i.e. without monitor, keyboard and mouse). This type of design is convenient in implementing environmental monitoring and data collection applications.

2.22.1 CHALLENGES:

The system lacks the sensing modality to report on real-time node failure, data visualization, information management and proper data analysis. The system resides on a private intranet IP and lacks the ability to synchronized real-time data to a second storage server. Hence there is a need for a cloud storage server for data synchronization due to the limited storage space on the Raspberry Pi which make it impossible for the system to store large volumes of data.

2.23 WSN FOR BOREHOLE MONITORING:

Kyukwang and Hyun [54] in a research of an RM of the presence of coliforms in water sources or food factories using sensor nodes are said have achieved some breakthroughs using Arduino and Intel® Edison on each node to capture live video streaming and web-based motor control UIs on an internal server via a Wi-Fi network and to control self-priming water pumps to sample the water, mix the reagent and remove the water sample after the test is completed. The sensor node repeats water testing until the test reagent depletes. The essence of the research was to reduce the cost and labour for testing samples in a factory and checking the water quality of local water sources in developing countries [54]. However, the challenges it had were (i) flexibility on

wireless connectivity, (ii) a sustainable thermal control method, (iii) self-test functions, and (iv) failure recovery and an easy UI.

2.23 AUTOMATIC REMOTE METER READING SYSTEM

The traditional method of distributing electric bills at month end is very slow and time consuming considering the small space and the noise in an urban environments. In a further development, use of cable systems or SMS-based GSM usage were introduced but suffered high costs and were difficult to install in an area where population density was high. Bushra [55] cited in Jingjing Wang proposed using GPRS-based wireless as an ad hoc network in order to acquire meter information remotely. Each node in a wireless ad hoc network was proposed to consist of an ammeter, microprocessor card which incorporates ammeter readings into percentage usage and a GPRS transmitter for communication on the network. GPRS was better for communication and low in installation and maintenance cost with a high data transmission rates.

In additional research Bushra, cited Liting Cao et al. in [55], where a proposal was made on an alternative way to automatically acquire meter readings by using the ZigBee technology Instead of GPRS wireless communication. The ISM band based ZigBee technology is used for communication purposes sandwiched between nodes for data collection. Communications here are between sensor nodes and data collector using the ZigBee technology while connection between data collector and server is through the Internet.

A diverse WSN for power electricity meter monitoring called EMMNet was designed in Lin et al. as in [55] with set of standards and guidelines reserved while designing a solution for efficient meter reading monitoring. The novel approach was especially designed for urban environments.

2.24 HEALTH CARE REMOTE MONITORING APPLICATION.

A system for displaying, storing and sending patient's physiological data was developed by Amna et al. in [56]. The system's Internet connectivity helps healthcare professionals to monitor and access patients' data remotely online in real time. Biomedical sensors were inserted into patients to transform the monitored physiological quantities, like temperature, blood-pressure, heartbeat

rate, muscles, ECG and glucose level, into electronic data for accurate measurement and recording.

Ahmad et al. in [39] implemented a system for RM of an electrical device via web-based application. Their monitoring system allows users to monitor device status from anywhere due to its ability to synchronize information on a website. The electric current and voltage readings of the monitored equipment such as ambient temperature and humidity level are stored and displayed on a web page. All sensors used were connected to the microcontroller and the data saved on a micro SD card and information transferred to a web page using the GPRS service connection synchronously. The collected data is transferred to a website for accessibility. The essence was to enable users to monitor device status and ambient changes periodically. This project was implemented in Selangor, Malaysia.

Although the project was interesting, it lacked the ability to store large volumes of data due to limited memory space on the digital SD card.

2.25 SUMMARY

WSN in the 21st century is a very important technology [57] and has attracted attention globally by its ability to solve different problems such as military surveillance, forest monitoring, wild life tracking, precision agricultural, rainfall, medical and health care, traffic monitoring, transportation, water pipeline, intruder detection, environmental monitoring – temperature, humidity (soil, leaf, ambient), soil moisture, wind (speed and direction), pressure, leaf, PH, redox, underground water level, and so on.

However, it is important to note that WSN is solution specific, meaning that it is applied in a given environment to solve a specified task. From the papers reviewed it is quite exciting that WSN has various achievements as regards environmental monitoring applications but many challenges and limitations are encountered in diverse implemented projects. Some of the major key challenges ravaging many projects were seen as sensor node failure detection (fault tolerance)

and limited storage space to store large volumes of data as the current state of art of a Raspberry Pi device (as BS) could only store small data.

It is based on these that this research has undertaken to implement a robust heterogeneous network system, good data aggregated system, less cost cloud storage server integration to synchronize and store continuous real-time online biodiversity big data for effective environmental monitoring and data analysis.

CHAPTER 3

3.0 METHODOLOGY (ANALYSIS AND SOLUTION DESIGN)

These topics are discussed: design of the study, area of the study, instrumentation for data collection and method of data transmission.

The challenging factors in WSN for Environmental Monitoring Application using Raspberry Pi, Arduino and XBee DTH11 temperature-humidity sensors as discussed in the previous chapter, have posed a lot of difficulties especially as regards online real-time node failure detection, user-friendly data presentation and inability of Raspberry Pi (BS) to store large data over a long period of time due to small memory capacity. The storage capability of Raspberry Pi in handling accumulated data was extensively discussed and cloud storage facility was proposed by Sheikh and Xinrong in [33] as a long-standing solution.

3.1 SPECIFICATIONS

The pragmatic aim of this experimental research was to implement a communication protocol for data collection, set up a cloud storage service to sync data online in real time from a remote BS and constantly monitor real-time node failure through a web interface.

3.2 SCOPE / AREA OF STUDY

Cloud storage service as an alternative solution for storing and managing of large data that may not be contained in Raspberry Pi was proposed in [33] by Sheikh and Xinrong. In their paper, they suggested integrating cloud service as a long-standing solution to maximize small storage capacity of Raspberry Pi. This proposal was implemented in this project to solve the long-standing challenge. While there may be other challenges in environmental monitoring, this research limits its scope to monitoring and collection of biodiversity data like temperature and humidity on terrestrial (above ground) [58] and management of the data over Amazon Cloud. The Amazon Cloud service (AWS) was used because the platform offers a free service.

3.3 HARDWARE/SOFTWARE DESIGN

The design of this project comprises hardware and software and other methodologies.

The whole design was made to ensure reliability in data communication among WSNs, Primary BS and AWS Cloud Service as secondary storage system for robust and efficient data management for decision making.

3.4 XCTU/XBEE SERIAL 2 (DEVICE COMMUNICATION)

XCTU is open-source software for configuring ZigBee devices to a selected firmware and network protocol. The software allows ZigBee devices to transmit or receive data either in 16 bit or 64 bit. The data formats are in hexadecimal. The network area configuration handles the main configuration of the XBees to ensure either one-way or full-duplex communication. The firmware update was used to choose the XBee type and communication type while the console serial ports were used to construct packets, transmit packets and view the packets' communication on both devices.

The XBee Serial 2 (S2) has a data broadcast rates, ranges from 20 to 900 kbps with a radio frequency bands of 2.4 GHz, 900 MHz and 868 MHz. Two XBee S2s were configured using XCTU software to enable data communication from RX source to TX destination. The source was

configured as COORDINATOR AT Mode with updated firmware and the Destination XBee was configured as ROUTER AT Mode with the updated firmware. The configurations allow data to be received in hexadecimal format from the source.

Table 3.1: COORDINATOR AT MODE configurations

Product family	XB24-ZB
Function set	Zigbee Coordinator AT
Firmware version	20A7
PAN ID	0
Scan Channel	Ffff
My 16 bit Network Address	0
Destination Address High	0
Destination Address Low	Ffff
Baud Rate	9600

Table 3.2: ROUTER AT MODE configurations

Product family	XB24-ZB
Function set	Zigbee ROUTER AT
Firmware version	20A7
PAN ID	0
Scan Channel	Ffff
My 16 bit Network Address	0
Destination Address High	0
Destination Address Low	0
Baud Rate	9600

See Figure 3.1 below.

Figure 3.1: XBee S2

Image source: From my project hardware

3.5 WIRELESS SENSOR NODE (WSNd)

The WSNd components include the following:

i. The Arduino UNO;

ii. Arduino Sparkfun Explorer;

iii. DHT11 temperature-humidity sensor; A high-performance 8-bit microcontroller;

iv. Bread board;

v. USB cable;

vi. Jumper cable; and

vii. Pebble.

3.5.1 ARDUINO UNO

The Arduino UNO is a development board that supports an ATmega 328 microcontroller developed by Atmel with 14 digital pins. Six pins out of the 14 can be used as PWM outputs), six analog inputs, a 16 MHz crystal oscillator, USB port, power port, ICSP header and a power reset button. It contains everything needed to support the microcontroller. It is simply connected to a computer with a USB cable (not included) or powered with a AC-to-DC adapter or battery to get started. It uses Arduino software for programming. The code to communicate to the DHT11 temperature-humidity sensor is uploaded to the Arduino UNO board to receive data and first display the data on the serial monitor, and where necessary, as in this project, it transfers the received data to a BS via ZigBee protocol.

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

External Interrupts: Pin 2 and 3. These pins were configured to trigger an interrupt a change in value of temperature and humidity. See Figure 3.2.

Figure 3.2: Arduino UNO

Image source: From my project hardware

Table 3.3: Specification as listed below [59]

Microcontroller	ATmega328P
Operating Voltage	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limit)	6-20 V
Digital I/O Pins	14 (of which six provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3 V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by boot loader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

3.5.2 DHT11 TEMPERATURE-HUMIDITY SENSOR

The DHT11 has the following functions:

1. The output is Digital Sensor;
2. Onboard sensor DHT11 detects both temperature and humidity of given area (terrestrial);
3. Temperature measuring range: 0℃~ 50℃;

4. Temperature tolerance: ±2℃;
5. Humidity measuring range: 20% ~ 95% (0℃ ~ 50℃);
6. Humidity tolerance: ±5%;
7. VCC: 3.3 V ~ 5.5 V power usage but the circuit connection in this project was 5.5 V; and
8. GND: power supply ground.

DHT11 RH01 temperature-humidity sensor has four pins and three probes out cables. All the three pins in the DHT11 temperature-humidity sensor are connected to the Arduino UNO's similar pin functions via breadboard. The GVS in the DHT11 sensor which represents (G = Ground, V = Voltage and S = Serial Output Data) were bridged and its function extended to the Arduino board. Pin 1 was used to power the sensor (VCC 5 V power) while the Pin 2 was for data output. Pin 3 was not connected but Pin 4 was connect as ground.

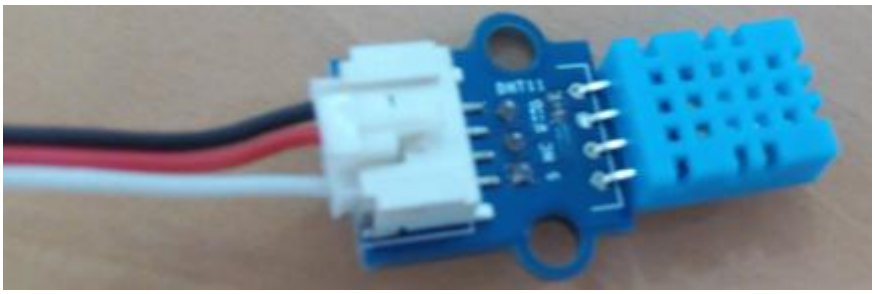


Figure 3.3: DHT11 Temperature-Humidity Sensor

Image source: From my project hardware

3.5.3 BREADBOARD

In this project the breadboard was used as a device construction base in developing electronic circuits for DHT11 temperature-humidity sensor. The breadboard used in this project was a solderless, made of plastic and perforated with several holes. Small tin-plated bronze clips are located under the holes which provided contact points to attach electronic pieces to create a circuit.

The breadboard was used to extend all the functions of the circuit diagram to the Arduino using the jumper cables. All the three pins in the DHT11 temperature and humidity sensor are

connected to the Arduino UNO's similar pin functions. The GVS in the DHT11 sensor were bridged and extended the function to the Arduino board. The breadboard functions as a host for jumper cable connections.

3.5.4 JUMPER CABLES/ WIRE

The jumper cables were components used with the breadboard to generate a functioning circuit. The exposed ends of the cable/wires were used to connect to the breadboard. And similarly three jumper cables were extended to the Arduino for connection to make the full circuit and enable the DHT11 RH01 Sensor to transmit data.

3.5.5 USB CABLE:

This was used as Direct Current (DC) to the Arduino UNO microcontroller board to power both the Arduino board and the DHT11 RH01 Sensor Circuit Diagram.

3.5.6 SPARKFUN ARDUINO EXPLORER

The explorer was used to expand the Arduino UNO board to accommodate XBee S2 ZigBee wireless network transfer protocol. It contains a reset button, 14 Digital Pins and all the analog pins in Arduino UNO. It was soldered to the Arduino UNO board with pebbles. The circuit was connected to the Arduino board through the Sparkfun Explorer.

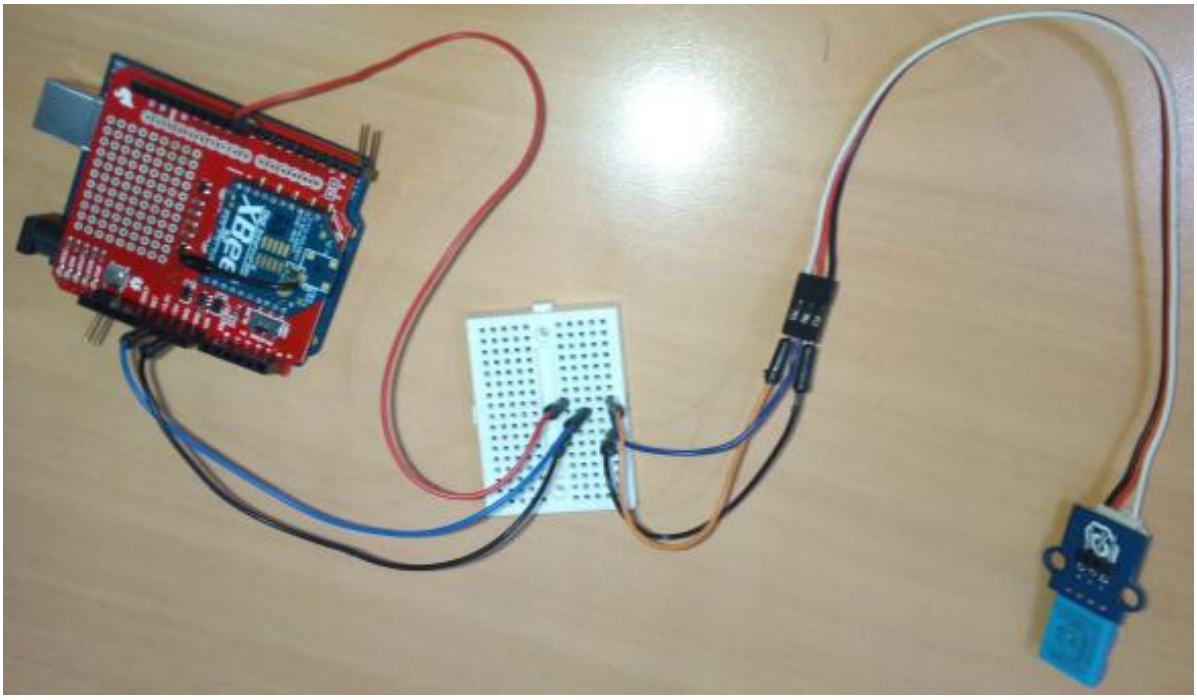


Figure 3.4: Base Station Raspberry

Image source: From my project hardware

3.6 BS (RASPBERRY PI).

The components of the BS include the following;

1. Raspberry (BS);
2. One XBee S2 (Configured as a Router AT mode);
3. Sparkfun XBee Explorer;
4. USB Cable; and
5. Ethernet Network Cable (on a local Network).

3.6.1 SPECIFICATION

Raspberry Pi Model is a low-power credit-card-sized single-board computer. The CPU is an ARM processor with 700 MHz clock speed. CPU performance is comparable to a Pentium II

300 MHz processor and the GPU performance works in line with the original Xbox. It has a variety of peripheral interfaces like USB port, HDMI port, 512 MB RAM, SD Card storage and interestingly an 8 GPIO port for expansion. Raspberry Pi is as good as a desktop computer that uses HDMI and USB connectors to connect IO devices like monitor, keyboard etc. It supports operating systems like Debian-based Linux distro and Raspbian. Raspberry Pi has Ethernet port and can connect to a local network via cable or USB Wi-Fi adapter, and can also be accessed via SSH remote login [33].

3.6.2 BS DESIGN

The Raspberry Pi was configured in this project as a BS and accessed via Secured Shell (SSH) on local area network through a static IP address. The XBee S2 configured as Coordinator AT Mode, was fixed to the Sparkfun Explorer and connected via a USB cable to the BS.

The Ethernet cable was used to link the BS to a local network in order to be accessed via SSH for programming the XBee Router device in AT Mode. The essence was to receive data from the XBee Coordinator AT Mode and stored in a local repository.

See the hardware design for the BS in figure 3.5 below.

Figure 3.5: Hardware design for the BS

Image source: From my project hardware

This wireless device receives data sent from XBEE COORDINATOR-AT Mode at the WSNd via RX of XBEE ROUTER AT Mode and stores it at the BS. The XBEE ROUTER AT Mode was fixed to the Sparkfun Explorer and connected to the BS for data reception.

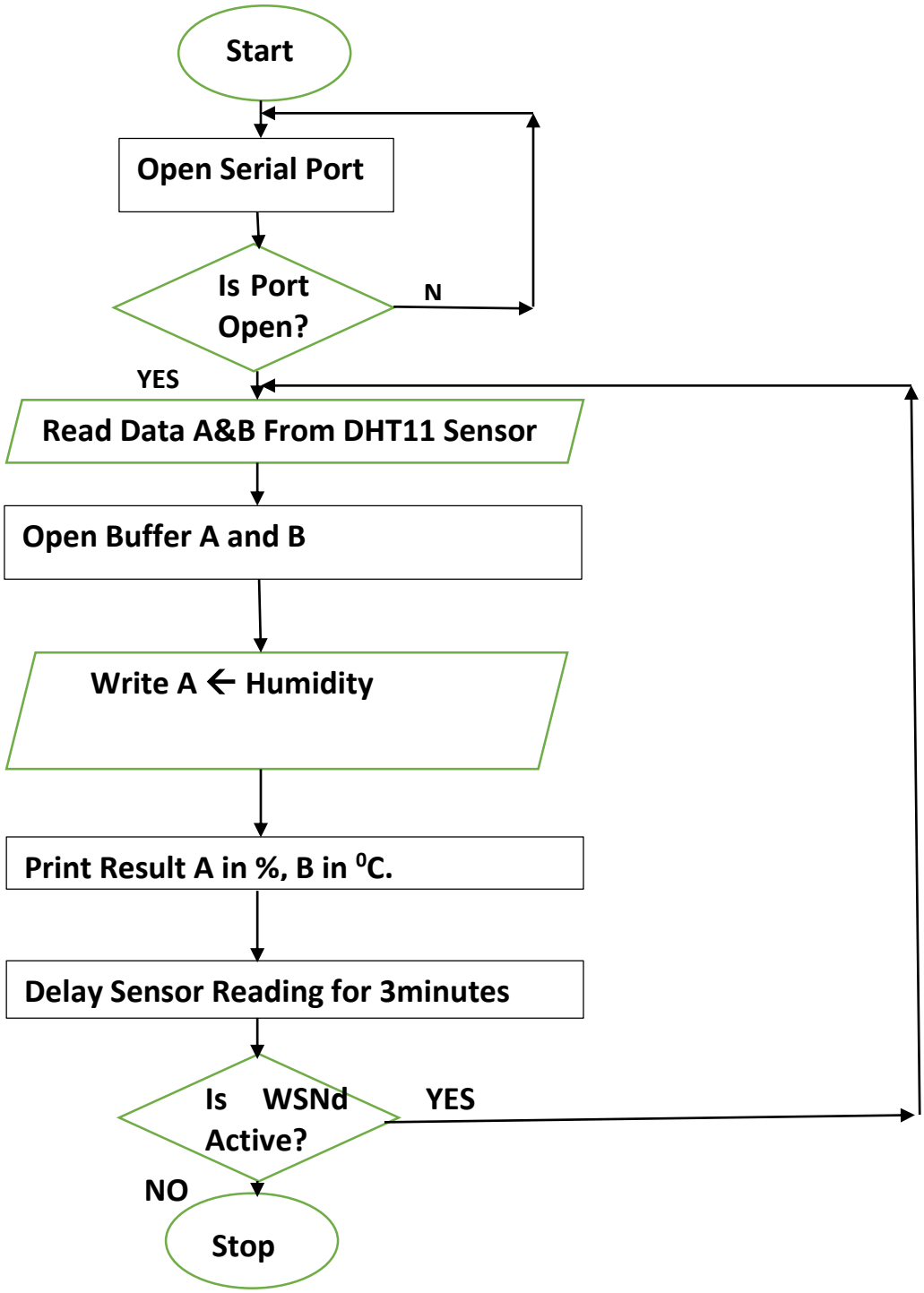
3.8 ETHERNET NETWORK CABLE

The Raspberry Pi was connected to the local area network via an Ethernet network cable. The remote BS was accessed through an SSH connection by binding the static IP address to a local machine.

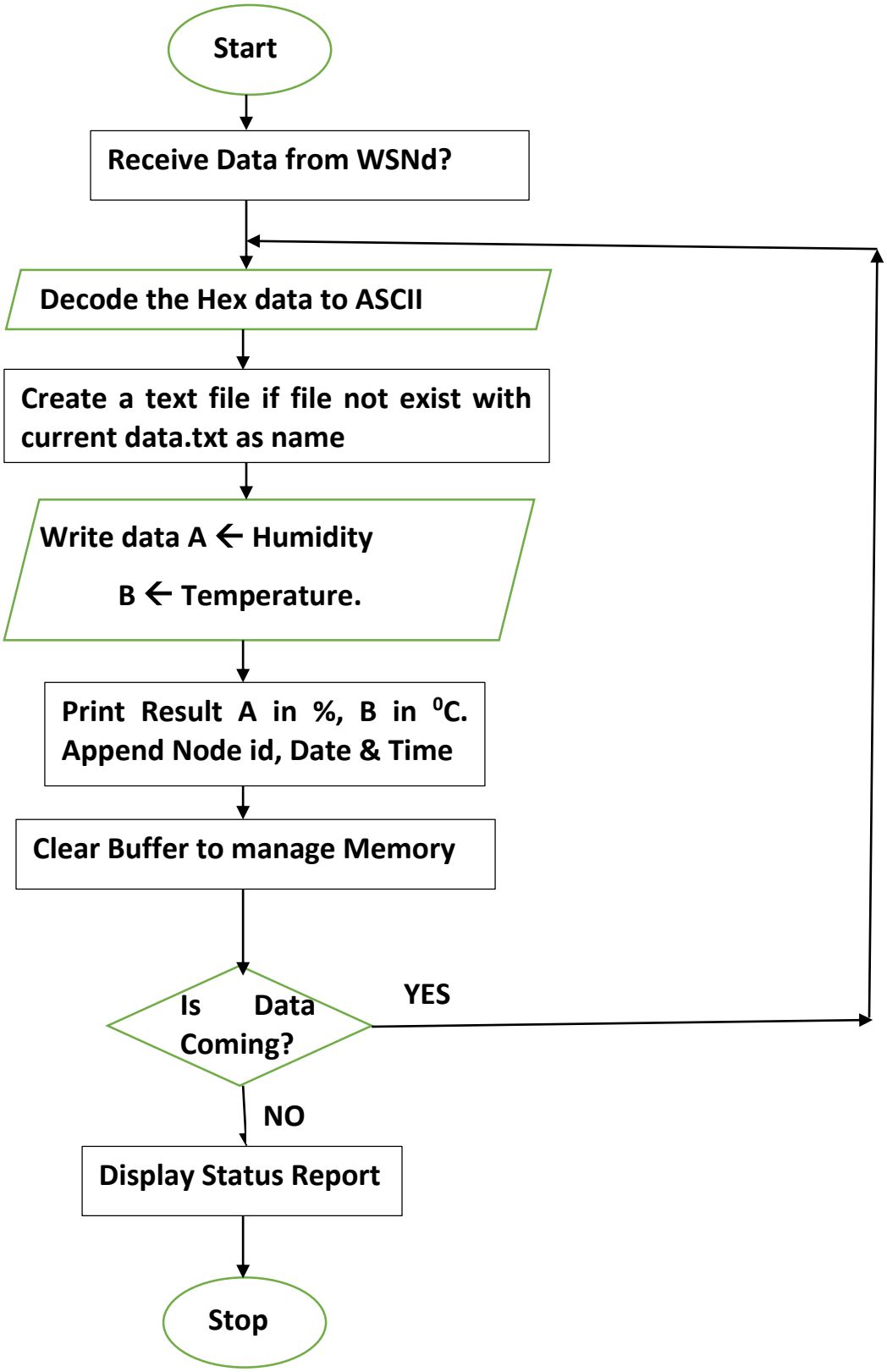
3.9 AMAZON EC2 CLOUD SERVICE (AWS)

AWS EC2 cloud is a free cloud service provider with storage capacity of 30 GB per subscriber and one public IP address. We secured the online server with a key to avoid unauthorized users from having access. Web service was enabled to allow web application programs to be hosted for data effectiveness and better data presentation. All the files, codes and services on the remote BS were periodically synced to cloud through the public IP address for online real-time data presentation.

3.10 WSNd SOFTWARE DESIGN / METHODOLOGY



3.11 BS SOFTWARE DESIGN



3.12 DATABASE DESIGN

The database schema was designed to use MySQL database for data storage and management. A simple data table with the following attributes; Node Id, Humidity, temperature, Date and Reading Time were created to ensure better storage information for future analysis and decision making.

CHAPTER 4

4.0 IMPLEMENTATION

The implementation of the concepts and methodologies detailed in the last chapter comprised over 1000 lines of C, Python, PHP, JavaScript and HTML code. The difficulties encountered in WSN for environmental application in the past, that led to this research as regards online real-time node failure detection, user-friendly data representation and using cloud as a secondary storage server to store large accumulated data that may exceed Raspberry Pi (BS) storage capacity have been solved in this project. There was a clear-cut defined communication protocol to transmit data (TX) and Receive data (RX) encoded in the AT Mode function set.

AWS EC2 cloud service was chosen because it was free to use and offers a reasonable storage capacity with other flexibilities. This chapter contains the full description of the implementation of the main functions and data structures that sums up the whole system.

4.1 WIRELESS SENSOR NODE (WSNd)

4.1.1 DHT11 Library

This library was downloaded from DHT serial official link to allow the DHT11 temperature-humidity sensor to communicate with the Arduino software by making the port open for connection.

WSNd as explained in Chapter 3 is the core determinant in the biodiversity/ecosystem data (temperature and humidity) gathering as implemented in this project. The WSNd comprising Arduino UNO microcontroller board, Temperature-Humidity Sensor, XBee, Jumper cables, breadboard, shield (pebble) for soldering and a USB cable serves as full wireless sensor circuit for data collection. The DHT11 temperature-humidity sensor was meant to collect terrestrial real-time ecological temperature and humidity of a deployment area. The sensor was powered by a 5 V electric power from the Arduino GND pin, 5 V power output pin and data output pin to constantly transmit data to the Arduino UNO board.

The dev/ttyUSB/ Arduino port is usually selected for the device to establish connection to Arduino software for data to be read on the serial monitor on the tool menu.

As described in the previous chapter the software design for the WSNd was carefully implement to ensure good performance and better time complexity as shown below.

```
#include <SoftwareSerial.h>
#include <dht.h>
#define dht_dpin 7 //no; here. Set equal to channel sensor is on
dht DHT;

SoftwareSerial XBee(2, 3); // RX, TX
String humidity, humid;

void loop()
{
DHT.read11(dht_dpin);

char msgBuffer[20];  // Set Buffer for temperature
char msgBuffer2[20]; // Set Buffer for Humidity
const char* one = dtostrf(DHT.humidity, 5, 2, msgBuffer); // maximum character display
const char* three = dtostrf(DHT.temperature, 5, 2, msgBuffer2);
const char* two = "||"; // Character Splitter.
const char* four = "\n"; // Line printing
char result1[50]; //
char result3[70];
sprintf(result1, "%s%s", one, two);// Stores value of humidity and separate the result with a comma (,).
char result2[60];
sprintf(result2, "%s%s", result1, three); // Store humdity and temperature and separate's it with a comma
sprintf(result3, "%s%s", result2, four); //Store humidity and temperature and ends the line(break)

  // If data comes in from serial monitor, send it out to XBee
    XBee.write(result3);
    Serial.print(result3);
    delay(180000)
}
```

First of all, the software serial port was set up to pass data between an XBee Shield and the serial monitor. The XBee Shield established all of the connections needed between the Arduino UNO Board and XBee S2. Another stage was to ensure that the shield switch is in the "DLINE" position to properly connect the XBee's DOUT and DIN pins to Arduino pins 2 and 3.

The Software Serial installed was to communicate with the XBee: In a clear-cut approach the:

1. XBee's DOUT (TX) is connected to pin 2 (Arduino's Software RX)
2. XBee's DIN (RX) is connected to pin 3 (Arduino's Software TX)
3. C library like #include <dht.h> and #define dht_dpin 7 was to set equal channel sensor to on.

To read the data a string was set up to append value of “humidity” as humid;

4.1.2 The void setup() Function

Here both ports are set at 9600 baud. This value is greatly crucial for the XBee. The essence was to ensure the baud rate matches the configuration setting of the XBee as configured with XCTU software discussed in Chapter 3.

The baud Rate 9600 for the XBee Coordinator AT MODE and the Arduino port of Serial.begin(9600) was to allow the baud to communicate with the Arduino.

The delay for 300 milliseconds allowed the system to settle before data reading.

The Serial.println("Humidity and temperature\n\n"); was to make sure that each data as collected the sensor at the node are displayed in line for clarity of purpose.

4.1.4 void loop() Function

This was implemented to read data if serial port is available **DHT.read11(dht_dpin);**

Buffer 1 and Buffer 2 of maximum size [20] were set up to store value for temperature and humidity.

Variable characters were cast to limited length data to display in both temperature and humidity

const char* one = dtostrf(DHT.humidity, 5, 2, msgBuffer);

const char* three = dtostrf(DHT.temperature, 5, 2, msgBuffer2);

The value caste as string character was economically managed because of the issue of memory concern in C and properly formatted with (||) to demarcate value of the data.

All the data coming in from the serial monitor, was sent out to the XBee **XBee.write(result3);** and printed to the Arduino Serial Monitor **Serial.print(result3);** for double validation.

The function **delay (180000);** of 180000 ms implemented each time is collected was save the sensor battery life time by avoiding high overhead cost and not allowing the sensor to access data regularly.

4.3 REMOTE BS

As explained in Chapter 3, Raspberry Pi which is the BS was configured with a static IP address and implemented as a remote server for flexibility and easy accessibility without necessarily being in exactly in the same location with the physical computer to access it. The essence was to make the system more robust and interactive and to avoid constant physical presence and some safety. Although Chapter 3 gave a clear design for the BS, the full implementation required a lot of consideration and logic to make the system operational.

The communication protocol established here was based on IEEE 2006 standard of 802.15.4 of ZigBee wireless communication model. Though Bluetooth wireless communication would have served as an alternative but its now old method. The Bluetooth wireless communication model has shorter range of wireless coverage and lower data bit transfer. So the adoption of the 802.15.4 IEEE communication protocol standard was to maintain international best practices of XBee packets Transfer (TX) from Sensor Node (Source) to Data Destination (RX). The following data structure was implemented to receive data from Sensor Node.

4.4 REMOTE BS (RMBS) DATA COMMUNICATION

Robust algorithm implemented with Python program were used to allow the ROUTER XBee in the remote BS to receive data in hexadecimal format and decode in to UTF8 format. The structure here, establishes a connection and open a serial port to allow data flow between the Wireless Sensor Node and the BS. Software serial library was also installed to enable the XBee Router to communicate the XBee Coordinator.

```
import serial
import base64
import time, os, fnmatch, shutil
import datetime
ser=serial.Serial(port='/dev/ttyUSB0',baudrate=9600,parity=serial.PARITY_NONE,
stopbits=serial.STOPBITS_ONE, bytesize=serial.EIGHTBITS,timeout=None)
count=1
while True:
    ser.flush()
    #timestr = time.strftime("%Y%m%d-%H%M%S")
    t = time.localtime()
    timestamp = time.strftime('3332||%b-%d-%Y||%H:%M:%S||',t)
    line = timestamp + ' ' + ser.readline()
    #timestr = time.strftime('%Y%m%d-%H%M%S)
    today = datetime.date.today()
    f = '/home/pi/'
    f += str(today)
    f += ".txt"
    #f += ".xml"
    f = open('/home/pi/%s.txt' % today, 'a')
    #f = open('/home/pi/%s.xml' % today, 'a')
    f.writelines(line)
    print (line + "\n")
    f.flush()
    f.close()
ser.close
```

The program is executed at the BS with some specific targets as listed in Chapter 1.

At the point of program execution, this function **f = open('/home/pi/%s.txt' % today, 'a')** generates a text file with the current date in the /home/pi/ directory. Sample file is "**2016-05-23.txt"**. The file is updated as long as the sensor node sends in data to the BS on the interval of

time specified on the sensor node. The reason for storing this daily text file with the current date was to keep history of daily environmental data gathering from the DHT11 temperature-humidity sensor for analysis or decision.

On arrival at the BS, the following information is appended to the data for clear description:

1. Current data;
2. Time;
3. Value of temperature;
4. Value of Humidity; and
5. Node Id (Source Node of data transfer).

The program is in a loop and runs **n** times, that is as long as the wireless sensor node sends in data to the BS.

4.5 SER.FLUSH() FUNCTION

This function clears the buffer whenever data is received to avoid mixing of data that may lead to data corruption. At the point of packets transfer, corrupt packets (incomplete data) might be sent. Therefore, the FLUSH() cleanses the memory to create constants space and ensure that packets are sent in full length like the Address (Source and Destination)Header, the Body, the End of packets and the Check Sum.

4.6 BASH SCRIPT

A Bash script as a sequence of instructions that works in a structured loop order for reuse was implemented to back up or transfer files to the online server (cloud) for real-time web service display. The structure is typically executed by entering the text file name of the script on the command line [60].

The bash script implementation performs the following functions.

1) Creates file with current date as file name and Increment backup's text file that contains the daily periodic temperature and humidity values.
2) Transfers the created file to a cloud address using the cloud storage public IP address and find the path to **/var/www/html/job/** to dump the file for further action.

The Rsync is configured to copy only changes made from source to the destination. This process is interesting and makes the whole system more efficient than manually coping bulk file each time. The backup is replicating the source to the destination. However, for the purpose managing the **BS** space which is the core part of this project, file on the BS are remove as soon as it synced to the cloud. Compression or wrapping of the backup data were not implemented, so that makes files in the backup to easily and quickly be retrieved and recovered [61].

```
#!/bin/bash
a="/home/pi/"
b='date '+%Y-%m-%d'`
c="$a$b.txt"
rsync -v -e 'ssh -i /home/pi/wsnema.pem' $c ec2- ser@54.200.108.170:/var/www/html/job/
```

4.7 CRON

This is a periodic job scheduler that enables commands to execute automatically. The cron job here was scheduled to execute a bash script to Rsync collected data file to cloud every 60 seconds to update the system with the last data collected by the WSNd on Online Real-Time bases.

4.8 BS MEMORY MANAGEMENT

Memory space management has been a prevailing issue in using Raspberry Pi as a BS for storing large files. In our work, we implemented a Python program to delete data from a text file as soon as it is written to the cloud and also delete files from BS where the date is older than the current date.

4.9 ONLINE DATABASE SCHEMA

The online database receives the collected environmental data for temperature and humidity every minute. To avoid data redundancy in the database, we implemented an SQL INSERT IGNORE command to ensure strict compliance in maintaining data with unique datetime stamp. We implemented a **Read()/Write()** data policy by first of all allowing the PHP script to open a data text file containing the data packets received from the WSNd, read the data every unique time and write the content to a MySQL data table.

4.10 DATA PRESENTATION CHART

As discussed earlier in Chapter 3, data presentation is one of the most interesting parts of this project because it forms the totality of the whole exercise. We implemented a structured system to store and display valid biodiversity data for temperature and humidity on an online real-time basis. The system also took cognizance of keeping a history of data for future reference and also form a guide in decision making processes for end users. On this part, PHP scripts were implemented to read and display only the value of the **temperature and humidity** on a bar chart to show aggregated data for some time range.

On the History page, any time frame could be queried to ascertain particular data. We implemented JavaScript to refresh the cloud online real-time page every 10 seconds to keep the page updated.

In addition, we implemented a digital representation of each data on the bar chart for easy readability. The scroll page data appends the data and time of every 30 minutes window frame.

CHAPTER 5

5.0 TESTING

5.1 WIRELESS SENSOR DATA COLLECTION

The WSNd data gathering implemented with Arduino code is one of the key breakthroughs in this project because it gave a basis for the work to be expanded to a further stage. The program was tested and the following data output was received from an air-conditioned room. This is a controlled environment but this node can work in any terrestrial habitation. The sensor node is strong enough to withstand any harsh weather. The data below was collected on 23 May 2016 in an air-conditioned office.

Current humidity = 31.00% temperature = 21.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C
Current humidity = 31.00% temperature = 22.00C

5.2 REMOTE BS DATA OUTPUT

As explained in Chapter 4, the Raspberry Pi which we used as remote BS was programmed with a Python script to receipt hexadecimal packets from source and decode to UTF-8. The BS is the heart of the primary storage. It receives packets and stores the file. More importantly it appends the node ID to any received data. Where many nodes are used, data can actually show which nodes transmit X or Y data.

See the output below.

5.2.1 Humidity/Temperature

Node Id:3332||May-23-2016||13:33:13|| 30.00||24.00
Node Id:3332||May-23-2016||13:34:55|| 30.00||24.00
Node Id:3332||May-23-2016||13:37:23|| 30.00||24.00
Node Id:3332||May-23-2016||13:38:06|| 30.00||24.00
Node Id:3332||May-23-2016||13:39:06|| 30.00||24.00
Node Id:3332||May-23-2016||13:39:29|| 30.00||24.00
Node Id:3332||May-23-2016||13:40:30|| 30.00||24.00
Node Id:3332||May-23-2016||13:41:30|| 30.00||24.00
Node Id:3332||May-23-2016||13:42:30|| 30.00||24.00
Node Id:3332||May-23-2016||13:43:30|| 30.00||24.00
Node Id:3332||May-23-2016||13:44:30|| 30.00||24.00
Node Id:3332||May-23-2016||13:45:30|| 30.00||24.00
Node Id:3332||May-23-2016||13:46:30|| 30.00||24.00
Node Id:3332||May-23-2016||13:47:30|| 29.00||25.00
Node Id:3332||May-23-2016||13:48:30|| 30.00||24.00
Node Id:3332||May-23-2016||13:49:30|| 29.00||25.00
Node Id:3332||May-23-2016||13:50:30|| 29.00||25.00
Node Id:3332||May-23-2016||13:51:30|| 29.00||25.00
Node Id:3332||May-23-2016||13:52:30|| 29.00||25.00
Node Id:3332||May-23-2016||13:53:31|| 29.00||25.00
Node Id:3332||May-23-2016||13:54:31|| 29.00||25.00
Node Id:3332||May-23-2016||13:55:31|| 29.00||25.00
Node Id:3332||May-23-2016||13:56:31|| 29.00||25.00
Node Id:3332||May-23-2016||14:01:16|| 29.00||25.00
Node Id:3332||May-23-2016||14:01:16|| 29.00||25.00
Node Id:3332||May-23-2016||14:01:16|| 29.00||25.00
Node Id:3332||May-23-2016||14:01:16|| 29.00||25.00

5.3 LINE GRAPH

We implemented a line graph for presenting and plotting of the temperature and humidity value but noticed that it was not really efficient because of the closeness of the two data values. There was no clear-cut distinction between the two coordinate points and we decided to adopt a **bar chart** data presentation. The adoption of the bar chart graph was due to the fact that it gives a better user-friendly view and gives more understanding of changes in temperature and humidity in relative to time.

See sample of line graph in Figure 5.1.

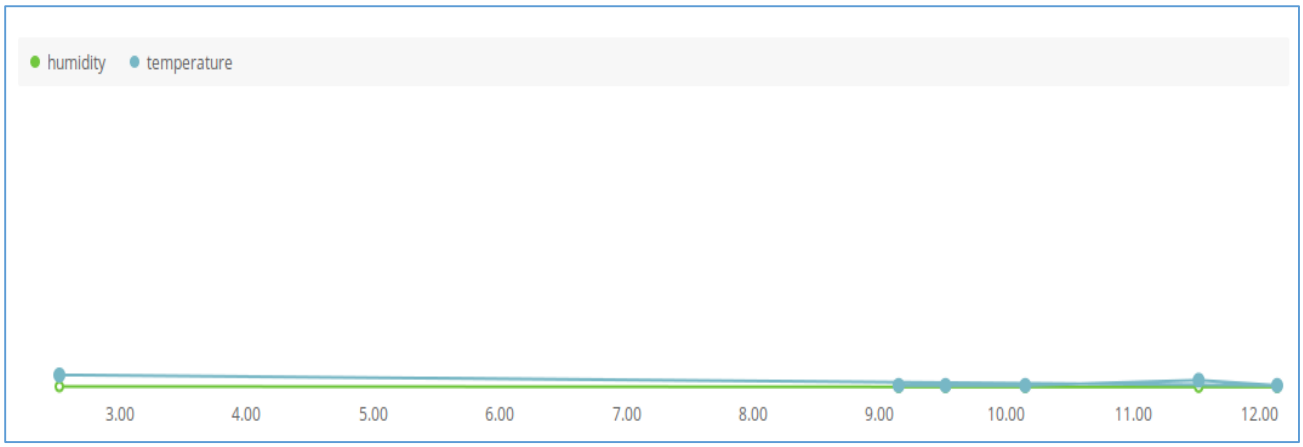


Figure 5.1: Sample of line graph

See Figure 5.2 below for the bar graph data view.

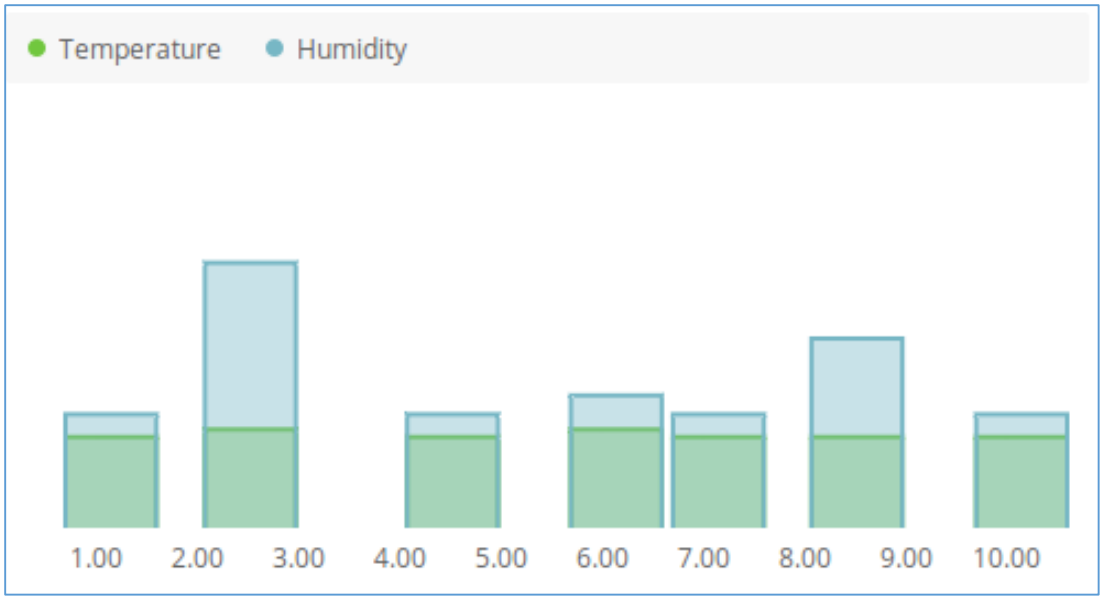


Figure 5.2: Bar graph view

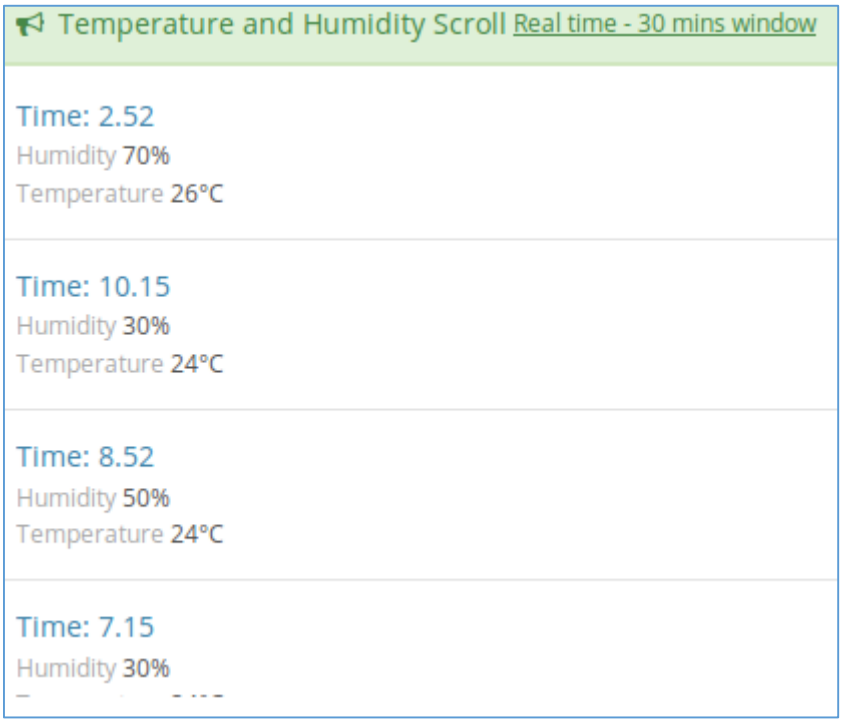


Figure 5.3: Temperature and humidity scroll

CHAPTER 6

6.0 CONCLUSION

We have designed and implemented a WSN for Environmental Monitoring Application (WSNEMA) that was leveraged on cloud as its main storage system for a real-time online ecological data management/storage system. Also a tool for detecting real-time Online Wireless Sensor Node failure was implemented.

6.1 KNOWN LIMITATION

The present implementation of WSNEMA only supports the collection of temperature and humidity data. It is important to expand the system and modify it to collect another type of environmental data like soil PH, wind (speed and direction), pressure etc.

6.2 FUTURE WORK

Further research should focus on developing a generic platform that will be simple for anybody to use. In most cases those who handle environmental data are not programmers and the existing platform requires programmers for implementation. Generic platforms with all sensor libraries installed should be able to allow anybody to buy any terrestrial sensor and select the type on an API (GUI) to configure and the needed code for the running of the sensor(s) automatically.

Due to data loss during packets transmission that leads to data corruption, there is need to further enhance this work and implement an **API MODE** for both the XBee router and the XBee coordinator to make data reception and transmission very accurate. (i.e. by defining Source Address, Destination Address, Message Part, Check Sum, etc.).

6.3 SUMMARY

In the introduction we highlighted some of the problems facing WSNEMA, which include but are not limited to first, lack of immediate detection of sensor node failure on an online real-time basis; second, the issue of small storage space in Raspberry Pi makes it difficult to store large

volumes of data accumulated during data gathering; and third, implementing a user-friendly way of data presentation. We have through this project solved the issues being experienced in the current state of the art of the WSEMA.

This work focused on how to solve the challenges mentioned in Chapter 1 of this project, relating to collection of environmental data (temperature and humidity) using Arduino UNO, Temperature-Humidity Sensor, Raspberry Pi and ZigBee (XBee S2). It could also serve as a prototype for collection of other environmental data that requires wireless sensor node communication. We implemented a scalable structure that can accommodate more sensor nodes where expansion becomes very necessary most especially when considering data aggregation.

Although substantial work has been done in designing and implantation of WSNEMA, there are yet unanswered questions. Taking the research further will not only advance the field of WSN but will complement efforts in understanding and making decisions in combating danger inherent to climate change.

APPENDIX A

- Setup Sensor node circuit, see figure 3.4
- Install DHT11 Temperature-Humidity Library.
- Run Arduino code in page 37
- Click Arduino tool menu to select serial monitor and view the real-time data
- Setup Raspberry Pi (BS) on any local network and assign a static IP Address
- SSH to the BS
- Install Python Serial in the BS
- Run Python code in page 40
- Configure a cron job at BS.
- Run a bash script to Rsync the data collected from the Sensor node. See page 42
- Setup a web service to run a PHP, Java Script to load data to online data base and display data.
- Rsync to cloud on Public IP/location (54.200.108.170/jobs/wsnema.php)

APPENDIX B GUI CLOUD CODE

```
<?php
exec("php transfer5.php");
$hostname = "localhost";
$user = "";
$pass = "";
$db = "wsndata";
$charset = 'utf8';

$dsn = "mysql:host=$hostname;dbname=$db;charset=$charset";
$opt = [
   PDO::ATTR_ERRMODE           => PDO::ERRMODE_EXCEPTION,
   PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
   PDO::ATTR_EMULATE_PREPARES   => false,
];

$pdo = new PDO($dsn, $user, $pass, $opt);
$tempdata = "";
$humiddata = "";
$stmt = $pdo->query('select time_stamp , temperature, humidity from temp_humi where reading_date
= CURDATE() order by time_stamp desc limit 10');
while ($res = $stmt->fetch()) {
        $tempdata .="[";
        $humiddata .="[";
        $time = strtotime($res["time_stamp"]);
        $time = ltrim(date("h.i", $time),'0');
        $tempdata .= $time.",";
        $tempdata .=$res["temperature"]."],";
        $humiddata .= $time.",";
        $humiddata .=$res["humidity"]."],";
}
$tempdata = rtrim($tempdata, ",");
$humiddata = rtrim($humiddata, ",");
$scrolljson = "[";
$stmt2 = $pdo->query('select time_stamp, temperature, humidity from temp_humi where reading_date
= CURDATE() order by time_stamp desc limit 10');
while ($res2 = $stmt2->fetch()) {
        $scrolljson .="{";
        $time = strtotime($res2["time_stamp"]);
        $time = ltrim(date("h.i", $time),'0');
        $scrolljson .= '"time": '. $time.",";
        $scrolljson .= '"humidity": '. $res2["humidity"].",";
        $scrolljson .= '"temperature": '. $res2["temperature"]."},";
```

```
}

$scrolljson = rtrim($scrolljson, ",");
$scrolljson.="]";
//echo $humiddata;
//echo $data;di
/*$cxn = mysql_connect($hostname, $user, $pass);
mysql_select_db($db);
$query = mysql_query("select node_id, temperature, humidity, date, time_stamp from temp_humi")
or die ("Cannot get readings ".mysql_error());
while ($res = mysql_fetch_array($query)) {
        echo $res["node_id"];
}*/
//data: [
?>

<!DOCTYPE html>
<!--
-->
<!--[if IE 8]>      <html class="ie8"> <![endif]-->
<!--[if IE 9]>      <html class="ie9 gt-ie8"> <![endif]-->
<!--[if gt IE 9]><!--> <html class="gt-ie8 gt-ie9 not-ie"> <!--<![endif]-->
<head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <title>Charts - LanderApp</title>
        <meta http-equiv="refresh" content="10; URL=http://localhost/job/wsnemaBar.php">
        <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no,
minimum-scale=1.0, maximum-scale=1.0">
        <!-- Open Sans font from Google CDN -->
        <link
href="http://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic,700italic,400,600,
700,300&amp;subset=latin" rel="stylesheet" type="text/css">
        <!-- LanderApp's stylesheets -->
        <link href="assets/stylesheets/bootstrap.min.css" rel="stylesheet" type="text/css">
        <link href="assets/stylesheets/landerapp.min.css" rel="stylesheet" type="text/css">
        <link href="assets/stylesheets/widgets.min.css" rel="stylesheet" type="text/css">
        <link href="assets/stylesheets/rtl.min.css" rel="stylesheet" type="text/css">
        <link href="assets/stylesheets/themes.min.css" rel="stylesheet" type="text/css">
        <!--[if lt IE 9]>
                <script src="assets/javascripts/ie.min.js"></script>
        <![endif]-->

</head>
<!-- 1. $BODY
-->
```

```
<body class="theme-default main-menu-animated">
<script>var init = [];</script>
<!-- Demo script -->  <!-- / Demo script -->
<div id="main-wrapper">
<!-- 2. $MAIN_NAVIGATION
========================================================================
        Main navigation
-->

        <div id="main-navbar" class="navbar navbar-inverse" role="navigation">
                <!-- Main menu toggle -->
                <button type="button" id="main-menu-toggle"><i class="navbar-icon fa fa-bars
icon"></i><span class="hide-menu-text">HIDE MENU</span></button>
                <div class="navbar-inner">
                        <!-- Main navbar header -->
                        <div class="navbar-header">
                                <!-- Logo -->
                                <a href="index.html" class="navbar-brand">
                                        <strong></strong> 
                                </a>
                                <!-- Main navbar toggle -->
<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#main-
navbar-collapse"><i class="navbar-icon fa fa-bars"></i></button>

                </div> <!-- / .navbar-header -->
<div id="main-navbar-collapse" class="collapse navbar-collapse main-navbar-collapse">
                        <div>
                                <ul class="nav navbar-nav">
                                        <li>
<a href="#"><strong>WIRELESS SENSOR NETWORK ENVIRONMENTAL MONITORING
APPLICATION</strong></a>
                                        </li>
                                                <li class="dropdown">
                                                </li>
                                </ul> <!-- / .navbar-nav -->
                                <div class="right clearfix">
                                        </div> <!-- / .right -->
                                </div>
                        </div> <!-- / #main-navbar-collapse -->
                </div> <!-- / .navbar-inner -->
        </div> <!-- / #main-navbar -->
<!-- /2. $END_MAIN_NAVIGATION -->
```

```
<!-- 4. $MAIN_MENU
================================================================================
Main menu

        Notes:

                * to make the menu item active, add a class 'active' to the <li>
                  example: <li class="active">...</li>
                * multilevel submenu example:
                        <li class="mm-dropdown">
                          <a href="#"><span class="mm-text">Submenu item text 1</span></a>
                          <ul>
                        <li>...</li>
                                <li class="mm-dropdown">
                                  <a href="#"><span class="mm-text">Submenu item text 2</span></a>

                                  <ul>
                                        <li>...</li>
                                  </ul>

                                </li>
                             </ul>
                        </li>
-->
        <div id="main-menu" role="navigation">
                <div id="main-menu-inner">
                        <div class="menu-content top" id="menu-content-demo">
                                <!-- Menu custom content demo
                                        Javascript: html/assets/demo/demo.js
                                 -->
                                <div>
<div class="text-bg"><span class="text-slim">WSNEMA</span> <span class="text-
semibold"></span></div>
                                <a href="#" class="close">&times;</a>
                                </div>
                        </div>
                        <ul class="navigation">
                                <li>
<a href="wsnemaBar.php"><i class="menu-icon fa fa-dashboard"></i><span class="mm-text">Real
Timne Monitoring</span></a>
                                </li>
                                <li>
<a href="#"><i class="menu-icon fa fa-dashboard"></i><span class="mm-text">View
History</span></a>
                                </li>
                        </ul>
```

```
                <!-- / .navigation -->

        </div> <!-- / #main-menu-inner -->

        </div> <!-- / #main-menu -->

<!-- /4. $MAIN_MENU -->
        <div id="content-wrapper">
            <div class="page-header">
                        <h1>Charts</h1>
                </div> <!-- / .page-header -->
                <div class="row">
                        <div class="col-md-6">
<!-- 9. $FLOTJS_GRAPH
================================================================================
                Flot.js Graph

-->
                        <!-- Javascript -->
                                <script>
                                init.push(function () {
                                                // Visits Chart Data
                                                var visitsChartData = [{
                                                        label: 'Temperature',
                                                        data: [
                                                    <?php echo $tempdata; ?>
                                                        ]
                                                }, {
                                                        label: 'Humidity',
                                                        data: [
                                                        <?php echo $humiddata; ?>
                                                        ],
                                                        filledPoints: true // Fill points
                                                }];
                                                // Init Chart
                                                $('#jq-flot-bars').pixelPlot(visitsChartData, {
                                                        series: {
                                                                bars: {
                                                                        show: true,
                                                                        barWidth: .9,
                                                                        align: 'center'
                                                                }
                                                        },
                                                        xaxis: { tickDecimals: 2 },
                                                        yaxis: { tickSize: 100 }
                                                }, {
                                                        height: 205,
                                                        tooltipText: "y + ' degrees at ' + x + '.00h'"
```

```
                                        });
                                });
                        </script>
                        <!-- / Javascript -->
                <div class="panel">
                <div class="panel-heading">
                        <span class="panel-title">Temperature and Humidity Charts</span>
                                </div>
                                <div class="panel-body">
                                        <div class="graph-container">
                                                <div id="jq-flot-bars"></div>
                                        </div>
                                </div>
                        </div>

<!-- /9. $FLOTJS_GRAPH -->
                        </div>
                <div class="col-md-5">
                        <!-- 10. $SUPPORT_TICKETS
================================================================================
                        Support tickets
-->
                <!-- Javascript -->
                        <script>
                                init.push(function () {
$('#dashboard-support-tickets .panel-body > div').slimScroll({ height: 300, alwaysVisible: true, color:
'#888',allowPageScroll: true });
                                })
                        </script>
                        <!-- / Javascript -->
<div class="panel panel-success widget-support-tickets" id="dashboard-support-tickets">
                                <div class="panel-heading">
<span class="panel-title"><i class="panel-title-icon fa fa-bullhorn"></i>Temperature and Humidity Scroll </span>
                                        <div class="panel-heading-controls">
                                                <div class="panel-heading-text"><a
href="#">Real time - 30 mins window</a></div>
                                        </div>
                                </div> <!-- / .panel-heading -->
                                <div class="panel-body tab-content-padding">
                                        <!-- Panel padding, without vertical padding -->
                                        <div class="panel-padding no-padding-vr">
                                                <?php
                                                        $data = json_decode($scrolljson);
                                                        foreach ($data as $item) {
                                                 ?>
                                        <div class="ticket">
                <a href="#" title="" class="ticket-title">Time: <?php echo $item->time; ?></a>
                                                <span class="ticket-info">
```

```
                Humidity <a href="#" title=""><?php echo $item->humidity; ?>%</a><br />
                Temperature <a href="#" title=""><?php echo $item->temperature; ?>&#xb0;C</a><br />
                                                                </span>
                                                        </div> <!-- / .ticket -->
                                                        <?php  }?>
                                                </div>
                                        </div> <!-- / .panel-body -->
                                </div>
<!-- /10. $SUPPORT_TICKETS -->
                                </div>
                        </div>
                </div> <!-- / #content-wrapper -->
<div id="main-menu-bg"></div>
</div> <!-- / #main-wrapper -->
<!-- Get jQuery from Google CDN -->
<!--[if !IE]> -->
        <script type="text/javascript"> window.jQuery || document.write('<script
src="assets/javascripts/jquery.min.js">'+"<"+"/script>"); </script>
<!-- <![endif]-->
<!--[if lte IE 9]>
        <script type="text/javascript"> window.jQuery || document.write('<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js">'+"<"+"/script>"); </script>
<![endif]-->
<!-- LanderApp's javascripts -->
<script src="assets/javascripts/bootstrap.min.js"></script>
<script src="assets/javascripts/landerapp.min.js"></script>
<script type="text/javascript">
        init.push(function () {
                // Javascript code here
        })
        window.LanderApp.start(init);

</script>
</body>
</html>
```

BIBLIOGRAPHY

[1] B. M. G. Jennifer Yick, "Wireless Sensor Network survey," Elsevier B.V. or its licensors or contributors. ScienceDirect® is a registered trademark of Elsevier B.V., pp. Volume 52, Issue 12, Pages 2292–2330, 2016.

[2] O. Thiare, "MSc Project topic proposal," unpublished, Senegal, 2016.

[3] M. B.-A. Rachid Souissi, "An Intelligent Wireless Sensor Network Temperature Acquisition System with an FPGA," *Open Access Article of Scientific Research,* 2014.

[4] E. a. Arne Bröring, "New Generation Sensor Web Enablement," 2011. [Online]. Available: file:///C:/Users/Chris/Downloads/sensors-11-02652.pdf. [Accessed 21 March 2011].

[5] e. a. Shu Yinbiao, "International Electrotechnical Commission," Switzerland, CH-1211 Geneva 20, 2014.

[6] W. S. Y. S. ,. E. C. I.F. Akyildiz, "Wireless sensor networks: a survey Computer Networks," *Published by Elsevier Science B.V,* vol. 38, no. 4, p. 393–422, 2002.

[7] K. a. e. a. Imran, "Wireless Sensor Network Virtualization: Early Architecture and Research Perspectives," *IEEE NETWORK Magazine,* 2015.

[8] S. D. Mohammad Abdur Razzaque, "Energy-Efficient Sensing in Wireless Sensor Networks Using Compressed Sensing," *Open Access Sensors,* vol. 14, 2014.

[9] C. Fischione, "An Introduction to Wireless Sensor Networks," in *Swedish Communication Technologies Workshop (Swe-CTW 2014),*, M□alardalan University, 2014.

[10] A. G. N. L. O. Y. Ado Adamou ABBA ARI, "CONCEPTS AND EVOLUTION OF RESEARCH IN THE FIELD OF WIRELESS SENSOR NETWORKS," *International Journal of Computer Networks & Communications (IJCNC),* vol. 7, no. 7, 2015.

[11] F. F. M. a. L. S. S. Jiang, "A distributed smart routing scheme for terrestrial sensor networks with hybrid Neural Rough Sets. In Fuzzy Systems," in *IEEE International Conference*, 2011, June.

[12] X. W. P. H. W. a. Z. Z. Yu, " Overview of wireless underground sensor networks for agriculture.," *African Journal of Biotechnology,,* vol. 11, no. 17, pp. 3942-3948, 2014.

[13] D. B. a. S. Lino, "A New MAC for Wi-Fi based Underground Networks," 2014.

[14] A. Tarun, "Wireless Sensor Networks and their Applications," [Online]. Available: https://www.elprocus.com/introduction-to-wireless-sensor-networks-types-and-applications/. [Accessed 22 March 2016].

[15] E. J. U. J. P. G. Shankara Rao, "Performance Analysis of Manet Routing Protocols - DSDV, DSR, AODV, AOMDV using NS-2," *Global Journal of Computer Science and Technology: E Network, Web & Security,* vol. 15, no. 6, 2015.

[16] E. a. Muhammad Umar Aftab, "A Review Study of Wireless Sensor Networks and Its Security," October 2015. [Online]. Available: http://www.scirp.org/journal/cn, . http://www.scirp.org/journal/cn. [Accessed 22 March 2016].

[17] S. R. K. a. E. k. Ali Dorri, "SECURITY CHALLENGES IN MOBILE AD HOC NETWORKS: A SURVEY," *International Journal of Computer Science & Engineering Survey (IJCSES),* vol. 6, no. 1, 2015.

[18] ITBusinessEdge, "TCP/IP - Transmission Control Protocol/Internet Protocol," 2016. [Online]. Available: http://www.webopedia.com/TERM/T/TCP_IP.html. [Accessed 17 March 2016].

[19] A. M. Mansuri, "Review of Internet Affinity and Efficiency Assessment of Ad Hoc Routing Protocols," *American Journal of Sensor Technology,* Vols. Vol. 2, No. 3, , pp. 40-43, 2014, .

[20] E. Lam, "TCP/IP Fundamentals TCP/IP Fundamentals, University of California," 25 7 99. [Online]. Available: http://www.sfisaca.org/download/lam.pdf. [Accessed 17 March 2016].

[21] D. Barrera, "Network Security – TCP/IP Refresher," Eidgenossische Technische Hochshedule Zurich, Swiss Fedreal Institute of Technology Zurich, 2014. [Online]. Available: http://www.netsec.ethz.ch/education/courses/netsec-2015/files/00-TCP-IP_refresher.pdf. [Accessed 17 March 2016].

[22] S. S. A. S. Anamika Pandey, "A Survey on Data Collection Techniques in Wireless Sensor Networks," *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 5, no. 1, pp. 273 - 276, January 2015.

[23] S. A. Sumedha Sirsikar, "Issue of Data Aggregation Mothed in Wireless Senmsor Network: A Survey," *Science Direct,* pp. 194-201, 2015.

[24] A. A. P. Ameya S. Bhatlavande, "Data Aggregation Techniques in Wireless Sensor Networks: Literature Survey," *International Journal of Computer Applications (0975 – 8887),* vol. 115, no. 10, pp. 21-24, April 2015.

[25] R. R. S.G. Santhi, "Clustering based Data Collection using Data Fusion in Wireless Sensor Networks," *International Journal of Computer Applications,* vol. 116, no. 9, pp. 21-26, 2015.

[26] "http://www.youtube.com/watch?v=G_ScqGDY2eo," [Online]. Available: http://www.youtube.com/watch?v=G_ScqGDY2eo.

[27] TechTarget, "The difference between ZigBee and IEEE 802.15.4," Search Mobile Computing.

[28] I. Digi, "Demystifying 802.15.4 and ZigBee® White paper," 2015. [Online]. Available: http://www.digi.com/pdf/wp_zigbee.pdf. [Accessed 2015].

[29] D. J. a. S. K. G. Kapil, "ZigBee: A Next Generation Data Communication Technology," *INTERNATIONAL JOURNAL OF INNOVATIVE TRENDS IN ENGINEERING (IJITE) ,* vol. 8, no. 1, pp. 28-33, 2015.

[30] X. Z. X. X. a. T. S. Huang Jinhui, "Zigbee-based Intelligent Home Furnishing," *Computer and Information Engineering, Beijing Technology and Business, University,* vol. 9, no. 1, pp. 61-68, September, 2015.

[31] P. Ian, "Radio Electronic," [Online]. Available: http://www.radio-electronics.com/info/wireless/zigbee/zigbee.php. [Accessed 18 April 2016].

[32] "http://www.nxp.com/files/abstract/press_release/MAXSTREAM_ZIGBEE_PR.html," [Online]. Available: http://www.nxp.com/files/abstract/press_release/MAXSTREAM_ZIGBEE_PR.html. [Accessed 24 March 2016].

[33] X. L. Sheikh Ferdoush, "Wireless Sensor Network System Design using Raspberry Pi and Arduino for Environmental Monitoring Applications," *Procedia Computer Science: The 9th International Conference on Future Networks and Communications, (FNC-2014)University of North Texas, Denton, Texas, 76203, USA,* pp. 103-110, 2014.

[34] B. Brad, "Digital Trends," Arduino vs. Raspberry Pi: Mortal enemies, or best friends?, 8 3 2015. [Online]. Available: http://www.digitaltrends.com/computing/arduino-vs-raspberry-pi/#ixzz44xajZk24. [Accessed 5 4 2016].

[35] D. S. R. a. S. C. D. Sushil Kumar, "Fault Tolerance and Load Balancing algorithm in Cloud Computing: A survey," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. Vol. 4, no. Issue 7, pp. 92-96, July 2015.

[36] A. a. J.-A. L. Sani, "An Autonomous Self-Aware and Adaptive Fault Tolerant Routing Technique for Wireless Sensor Networks," *Open Access Sensors,* pp. 20317-20352, August 2015.

[37] U.S. Agency for International Development, Biodiversity And Development Handbook, Washington, DC 20523: USAID From American People (U.S. Agency for International Development), 2015.

[38] e. a. David Slotwiner, "HRS Expert Consensus Statement on remote interrogation and monitoring for cardiovascular implantable electronic devices," *Heart Rhythm, ,* vol. 12, no. 7, July 2015.

[39] Z. A. M. H. J. E. J. S. A. M. A. J. a. A. I. M. Y. Ahmad Faizal, "Real-Time Remote Monitoring with Data Acquisition System," *4th International Conference on Electronic Devices, Systems and Applications 2015 (ICEDSA),* 2015.

[40] J. S. a. P. L. Singh, " Development of Http Server for Remote Data Monitoring and Recording System," *International Journal of Computer & Technology,,* vol. 11, no. 4, pp. 2440-2445, 2013.

[41] M. S. G. X. a. S. N. Ghayvat H., "WSN- and IOT-Based Smart Homes and Their Extension to Smart Buildings.," *Sensors,* vol. 15, pp. 10350-10379, 2015.

[42] M. T. Lazarescu, "Design and Field Test of a WSN Platform Prototype for Long-Term Environmental Monitoring.," *Sensors ,* vol. 15, pp. 9481-9518, 2015.

[43] N. a. M.-X. D. Ling, "Design of a Remote Data Monitoring System based on Sensor Network," *International Journal of Smart Home,* vol. 9, no. 5, pp. 23-30, 2015.

[44] J.-J. Wang and S. Mu, "Continued Rise of the Cloud: Advances and Trends in Cloud Computing," Security issues and countermeasures in cloud computing. In Proceedings of the 2011 IEEE International Conference on Grey Systems and Intelligent Services (GSIS),, p. pp. 843–846, 15–18 September, 2011.

[45] J. C. Alexa Huth, "The Basics of Cloud Computing," 2011.

[46] C. A. D. R. M. A. P. Bodkhe, "Cloud Computing Security: An Issue of Concern," *International Journal of Advanced Research in Computer Science and Software Engineering,* Vols. Volume 5,, no. Issue 4, p. pp. 843–846, April 2015 .

[47] P. a. G. T. Mell, "The NIST Definition of Cloud Computing, NIST, USA," pp. National Institute of Standards and Technology Special Publication 800-145.

[48] T. Harris, "Cloud Computing - An Overview," [Online]. Available: http://www.thbs.com/downloads/Cloud-Computing-Overview.pdf. [Accessed 7 March 2016].

[49] B. Kepes, "CloudU Understanding the cloud computing stack: PaaS, SaaS, IaaS," 2011. [Online]. Available: broadcast.rackspace.com/hosting_knowledge/whitepapers/understaing-the-cloud-computing-stack.pdf. [Accessed 2 March 2016].

[50] Tutorialspoint.com, "Cloud Computing Tutorial," [Online]. Available: http://www.tutorialspoint.com/cloud_computing/cloud_computing_tutorial.pdf. [Accessed 7 March 2016].

[51] L. T. N. X. Mou Wua, "Dataprediction, compression, and recovery in clustered wireless sensor networks for environmental monitoring applications," *Science Information, Elsevier Inc. www.elsevier.com/locate/ins,* pp. 800-818, 2015.

[52] D. C.Caione, "Distributed compressive sampling for life time optimization in dense wireless sensor networks,," *IEEETrans.Ind.Inf.,* vol. 8, no. 1, p. 30–40., 2012.

[53] R. Y.Quer, "Sensing, compression and recovery for wireless sensor networks: monitoring framework design,," *IEEETrans.Wirel.Commun.,* vol. 11, p. 3447–3461, 2012.

[54] K. a. H. M. Kyukwang, "Sensor Node for Remote Monitoring of Waterborne Disease-Causing Bacteria," *Sensors,* pp. 10570-10579, May 2015.

[55] M. H. R. Bushra Rashid, "Review Applications of wireless sensor networks for urban areas: A survey," *Journal of Network and Computer Applications ,* September, 2015.

[56] A. I. A. R. A. A.-E. a. M. T. Amna Abdullah, "REAL TIME WIRELESS HEALTH MONITORING APPLICATION USING MOBILE DEVICES," *International Journal of Computer Networks & Communications (IJCNC),* vol. 7, no. 3, pp. 13-30, May 2015.

[57] Y. Dunfan, "Application of wireless sensor networks in environmental monitoring: IEEE," 2009. [Online]. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5407035&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5407035. [Accessed 6 April 2016].

[58] M. Bilal, "WIRELESS SENSOR NETWORKS (WSN) & APPLICATIONS," August, 2015. [Online]. Available: http://microcontrollerslab.com/wireless-sensor-networks-wsn-applications/. [Accessed 19 May 2016].

[59] C. Arduino, "ARDUINO & GENUINO PRODUCTS > Arduino UNO & Genuino UNO," [Online]. Available: https://www.arduino.cc/en/main/arduinoBoardUno. [Accessed 20 05 2016].

[60] S. Debugging, "Creating and Running a Bash Script," [Online]. Available: tldp.org/LDP/Bash-Beginners-Guide/html/sect_02_01.html. [Accessed 23 May 2016].

[61] Roelio, "Rsync backup script (bash)," 10 August 2013. [Online]. Available: http://thebestsolution.org/rsync-backup-script-bash/. [Accessed 23 May 2016].