



Data Collection and Transmission for Leisure Time Boats Based on Arduino WSNs and LTE

by

Baddam Sriramreddy, Xiaolong Chen

Supervisor: *Lei Jiao, Frank Yong Li*

Master Thesis in Information and Communication Technology

IKT590, Spring 2013

Faculty of Engineering and Science

University of Agder

Grimstad, 3 June 2013

Status: Final



Abstract:

There has been an astonishing research development in the field of wireless sensor networks (WSNs) in the last decade. A large number of low power capacity devices have been implemented in different vehicles, where sensor nodes act as a team to monitor the environment and forecast the potential defects. In this thesis, we aim to design a data collection system using a WSN on a leisure boat in order to monitor and maintain the boat after sale. The designed system aims to collect data from different sensors on board using WSNs and transmits the collected data to a remote server through cellular network. For the WSNs part, we select a low-power driven Arduino LilyPad as a controller and a XBee interface as transceiver for each sensor node in order to provide a reliable data collection mechanism with a low amount of power consumption. Furthermore, to upload the collected data to a remote server, we adopt a 3G/LTE cellular network for the long range wireless communication. We utilize a PandaBoard as a gateway to connect the WSN and the 3G/LTE network. The designed network is implemented and tested in a lab scenario at university and on a Marex boat along the coast.

Keywords: XBee, WSN, Arduino, Data Processing, MySQL

Preface

This report is the result of the Master Thesis IKT-590 (30 ECTS) course in MSc ICT degree to fulfill the requirements of final semester course content, at Faculty of Engineering and Science, University of Agder (UiA) in Grimstad, Norway. The work on project has been started from 01 January 2013 and ended on 03 June 2013. We have completed the main goals of our master project Thesis "Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE."

This project is a part of the ECO-Boat MOL project which is funded by the Research council of Norway. We would like to thank our project supervisors Professor Frank Y. Li, Ahmed Noor and Lei Jiao for their assistance in giving feedback on both the technical content of report and project presentation throughout this session. In their supervision we learnt a lot about project content and technical report writing. We would also like to thank Marex Boat Arendal for allowing us to test our project on their boats, the Telenor Norway for providing the 4G network router and Spark fun electronics for providing all the electronic components.

Baddam Sriramreddy,

Xiaolong Chen

Grimstad

03 June 2013

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

Contents

Abbreviations.....	12
1 Introduction.....	13
1.1 Background and motivation.....	13
1.2 Problem statement.....	14
1.3 Approaches.....	14
1.4 Report outline.....	15
2 Technology Background.....	16
2.1 Introduction to Wireless Sensor and Ad hoc Networks.....	16
2.1.1 Testbeds.....	17
2.2 ZigBee network.....	17
2.3 Arduino Platform.....	18
3 Hardware and Software Preparation.....	19
3.1 Hardware Components.....	19
3.1.1 Analog and digital sensors.....	19
3.1.2 Lilypad Arduino.....	19
3.1.3 LilyPad XBee.....	20
3.1.4 XBee S2 Module.....	20
3.1.5 XBee USB explorer.....	21
3.1.6 Arduino Mega 2560.....	22
3.1.7 XBee shield.....	22
3.1.8 FTDI breakout board.....	23
3.1.9 BeagleBoard and PandaBoard.....	23
3.1.10 Breadboard.....	27
3.2 Software Tools.....	27
3.2.1 X-CTU.....	27
3.2.2 XBee API.....	28
3.2.3 XBee API frame protocol.....	29

3.2.4	XBee API frame types.....	29
3.2.5	Arduino Platform	33
3.2.6	MySQL.....	34
3.2.7	Windows7 and Ubuntu 12.04	35
4	System Design and Implementation	36
4.1	System Requirements.....	36
4.2	System Architecture	36
4.2.1	System Design.....	37
4.2.2	Local Data Collect System	37
4.2.3	Remote System	37
4.3	Design of ZigBee WSN testbed.....	38
4.3.1	Design specifications of our ZigBee WSN model	38
4.4	Implementation of our ZigBee WSN model.....	42
4.4.1	Preparing the XBee Modules for Data Transmitter & Receiver Side.....	43
4.4.2	Preparing the Remote Sensor Boards for Data Transmitter Side	47
4.4.3	Preparing coordinator node for data collection and processing the data.....	56
4.5	Data Storage and Transmitting to Remote Server	58
4.5.1	Data base architecture and design.....	58
4.5.2	Data sniffing.....	60
4.5.3	Data storing	60
4.5.4	Remote Server.....	61
4.5.5	Data Transmission to Remote Server	62
5	Experiments and Testing Results	63
5.1	Testing WSN at UIA.....	63
5.1.1	Testing Procedure.....	64
5.1.2	Results.....	65
5.2	Testing With Gateway PandaBoard	67
5.3	Testing WSN on Leisure Boat	69

6	Discussions:	73
6.1	Bug of First Message	73
6.2	Unstable MySQL remote connection	74
7	Conclusion and Feature work	76
8	References	77

Figures Contens

Figure 1.1 Application scenario of the data collection system	14
Figure 2.1 Wireless networks. An ad-hoc multi-hop network is formed by sensor nodes to transfer data to the base station. [22].....	16
Figure 2.2 Ad hoc wireless sensor networks. [26]	17
Figure 2.3 Arduino representation [19].....	18
Figure 3.1 LilyPad Arduino [23].....	20
Figure 3.2 General view of LilyPad XBee [32].....	20
Figure 3.3 XBee S2 Module [32].....	21
Figure 3.4 XBee USB explorer [33].....	21
Figure 3.5 Arduino Mega 2560[34].....	22
Figure 3.6 XBee shield [35].....	22
Figure 3.7 XBee Shield connected with Arduino mega 2560.....	23
Figure 3.8 FTDI breakout board [31].....	23
Figure 3.9 Overall view of BeagleBoard-xM [57]	24
Figure 3.10 Design of PandaBoard [57]	25
Figure 3.11 PandaBoard OMAP4430 Platform [25].....	26
Figure 3.12 Breadboard[30]	27
Figure 3.13 Initial window of X-CTU.....	28
Figure 4.1 Overview of system architecture	37
Figure 4.2 ZigBee WSN model	38
Figure 4.3 Sensor nodes flow chart data transmission algorithm	40
Figure 4.4 code for reading the sensor data, to have periodic, event transmission	41
Figure 4.5 Code for transmitting the packet to data receiver	42
Figure 4.6 Modem firmware version of remote XBee module.....	44
Figure 4.7 Modem configurations for coordinator API	44
Figure 4.8 Successful modem configuration message for coordinator XBee module	45
Figure 4.9 Modem firmware version of remote XBee module.....	46
Figure 4.10 Mode configuration parameter settings for XBee remote Module with parameters ...	47

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

Figure 4.11 Successfully modem configuration message for remote XBee module	47
Figure 4.12 Constant 3.3 volts power jack breadboard [36]	47
Figure 4.13 Decoupling the circuit voltage [59]	48
Figure 4.14 LM35 Analog temperature sensor circuit with respect to LilyPad Arduino	48
Figure 4.15 Node 1, lm35 Analog temperature remote sensor	49
Figure 4.16 Code to convert sensor data to real data for lm35 analog sensor.....	50
Figure 4.17 Node 2, flexi force pressure sensor circuit with respect LilyPad Arduino	50
Figure 4.18 Node 2 flexi force pressure sensor.....	51
Figure 4.19 Code to convert sensor data to real data for node 2 pressure sensor	52
Figure 4.20 Node 3, vibration sensor circuit with respect LilyPad Arduino	52
Figure 4.21 Node 3, vibration sensor	53
Figure 4.22 Code to convert sensor data to real data for node 3 vibration sensor.....	54
Figure 4.23 Node 4, one wire digital temperature sensor circuit with respect to LilyPad Arduino	54
Figure 4.24 Node 4, one wire digital temperature sensor	55
Figure 4.25 Coordinator Arduino XBee	56
Figure 4.26 Coordinator sensor data processing algorithm	57
Figure 4.27 Flow chart of data base architecture	58
Figure 4.28 Codes used in creating data base.....	59
Figure 4.29 Architecture of database table in PandaBoard	59
Figure 4.30 Codes for data sniffing	60
Figure 4.31 Codes for database connection.....	61
Figure 4.32 Codes of insert data.....	61
Figure 4.33 Codes of remote database connection.....	62
Figure 4.34 Codes of insert data to remote database	62
Figure 5.1 WSN Testbed in college lab.....	64
<i>Figure 5.2 Individual node behaviour of WSN.....</i>	<i>65</i>
<i>Figure 5.3 Unprocessed data of an individual node received at coordinator node.....</i>	<i>66</i>
<i>Figure 5.4 Processed data of an individual node received at coordinator node.....</i>	<i>67</i>

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

<i>Figure 5.5 Received coordinator data stored in database of gateway</i>	68
<i>Figure 5.6 Viewing the inserted data in MySQL database of gate way</i>	68
<i>Figure 5.7 WSN in leisure boat</i>	69
<i>Figure 5.8 Data stored in PandaBoard gateway on leisure boat</i>	70
<i>Figure 5.9 Data stored in remote server in UIA campus</i>	71
<i>Figure 5.10 Result for two systems</i>	71
Figure 6.1 Modified codes of sniffing data.....	73
Figure 6.2 Incorrect message	74
Figure 6.3 Modified flow char of the program	75

Table Contents

Table 3.1 Relation between hardware and software	19
Table 3.2 Features on PandaBoard [57]	25
Table 3.3 API Basic frame structure [24].....	29
Table 3.4 Examples of some XBee API frame types [29]	30
Table 3.5 Sample ZigBee transmit request API frame format [52].....	32
Table 3.6 Sample ZigBee receive packet API frame format [54]	33
Table 4.1 List of components used to build our ZigBee WZN module	43
Table 4.2 Pin connections of lm35 sensor with respect to LilyPad Arduino and LilyPad XBee	49
Table 4.3 Node 2, Fleiforce pressure sensor pin connections with respect to LilyPad Arduino, LilyPad XBee and bread board	51
Table 4.4 Node 3 Pin connections of vibration sensor with respect to LilyPad Arduino and LilyPad XBee.....	53
Table 4.5 Node 4, one wire digital temperature sensor pin connections.....	55

Abbreviations

WNS	Wireless Sensor Network
CAN	Controller Area Network
LET	Long Term Evolution
AT	AT Command
API	Application Program Interface
IDE	Integrated Development Environment
DIN	Data in
DOUT	Data out
LED	Light Emitting Diode
TXX	Transmission
RXX	Receiver
PAN	Personal Area Network

1 Introduction

In this report, as a part of our master project, we present our ZigBee Wireless Sensor Network (WSN) data collection system to leisure boat which is based on an ARM-based single-board computer. It is small in size but provides excellent performance. In this chapter, we will give a brief presentation of this thesis.

1.1 Background and motivation

Nowadays, in North America and Europe, the market of leisure boat is increasing steadily. Especially in Norway, almost every two families own a leisure boat. The Asian market, especially China, is developing significantly fast. Therefore, improving the feature of leisure boat is the most important aim for boat manufacturers. We focus on data monitoring part of the leisure boat.

For now, data monitoring system on boat is stable and secure but is not so flexible. Since wired system is used, it's hard to add a new device into the system after the leisure boat is manufactured. WSN is a solution for this problem.

WSN offers a variety of technologies which reliable computing can be done. These WSNs make use of several smaller sensors to analyse the properties of the environment. WSNs consist of the combined effect of the embedded system along with the wireless communication that transfers the data through nodes in an ad hoc wireless network. These wireless sensors do not require any fixed infrastructure just like the one used in WLAN or the cellular networks.

This paper provides an overview of how wireless sensor network can be used for the proper transmission of data by using ZigBee network sensor for overcoming the problem of improper and slow data transmission. In order to achieve this kind of system a wireless sensor network needs to be created that have various numbers of nodes and these nodes communicate with each other in full duplex mode. In this project we will use four sensor nodes. The communication between nodes mainly consists of data transfer to the receiver node. ZigBee network has already been used in different models. This technique is used in order to bring down the electricity power consumption. Some of the key features of ZigBee technology can be mentioned as:

- High performance and low cost
- Low power consumption
- Easy to use

ZigBee protocol is used in order to carry out wireless communication. The main advantage of using this ZigBee protocol is that by using this protocol, very less amount of power is required by the nodes, as a result of which these nodes can be operated easily with the help of using simple batteries. This low power consumption by using this technology motivated us in order to use this type of technique. And in this way, the available power can be managed easily by using wireless sensor network that work on ZigBee protocol. An overall operation of the system is controlled by the control device.

Compared with wired system, WSN can accommodate new devices at any time. It's flexible to implement physical partition. For a leisure boat, new sensors can be added easily and the arrangement of sensors can be changed according to consumer demands after manufactured. In this project, we aim at developing a wireless sensor network system by using the power in an efficient way for sensor data collecting, filtering and storing it. Furthermore, we want to integrate this system with another wired data monitoring system which is named Controller Area Network (CAN) bus. This system is developed by another group in our university.

1.2 Problem statement

In this project, the main task is to build a ZigBee WSN module and integrate it with an embedded system which is used as a gateway to filter and store the collected sensor data. Furthermore, with the Internet accessibility, the system transmits the collected sensor data to the remote server, so that the data can be used for commercial or safety analysis. The problems need to be solved can be summarized as follows:

- Implement a wireless sensor network testbed to collect data.
- Process the data so that it is easy to read and understand.
- Improve the quality of communication of the wireless sensor network.
- Reduce the power consumption of every node to increase the life time of the system.
- Develop a program to implement the functionalities of data collection, filtering, storage and transmission.

1.3 Approaches

In order to solve our problems mentioned above, we divided our approach into two part.

- Build a ZigBee WSN to collect and process the sensor data, then transmit it to gateway.
- Design two databases which are located in local gateway and remote server to store the processed sensor data.

In building ZigBee WSN, we use Xbee modules as transceiver and different sensor nodes for collecting data. The Arduino board is used to control the WSN. Firstly, it will process the data which are collected by different sensors thus the results will be easy and clear for reading and understanding. Secondly, the Arduino board can switch sensor nodes into different modes to adapt various demands. For example, switching the node into sleep mode can reduce power consumption so that network can be established with low battery power supply. After implementing the network, the next phase is data filtering and storing the data in local database. Simultaneously, the data will be transmitted to the remote server and stored in remote database.

The application scenario of the data collection system is shown below in Figure 1.1.

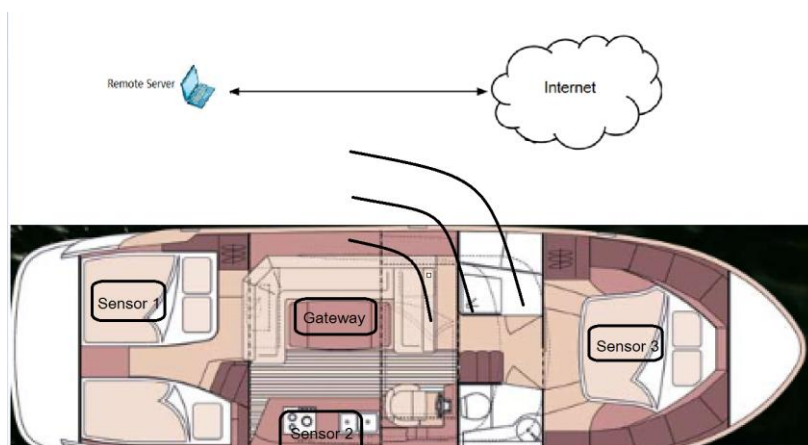


Figure 1.1 Application scenario of the data collection system

As shown in Figure 1.1, the system is implemented on a boat. Three sensors are placed on boat to collect sensor data and transmit them to the local gateway. The local gateway is designed to receive, filter and then store the sensor data. Simultaneously, the local gateway forwards the data to a designed remote server through Internet. Depending on various circumstances, wired or wireless connection can be chosen for the Internet connection

1.4 Report outline

The complete report of this proposed system have been organised in seven chapters. Chapter 1 mainly deals with the introduction part. In this chapter we have explained the main aims and objectives of the system along with the background and motivation for this project. The problem statement is also described in this chapter. Chapter 2 deals with the theoretical background and the literature review part. It explains the theoretical background and technologies that will be used in this proposed system. It gives a brief description about various network technologies. Chapter 3 presents the hardware and software requirement of the system. The main hardware with the pin diagrams have been described in this system together with the software requirements. Chapter 4 explains the main system architecture and structure. It explains about the database used, sensor data transmission etc. Chapter 5 demonstrates the testing and validation part. Various test cases have been designed in order to test and validate the system. Chapter 6 shows the discussions related to this system. It explains how the data is processed. Chapter 7 is the last chapter and it explains the main conclusion and the future work that can be done in this system.

2 Technology Background

In this chapter we discuss about deals with the theoretical background and the literature review part. It explains the theoretical background and technologies that are being used in this proposed system. It gives a brief description about various network technologies and the existing testbeds. The software platforms which are used to build the application sketch.

2.1 Introduction to Wireless Sensor and Ad hoc Networks

The nodes of wireless sensors are implanted systems usually outfitted with the radio, micro controller, energy supply, and memory along with various sensors. Many of these node are used for the doing the environmental related measurements like the others. These are generally used in the high altitude areas. Therefore, these nodes are arranged in a section of study dealing with the mountains [22]. The sink node is not connected directly to all the other nodes, due to which the data measured is passed through an ad-hoc multi-hop network which moves from every node to node until it reaches to the sink node. Figure 2.1 gives us a good idea regarding the nodes in which the 'x' sensor node transfers its data to the sink node through the 'y' node since it doesn't has a direct connection to the sink node.

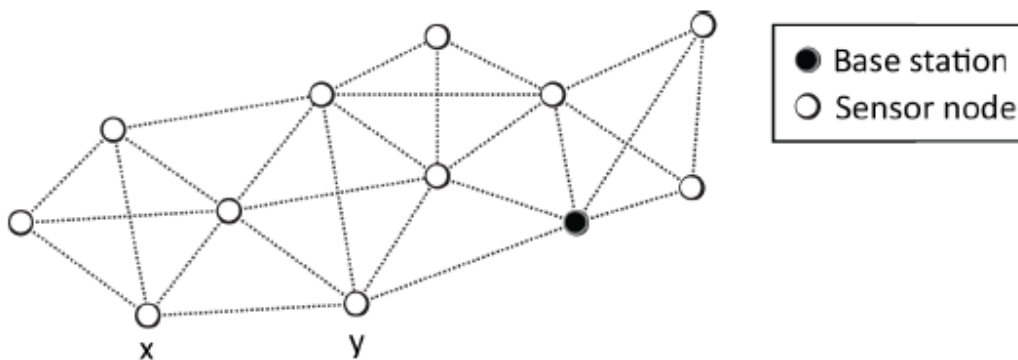


Figure 2.1 Wireless networks. An ad-hoc multi-hop network is formed by sensor nodes to transfer data to the base station. [22]

The base station sends the information collected from all the nodes to the external area where additional dealing of the information is done. The energy efficiency is the major issue since the power supplied to the nodes is through the batteries. There must always be optimized software used for running on nodes of sensors, for example, the updating of the software was done [7]. Some other limitations regarding the sensor nodes other than the energy consumption are memory volume, processing influence and communication area.

In the area of information technology, wireless sensor networks are having a very good scope and also they are coming up rapidly. The initial sensors which were typically wired sensors which were developed in early 1950's like SOSUS [37], which had major usage in military field, hence the sensor networks have become one of the important study areas and being used in many areas.

In the recent times the main issues faced is to reduce the use of energy consumption along with global warming. At the same time many technologies used are fulfilling these needs and also provide better living. WSN offers are those variety of technologies with which omnipresent computing can be done by [14]. These WSNs make use of several smaller sensors for analysing the properties of the environment. WSNs consists of the combine effect of the implanted system along with the wireless communication that transfers the data through these nodes through these

ad hoc wireless networks as shown in the Figure 2.2 These wireless sensors do not require any proper infrastructure just like used in WLAN or those cellular networks. The microcontroller acts as the brains for these each WSN and further transfers the readings through its sensors and also many a times it transfers the readings from the adjacent nodes also since the sensors which are located near them could be closely related to each other due to which the quantity of the data to be transferred through the two nodes of sensors can be considerably minimized.

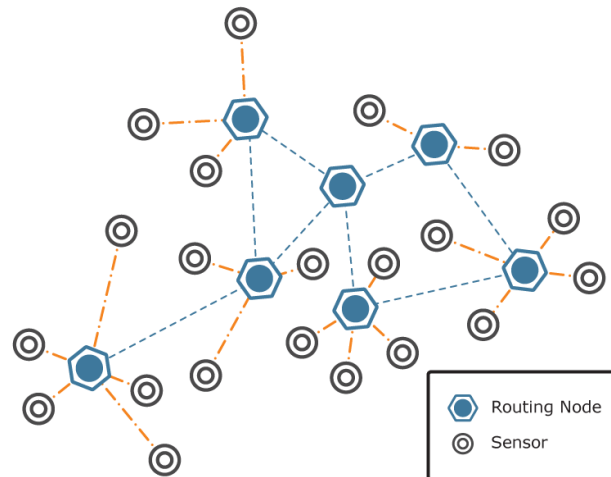


Figure 2.2 Ad hoc wireless sensor networks. [26]

2.1.1 Testbeds

The most common testbeds are DSN, MoteLab, TWIST and Kansei. Testbeds and simulation tools are having various different pros and cons. These are therefore used in WSN for better performance during the corroboration period. The major differences are; in simulation the hardware cost is low, good visibility, proper controllability, and repeatability and also consist of god speed but it has model data quality which is incorrect and is not very sophisticated. Whereas in testbeds, all the pros of simulation are not available but all the limitations of simulation are fulfilled here. Software of simulation such as ns-2, TOSSIM or GloMoSim help in checking the progress of the program. Many nodes can be simulated on one computer which results in minimal cost of hardware required for its testing. The visibility feature of simulation makes the program look better and simplifies the testing of nodes. Since the visibility of simulation is better, each and every node can be inspected properly at a time whereas the simulation data quality need not be always good. After all these advantages, simulation has inadequate channels of communication models whereas testbed makes use of the actual devices and actual communication channels due to which they becomes the choice of most common choice. Testbeds give good quality data but has poor visibility. Simulation can be paused or started any time and gives a good advantage of controllability and the repetition is also made easier by making the nodes go into the process of interest and running the simulation. On the other hand, testbeds doesn't have this feature since they run continuously.

2.2 ZigBee network

Many a times it is misunderstood between the ZigBee and XBee that they are one and the same, which is completely incorrect. ZigBee is basically a typical communication protocol used for the low consuming power, and wireless mesh networks. Whereas XBee is radio brand that supports various types of protocols that also includes ZigBee 802.15.4 along with the Wi-Fi amongst the others [38]. The way Bluetooth is considered as standard in the similar way, the ZigBee protocol is also considered as standard. There are no issues faced by ZigBee to communicate since the ZigBee from any company's device can communicate with any other company's ZigBee device

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

and completely supports it. Just the way Bluetooth of the Motorola headsets can very conveniently communicate with the Apple iPhone, similarly a ZigBee light switch can communicate with Black and Decker door locks. Their working is many times a big question faced. But strong protocol networks consist of layers.

2.3 Arduino Platform

Arduino is very popularly used an open system microcontroller by the designers, educationalist, users who wants to try stuff on their own, and communication of designers. This system is so made which is easy to understand, simple to use, adjustable and also speedy to build up with. Particular jobs are done by the microcontrollers which are equivalent to the computers which is similar to collecting the data from the sensors and switches and then come up to a decision of whether to ring a bell or turn on the light. Due to this function they are widely used in mobile devices, along with the types of microcontrollers which are used in wireless sensor network [43].

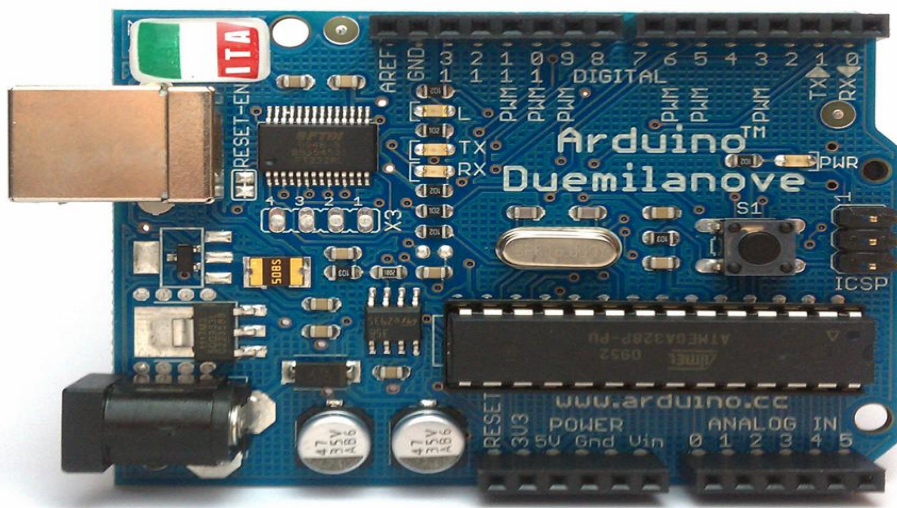


Figure 2.3 Arduino representation [19]

Arduino is able to receive the input from various sensors and sense the environment which can also have control on the various elements such as motors, actuators, and lights and accordingly affect the environment. On the board, the microcontroller is basically programmed with the help of Arduino programmed language which depends on the wiring and development of environment with the help of Arduino that depends on the processing. These projects can work on it by itself or else can also communicate with the software that is running on the computer for example; processing, flash, MaxMSP etc. The basic Arduino board is shown in Figure2.3

3 Hardware and Software Preparation

In this chapter, to set up the testbed, we discuss about Hardware components and software applications that are been used in this project are essential for understanding the complete working of the project. All the required hardware and the software which are used in this project are the latest version and also it is tried, which made cost efficient. The relation between hardware and software is show in the below Table 3.1.

Serial no.	Hardware component	Software application
1	XBee s2	X-CTU
2	LilyPad Arduino	Arduino platform
3	Arduino mega 2560	Arduino platform
4	PandaBoard	MySQL

Table 3.1 Relation between hardware and software

3.1 Hardware Components

In this section, we discuss about the hardware components used to build ZigBee WSN model in our project and its specifications.

3.1.1 Analog and digital sensors

Analog and Digital sensors are two types of sensors playing major role in both wired and wireless sensor networks. We find same kind of sensor in both the types, considerably different in their working and functional application. An analog sensor generates a constant varying value of the output of its range of the measurements. On the other hand, digital sensors consists of only two stages which are mostly known as the 'on' and 'off'. Example of this digital sensor is the touch switch. This switch is generally open when it is not touched, and it is a short circuit when it is pressed or touched.

When these sensors used in the Arduino, it helps in receiving the responses at a faster rate. The voltage output is directly varied along with the temperature in Celsius. In our project we used both analog and digital sensors. Analog sensors are LM35 analog temperature sensor, piezo vibration sensor, fliforce pressure sensor andDS18B20 one wire digital temperature sensor, the circuit connection with respect to Arduino for all the sensors are explained clearly in chapter 4.

3.1.2 Lilypad Arduino

In our project we use this LilyPad Arduino to connect with sensors and to flash the program on the ATmega328 or ATmega168V microcontroller board, which is basically designed for the wearable e-textile industry. Where ATmega 328 at 8MHZ is the updated version of Arduino which is very compactable for programming and easy to reset. Arduino SDK is used to program LilyPad Arduino It has the ability of getting sewn into the fabrics in the same way the power supplies, actuators and the sensors are mounted with the conducive threads.. The outer diameter is 50mm and 0.8mm thickness as shown in the Figure 3.1 [23] LilyPad Arduino consists of 22pins, 14 are digital input output pins, 6 are Analog input pins,

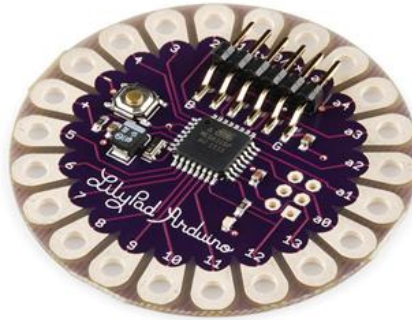


Figure 3.1 LilyPad Arduino [23]

The operating voltage and input voltage should be in between 2.7 and 5.5volts power supply is given either by using USB or external power supply. Each input output pin will effort 40mAof current. The flash memory of LilyPad is 16 KB out of which 2KB is used for boot loader with 1KB SRAM.

3.1.3 LilyPad XBee

The LilyPad XBee is most commonly used as breakout board for all the XBee Modules, works as a radio transceiver and also the asynchronous data transmission between the LilyPad Arduino and the LilyPad XBee. The tabs are made easy to sew along with the directions of the power for making the LilyPad system work and provided with big connecting pads so that it could be easily sewn into the clothes [32]. A variety of the input, output and power sources are present. In one word we can say LilyPad XBee is the reflection of XBee Module with different pin lay out. In order to make LilyPad Arduino projects easier some pins have been swapped to different pin alignment from XBee. The female pins on the top of LilyPad is connected with male fins of XBee module. The power supply should be in between 2.7 and 5.5 volts. The general view of LilyPad XBee is shown in the Figure3.2.



Figure 3.2 General view of LilyPad XBee [32]

3.1.4 XBee S2 Module

In our project we connect XBee module with the LilyPad Xbee for simple reliable sensor data communication between the microcontrollers and in transmitting the sensor data to coordinator. XBee Series 2 module is developed by Digi as shown in the Figure 3.3.



Figure 3.3 XBee S2 Module [32]

It is used in enabling variety of various flavours of specific of ZigBee mesh networking with respect to ZigBee mesh firmware. The Robust network sensors are generally created with the help of the Mesh networks, the systems can give rise to the highly rich datasets or also support the interactions at the human level. XBee supports low power point to point, multi- point and serial port communications with computers. By using ZigBee firmware it is easy to configure our Module as a router, end device and coordinator. The features are XBee series2 module [58] is listed below. The Data sheet is referred in Appendix

1. It has inbuilt antenna.
2. Transmission range in urban area is approximately 40m and 133ft.
3. Line of sight range is approximately 20m and 400.
4. Input output voltage is 3.3V.
5. Peak transmission & receiving current is 40mA.
6. Current power-down is 1micro A.
7. Transmitting power is approximately 2mW.
8. It has 8 digital input output pin.
9. It can configure with AT or API.

3.1.5 XBee USB explorer

XBee USB explorer is used to connect all the XBee Modules to female port pins on the top of it as shown in the Figure 3.4[33] .Use mini USB cable to connect XBee explorer, so that we will have direct access to the serial programming of XBee. It helps in configuring the XBee Module.



Figure 3.4 XBee USB explorer [33]

3.1.6 Arduino Mega 2560

In our project we use Arduino mega 2560 for processing the data .Basically it is ATmega 2560 microcontroller board as shown in Figure 3.5.

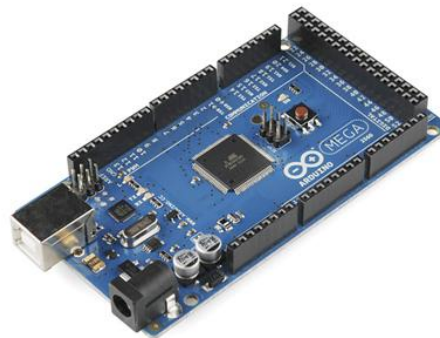


Figure 3.5 Arduino Mega 2560[34]

Arduino mega 2560 consists of 54 digital I/O pins, of which 15 pins are used for PWM outputs.it contains 4 UART pins which helps us in serial port connection to another external hardware devices, and analog input pins are 16 . It can be powered by connecting it to a computer or by connecting 7 to 12 volts AC –to- DC standard [2]adapter [34]. It is flexible and compactable to all other Arduino microcontroller boards. By pressing the reset switch on the board for 10 seconds it will reset the earlier program flashed on the board and resets the settings configured settings.it contains 256Kof flash memory with clock speed of 16 MHz

3.1.7 XBee shield

The XBee Shield as shown in the Figure 3.5 made easy to connect Arduino boards, which helps in Arduino wireless projects. In our project we connect this to Arduino 2560 microcontroller board in preparing the coordinator node and also used to process the received data from different remote nodes.

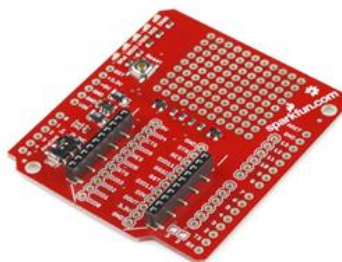


Figure 3.6 XBee shield [35]

This shield connects to any board which consists of the Arduino level prints and updates it with the wireless communications with the help of module of XBee [35].The DIN and DOUT of the XBee serial pins are connected to SPDT switch, that enable us to choose the required connection either the digital pins or the UART pins. The Arduino supplies it with the power of 5V pin and then prior to supplying to the XBee, it converts it to the 3.3V DC. It also provides the easy reset pin on the board. The DIN, DOUT & RSSI of XBee are indicated with LED's which indicates power and activity on the board. The XBee shield mounted to an Arduino mega2560 board can be seen the Figure 3.7

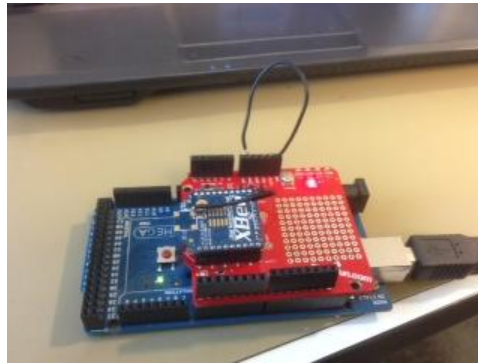


Figure 3.7 XBee Shield connected with Arduino mega 2560

3.1.8 FTDI breakout board

The FTDI breakout board is used to connect LilyPad Arduino, while uploading the program from Arduino SDK on to it. It is also used to monitor the output from serial port monitor. FTDI Basic Breakout 3.3v as shown in the Figure 3.8 is the updated version, the SMD-6 header pin is very compactable to connect to the board, which is adjustable to 3.3V to 5V.



Figure 3.8 FTDI breakout board [31]

The FTDI cable's output and the pin of this board are matched which helps in working with the Arduino and replicate the 5V boards of Arduino. The GRN and BLK usually represent respective colour wires of the FTDI cable. These are used to align the basic and the Arduino board. They consist of the LED lights of TX and RX which are better for using with respect to the FTDI cables.

3.1.9 BeagleBoard and PandaBoard

In our project we use BeagleBoard and PandaBoard in connecting with the coordinator node, which acts as gateway, as backbone of our network and to store the received coordinator sensor data.

BeagleBoard

BeagleBoard-xM provides with the extra ARM Cortex A8 MHz presently at 1 GHz along with additional memory of 512 MB of DDR RAM having low power which allows the researchers, hobbyists and also the engineers to achieve new heights by going beyond their imaginations.

The BeagleBoard xM helps the researchers, engineers and the hobbyist to think and achieve beyond their imagination by providing them with the additional ARM® Cortex™ -A8 MHz having less power DDR RAM available presently at 1 Gz and additional 512MB memory. Modelling with the help of this, improves the design of the hardware of laptops' presentation also the expandability at the time of maintaining power at hand help levels. The hub consisting of four ports having 10/100 Ethernet along with keeping the small 3.25" x 3.25" footprint constant, supports the Direct Current [57]

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

As compared to the initial BeagleBoard, BeagleBoard-xM does not support entire environment development, but on the other hand supports a community platform which is used as foundation for constructing many complete systems of development and also as a baseline of community targets.

A clear view of all the components situated on the BeagleBoard can be observed in the Figure 3.9

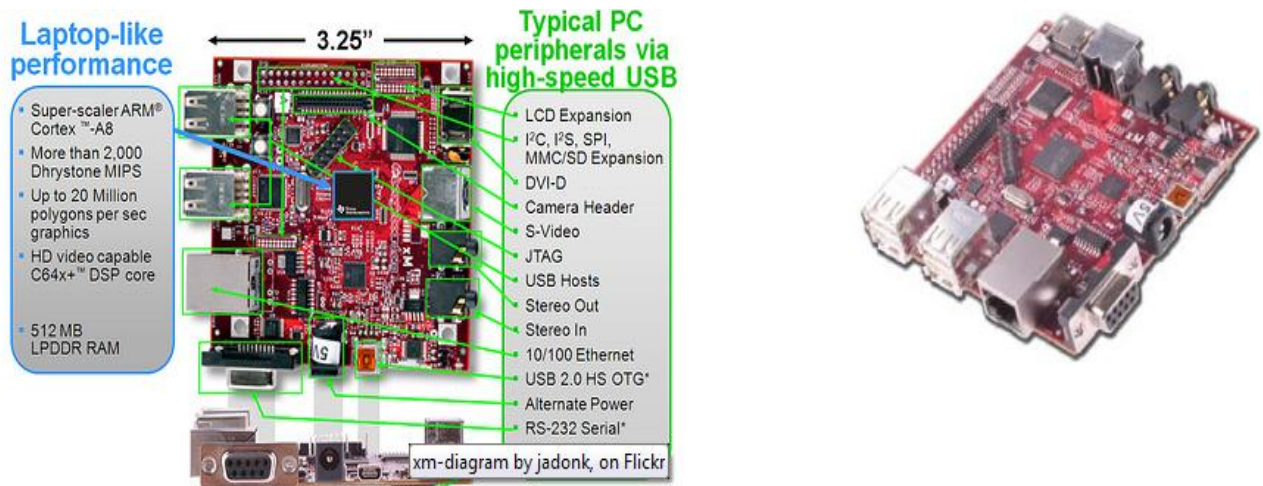


Figure 3.9 Overall view of BeagleBoard-xM [57]

PandaBoard

Basically PandaBoard is used due to its qualities like the low cost, low power computer which is single power platform whose development depends on the System of Instruments of Texas OMAP4430 on the chip (SoC). It is a widely supported platform used for development.

PandaBoard ES is the recent version that depends on the OMAP4460 SoC, consisting the GPU and the CPU working at clock rates which are higher [57].

Basically, PandaBoard is a platform of an OMAP4430 specifically designed for letting the easy path to many of the strong features of multimedia processor OMAP4430 as possible along with keeping the cost minimal. This will help the user to introduce new software which will use the features of the processor OMAP4430 which is powerful [57]. Along with all these, expandability is introduced through connectors on board; this PandaBoard encourages the improvement of extra properties and features. Table 2.1 shows the detail features of the PandaBoard.

Feature	
Processor	OMAP4430
POP Memory	Micron 8Gb LPDDR2 (EDB8064B1PB-8D-F)
PMIC	TI (TWL6030 Power Management Companion IC)
Debug Support	14-pin JTAG
	GPIO Pins
PCB	UART via DB-9 connector
	LEDs
Indicators	3 LEDs (two user-controlled, one overvoltage indicator)
HS USB 2.0 OTG Port	Mini-AB USB connector, sourced from OMAP USB Transceiver
HS USB Host Port	Four USB HS Ports, up to 500mA current out on each, two to onboard connectors, two to expansion connectors
Audio Connectors	3.5mm, L+R out
SD/MMC Connector	6 in 1 SD/MMC/SDIO
User Interface	1-User defined button
Video	3.5mm, Stereo In
	Optional user provided plug-in display
Power Connector	Optional Composite Video out to two-pin header
Camera	USB Power
	DC Power
	Not included, but supported via camera expansion connector

Table 3.2 Features on PandaBoard [57]

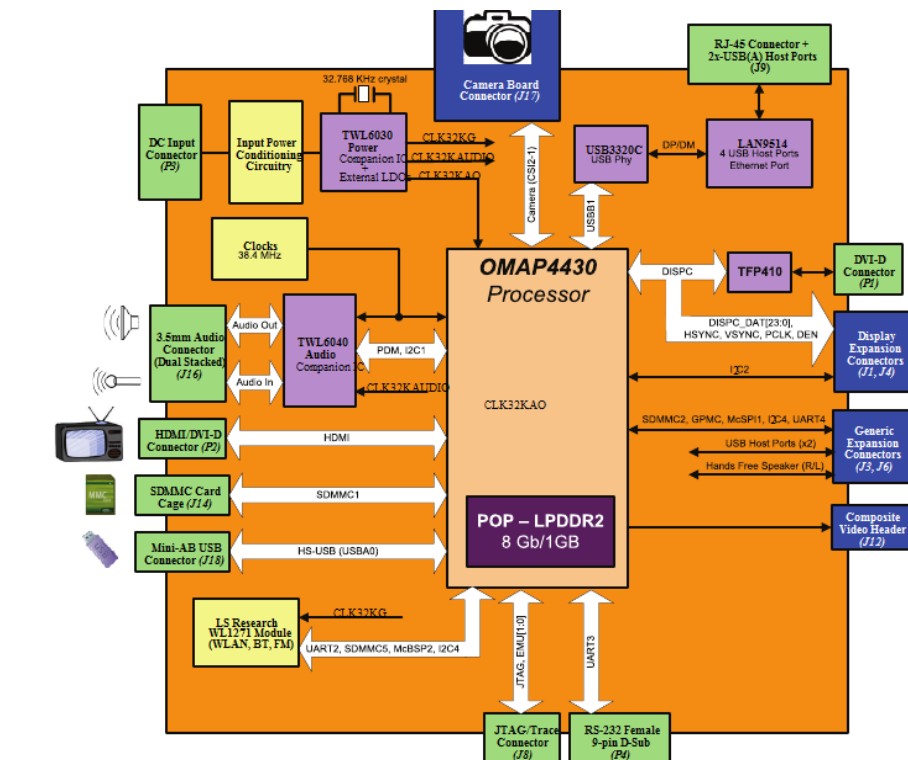


Figure 3.10 Design of PandaBoard [57]

The important features and components of the PandaBoard OMAP4430 are shown in Figure 3.10.

Following is the list of all the major components of PandaBoard [57]:

1. Processor of OMAP4430.

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

2. TWL6030 (Phoenix) Power Management Companion Device.
3. TWL6040 (Phoenix) Audio Companion Device
4. POP Mobile LPDDR2 SDRAM memory.
5. HDMI connector (type A) for OMAP4430 HDMI output Transmitter
6. HDMI connector (type A) for DVI D sourced output.
7. Audio input and output connectors [3.5mm]
8. SD/SDIO/MMC Media Card Cage

The extra connectors are also introduced which help the platform to improve the connectivity and expansion reasons. These connectors are setup by the user on the platform and are not populated. They are basically represented by the blocks on blue colour. Figure 3.10 shows these blue blocks also the following:

1. Connector for camera [J17]
2. Connectors for expansion of LCD [J1 and J4]
3. Connectors for Generic Expansion [J3 and J6]
4. Composite Video Header [J12]
5. The Figure 3.10 represents the top view of the PandaBoard.

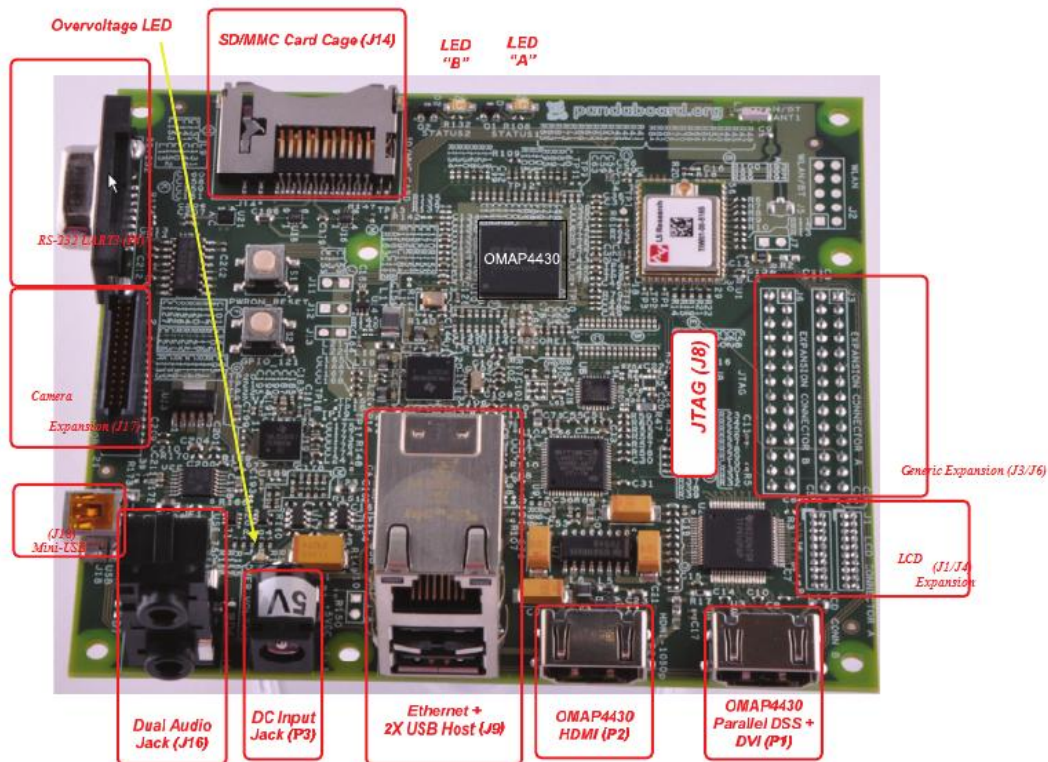


Figure 3.11 PandaBoard OMAP4430 Platform [25]

3.1.10 Breadboard

For making sure whether the circuit designed is working correct or not, breadboard is used. This breadboard consists of number of holes through which the components of the circuit can be inserted like the ICs and resistors. The top and bottom rows of the board are basically used for the connections of the power supply. ICs are inserted in the center of the board so that the half of the legs is on the either side of the center line.

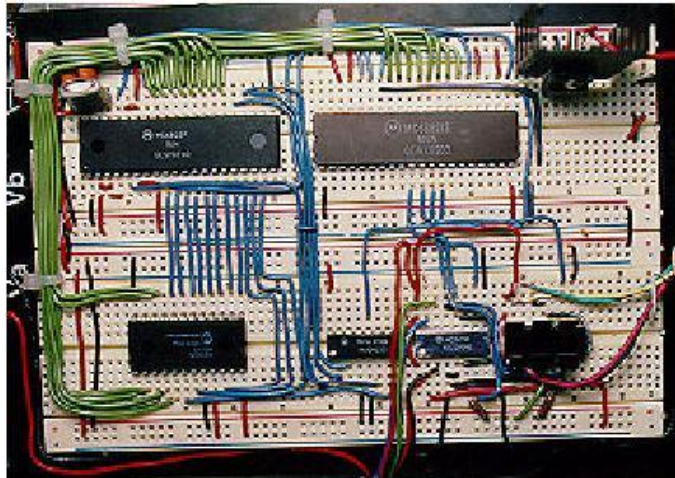


Figure 3.12 Breadboard[30]

3.2 Software Tools

In this section we discuss about the various software and application tools used in our project. Mainly we use X-CTU to configure XBee Modules as transmitting device and Arduino IDE for ATmega328 microcontrollers for Arduino boards.

3.2.1 X-CTU

The X-CTU is an application used for XBee modules, developed on windows by Digi. The main reason behind this application design, is to provide user-friendly interaction between the firmware files provided on the Digi's RF material and to have easy user access graphical interface to them. [17].

The XCTU is the program which is of standard configuration for radios of XBee. This XCTU is present only for the Microsoft windows operating system. We will find the application on digi.com, is open for everyone to download and use, detailed configuration instructions can be found in the downloaded file which also includes completely observed setup of instructions, range tests along with simple access to the features of API. Nowadays, the XCTU is doing very rarely updating the firmware.

3.2.1.1 X-CTU configuration

Once after installing the application on windows operating system we will find the X-CTU application icon on the desktop of our computer as shown in, it can also be seen in start menu followed with programs, Digi and X-CTU.

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

Once if we run the application by giving double click on the X-CTU, available on desktop of your computer, it will open in a window titled with X-CTU as shown in the Figure 3.13 .on X-CTU window we will see four tabs, each tab has its own unique features, they are:

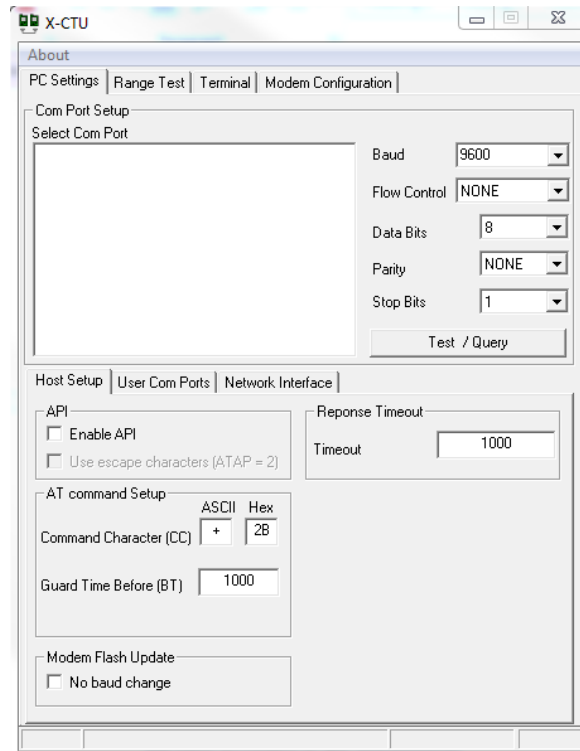


Figure 3.13 Initial window of X-CTU

1. PC settings: This tab makes the user to choose the COM port of their choice and the port is used for the configuration of the radio, in this we set the Baud rate default it is 9600.As we know XBee supports both AT and API mode of operations ,in host setup we chose the mode of our radio XBee.
2. Range Test: makes the user to conduct the tests to know the range between the two XBee radios.
3. Terminal: permits the way to the COM port of the computers consisting the ending program of emulation. It is more feasible to accesses to AT commands to the radios. In our project mostly we use terminal to see the output of our XBee by selecting the serial com port from pc settings tab.
4. Modem Configuration: In this tab we program our XBee radio Modules firmware settings. This tab is very user friendly to program our radios and in updating the firmware .it provides the feasibility to choose the different versions of firmware.

We discuss more about these tabs while preparing the remote radio node in chapter 4

3.2.2 XBee API

In our project we API communication, the major advantage of API over AT mode is the communication to be completed via the structured boundary with the module (the communicated

data is always in the standard order). The API makes sure that the sent commands, reaction to the command and the messages of module status and is received with the help of the module by using the UART Data Frame.

3.2.3 XBee API frame protocol

The API frame of XBee contains a sequence of the bytes which is been built on the every data that is being transmitted. The following table gives the idea regarding the basic frame structure,

Start Delimiter	Length	Frame data	Checksum
Byte 1 0x7E	Byte 2 Byte 3 MSB LSB	Byte 4.....Byte n Specific API frame structure	Byte n+1 Single byte

Table 3.3 API Basic frame structure [24]

Start Delimiter

As we know the output of API fame is Hexa characters, and the data is carried in a structured format in order to identify the API frame, the entire API frames are initiated with the star byte. This shows that the user is at the initial stage of the data frame. If the reading of bytes begins from the midstream of the XBee’s serial port, the user will not understand what it is representing until the order is known. Therefore, always it should have an initial search for the initial byte 0x7E. After which the user will know its location and everything will work fine.

Length Bytes

The two numbers which are received after the start delimiter, represents the complete length of the frame. This helps in knowing the time of how long it should be working [28]. Usually for this situation, the byte which is second is known as the MSB is mostly zero and the third consists of complete length which is known as LSB.

Frame Data Bytes

This type of frame data is different for every message that is received from the XBee radio. Few of the frames will have good amount of internal information, whereas the minor frames contain only 2 bytes. For example, if we consider the data to be like the type of book and every book has its specific structure. Since the length byte is considered as size of the frame data is already known [28]

Checksum

The final byte of the frame is every time a checksum. It is based on the calculations of the bytes which come before it. The calculations are basic sum of entire bytes used for that frame, used at the initial point and checked whether there is any complications during transmission [29].

3.2.4 XBee API frame types

In general XBee API frame structure is categorised into various substructures, which includes all types of information that is to be sent or received with the help of local XBee radio .The different

types of frames consists different data structures [29].In different API type different data frames will be observed and their internal structures are far different from each other. There are different types of API frame types which form specified structures for API frames explained for XBee. Here some of them are explained in details which are required in our project. The bytes of frame type gives us the idea regarding what type of frame is been observed. It is very essential to understand the type of frame for knowing the next upcoming information.

For example, if we consider frame type indicated with 0x08, which shows it is an AT frame command [29]. Therefore on reading the first four bytes we know what kind of the information it is:

- The start byte, i.e. from where the frame starts
- Length bytes, i.e. what will be the length of frame
- Frame type, i.e. what is the type of frame that is been observed.

Each and every individual frame is assigned with a number and we have many kinds of frames are illustrated in the below Table3-3.in our project we use the frame type with 0X10, 0X8B, 0X90and 0X92.

Frame Type	Description
0x08	AT command[immediate]
0x09	AT command[queued]
0x88	AT command response
0x17	Remote Command Request
0x88	AT command response
0x8A	Modem Status
0X10	API TX request
0X8B	API TX response
0X90	RX received
0X92	RX I/O indicator

Table 3.4 Examples of some XBee API frame types [29]

3.2.4.1 ZigBee transmit request

In our project we used this frame type in designing our wireless sensor network. With the help of this frame type, either the remote or the local radio is instructed to transfer the information to another local radio or remote radio, this depends upon where and how we use this frame and changes according to our design specification and requirement.The frame captures the entire real data to the payload along with the set of source addressing and transmission options which explains how the data carried in payload is to be delivered. This frame is been proved as a better example which explains the working of the API mode, when something cannot be achieved effortlessly in the mode of transparent or command mode. For example if we want to change or set new destination addresses on air, we will just simply attach a set of destination address to

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

each frame of data on its path ,instead of sending the commands each time in command mode. The ZigBee transmit request frame format [51] is shown in the Table3-4.The example which we have explained in table is the result obtained in our project.

Frame fields	Offset		Example	Description
Start Delimiter	0		0X7E	Starting byte of frame
Length	MSB	1	0X00	Length indicator
	LSB	2	0X16	
Frame Data	Frame type 3		0X10	Indicates ZigBee Txx request
	Frame id 4		0X01	Acknowledge is sent/if it is 0X00 no ACK is required 0X01 PACKET IS ACKNOWLEDGED
64-bit destination address	MSB 5		0X00	XBee Destination radio module address. This is an 8byte address which is unique. To send broadcast delivery message use 0XFFFF To send message to coordinator use 0X00
	6		0X7D	
	7		0XA2	
	8		0X00	
	9		0X40	
	10		0X79	
	11		0XD0	
	LSB12		0XEC	
16-bit destination network address	MSB13		0X62	Known Destination radio module
	LSB 1		0X5E	If address is unknown
Broadcast radius	15		0X01	Transmission broadcast Can set maximum number of hops
Transmit options	16		0X00	Transmission options can set here If it is 0X00=1 ACK disabled

Real RF data [payload]	17 18 19 20 21	0X10 0X 00 0X 08 0X 43 0X 31	The real data is carried out the size will vary with respect to data	
Check sum	22	F8		0Xff-sumoff the bytes fromoff3 to till last byte gives check sum value

Table 3.5 Sample ZigBee transmit request API frame format [52]

3.2.4.2 ZigBee receive packet

The ZigBee receive packet frame formation in API provides us with the better results as compared with the basic AT transparent /the command mode interactions because the receiver node will not identify ,from which node the data is transmitted and who the sender was. In the case of single network group the receiver will identify, but in case of large network it is substantially essential to know who the sender was and what the data is received.so this is problem is over rectified if by using this inserting frame type and this is identified by 0X90 and the fame id is also transmitted by sender. The sample ZigBee receive packet fame is illustrated in the table and the example is taken from the out of our project result.

Frame fields	Offset		Example	Description
Start Delimiter	0		0X7E	Starting byte of frame
Length	MSB	1	0X00	Length indicator
	LSB	2	0X12	
Frame Data	Frame type 3		0X90	Indicates ZigBee receive packet
64-bit destination address	MSB 4		0X00	XBee Destination radio module address. This is an 8byte address which is unique. To send broadcast delivery message use 0XFFFF To send message to coordinator use 0X00
	5		0X7D	
	6		0X33	
	7		0XA2	
	8		0X00	
	9		0X40	
	10		0X79	
	11		0XD0	

	12	0XEC	
16-bit destination network address	MSB13	0X62	Known Destination radio module
	LSB 114	0X5E	If address is unknown
Broadcast radius	15	0X01	Transmission broadcast Can set maximum number of hops
Transmit options	16	0X01	0X01 packet is acknowledged 0X02 broadcast packet 0X20 APS packet encrypted 0X40 end device packet transmission
Real RF data [payload]	17	10	The real data is carried out the size will vary with respect to data
	18	00	
	19	08	
	20	43	
	21	31	
Check sum	22	F8	0Xff-sumoff the bytes fromoff3 to till last byte gives check sum value

Table 3.6 Sample ZigBee receive packet API frame format [54]

With the help of 64-bit destination address the receiver radio node will understand from which remote radio the data is transmitted. And also with the payload information which is then followed by the checksum [53].

3.2.5 Arduino Platform

Arduino software environment is easier to program with minimum programming knowledge, mostly it is being used by students and researches. The programs which were developed using Arduino are named as sketches, and these sketches are developed on computers using the Arduino integrated development environment "IDE". The Arduino IDE is compactible for Linux, mac operating systems. It is easy to edit the sketches and easy to convert into Arduino hardware instruments. In our project we convert the sketches into LilyPad Arduino instrument .

3.2.5.1 Arduino programming structure

For example, a simple program may look similar to this program which is taken from the Arduino cook book:

```
// definitions of variables always come initially

int ledPin = 13;

// The setup() method runs only, after the sketch starts

void setup() {

// starting the digital pin as an output:

pinMode(ledPin, OUTPUT);

// the loop() method runs continuously again and again,

// as far as the Arduino has strength

void loop()

{

digitalWrite(ledPin, HIGH); // switch the LED on

delay(1000); // stop for one second

digitalWrite(ledPin, LOW); // turn the LED off

delay(1000); // stop for one second

}
```

Basically the program starts with the sentences which speaks out the types, names and the basic values for the allotted containers are been used all the way through the program known as the global variables. The next part starts with the void setup (). Once the Arduino is powered up all the commands between the curly bracket is only performed once. Basically this part consists of methods ,that are used to make the Arduino board ready to work, such as starting of pins, arranging the serial ports and all the remaining thins which are supposed to take place only once after the starting up. Lastly consists of void loop (), all the commands in its curly bracket executes continuously. In the above given example, the LED light will switch on stay for some time until it switch off and again wait for some time. This will blink for one complete cycle, immediately after this one cycle is completed; the program starts again and turns the light on and off continuously. Many a times there will also be other parts that will follow the loop.

3.2.6 MySQL

MySQL basically, is database management system. The organized collection of the data is known as database. For inserting, permitting and working the data that is present in the database of computer MySQL management database system is required.

MySQL such a database management system which is relational based. This relational database helps in dividing and storing the data in the different tables instead of storing all the in one big room. This helps in improving the pace and suppleness. All the tables present and

related with each other by the definitions which make it simpler for mixing the data by the request of several tables [49]. The SQL in MySQL basically stands for 'Structured Query Language'. This is most widely used and is a standard language for accessing the database. MySQL is an open Source Software, which indicates that any user can add or modify the according to their requirements. It is free so user can also download it from the internet for using without paying for it. Everyone who is interested can also studying the source code and modify it according to their requirements.

MySQL is mostly used since it is very speedy, trustworthy and also simple in using. Mostly these are the basic requirements of any user. It also consists of many practical groups of features which can be basically developed for better usage of the user. This MySQL was basically so designed so that it could handle database which are of larger sizes, also have speed as compared to the present solutions and due to this they have been used on a very large scale from many years. Even though it is under regular development process, in today's time, MySQL provides us with the rich and useful group of features [50]. Various features like the connectivity, faster rate, and safety of MySQL makes it most suited for the internet accessing of the database server or client system which contains multi-related servers of SQL which supports various back ends, various different programs for users and libraries, program interface and tools for administration. MySQL is basically some of the technical features are given. Mainly MySQL consist of much software present which is contributing. It can be very easily found that whichever is your preferred application or the language, MySQL must be already supporting it.

The main values of this MySQL and which we desire it to be:

- The most common and the best database that is used in the entire world.
- Low cost and easily available for everyone.
- Simple in usage.
- Constant developing and during which it should remain safe and fast.
- Ease in using and improving it.
- It must be Bugs free.

3.2.7 Windows7 and Ubuntu 12.04

As we everyone know without these operating systems we can't accesses to any of these applications .in our project we mostly use windows 7 as our operating system to use XCTU,Arduino and to connect all the hardware components during their configuration settings.

In our project the BeagleBoard and PandaBoard works on Ubuntu, is the operating system of the computer which depends on the distribution of the Debain Linux which makes the use of its own desktop area.

4 System Design and Implementation

In this chapter we discuss about the architecture of our system, the design specifications of our ZigBee WSN model, the algorithm we used in WSN model, the configuration settings of our XBee modules, the circuit design of remote sensor boards with respect to Arduino, data processing of our coordinator radio node and the database architecture of to store the received coordinator data.

4.1 System Requirements

The requirements listed below are to be considered in our system design and implementation.

1. A ZigBee WSN testbed is required to collect sensed data.
2. The collected sensor data is transmitted to local PC or PandaBoard.
3. To receive the collected sensed data, a coordinator node is to be prepared.
4. Processing of collected sensed data is to be done at coordinator.
5. Store the collected sensed data in PandaBoard.
6. Data transmission from PandaBoard to remote server.

The wireless sensor network should be stable and reliable for collecting data. Since nodes are battery dependent, power consumption is a significant concern. Sleep mode is a good choice to lower the power consumption. At the same time, the transmitting payload should be minimized and optimized. The output of the data needs to be short and simple so that consumer can understand it easily. A simple but stable database is needed for storing data. Meanwhile, this stored data will be transmitted to the remote server.

4.2 System Architecture

The whole system is designed in three parts including data collection, data storing and data transmission. The overview of system architecture is shown Figure 4.1. A local data collection system is designed in this project and we will use a remote system from another project named "Marine Data Collection based on Embedded System with Wired and Wireless Transmission". We need to integrate these two systems together. Data from both sides should be collected and stored in the same remote database at the same time.

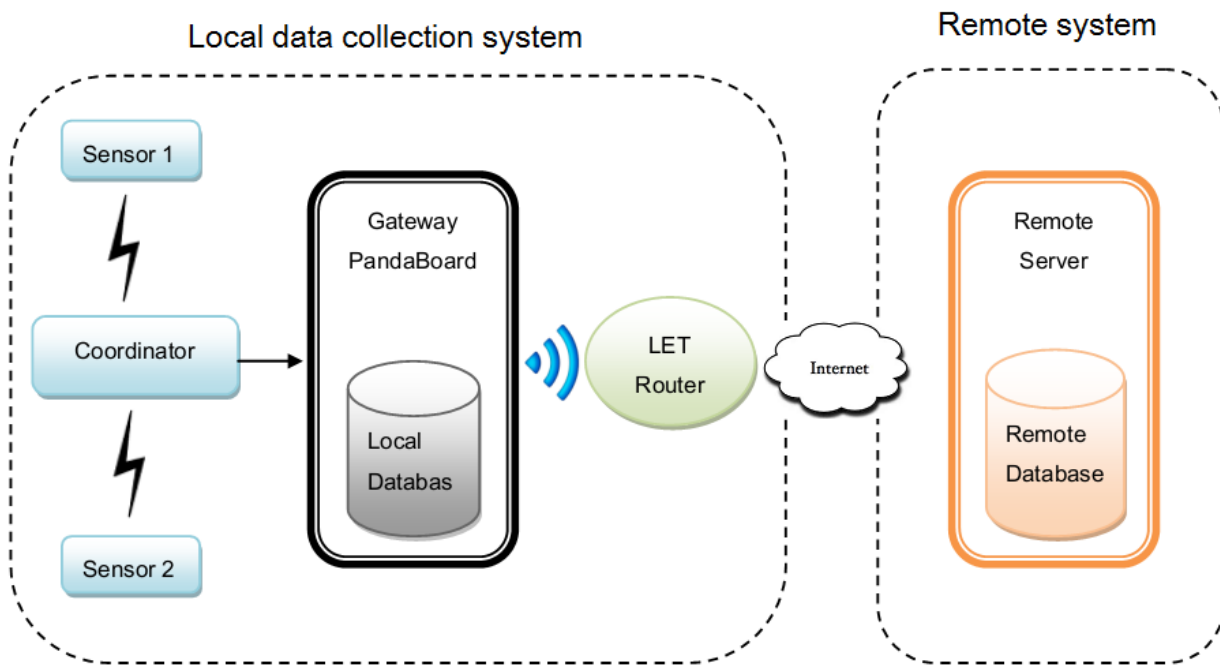


Figure 4.1 Overview of system architecture

4.2.1 System Design

The first part is the data collection part which is based on a ZigBee WSN. For the moment, four different sensors are used for collecting data. Xbee devices are used to transmit and receive data. In this network, every node sends the collected data to the coordinator through ZigBee network. For the data storing part, coordinator node forwards the data to the gateway PandaBoard which is connected by USB serial port. In the gateway, the data will be received, analyzed, processed and stored locally. At last, gateway PandaBoard connects to a LET router by WIFI connection and then the LET router connects to Internet. The gateway and remote server will communicate through the Internet. The data which is stored in local database will be transmitted to the remote server and then be stored in remote database.

4.2.2 Local Data Collect System

For local data collection and transmission, we use LilyPad Arduino microcontroller boards to read analog and digital sensors values form different nodes in the network, for transmission and receiving the data we use XBee modules. By using XCTU application these XBee modules are configured as 'ZigBee router API' for transmitting the sensor data, for receiving the sensor data we configure the XBee module as 'ZigBee coordinator API' and these configuration settings for both transmission and receiver side is explained during the implementation process of our project. To see the received data and store it, the coordinator is connected to gateway. This gate way will act as a back bone of our network while transmitting the data to remote server.

4.2.3 Remote System

The remote server is done by another project which is named 'Marine Data Collection based on Embedded System with Wired and Wireless Transmission'. Since we will integrate these two systems together, we use the same remote system. A database is created in this remote server for storing our data.

4.3 Design of ZigBee WSN testbed

In general in any network we will have both transmitter and receiver, even in our ZigBee WSN Model as shown in Figure 4.2. First we discuss about our data transmission side, then data receiver side. In our model we use XBee API frame protocol.

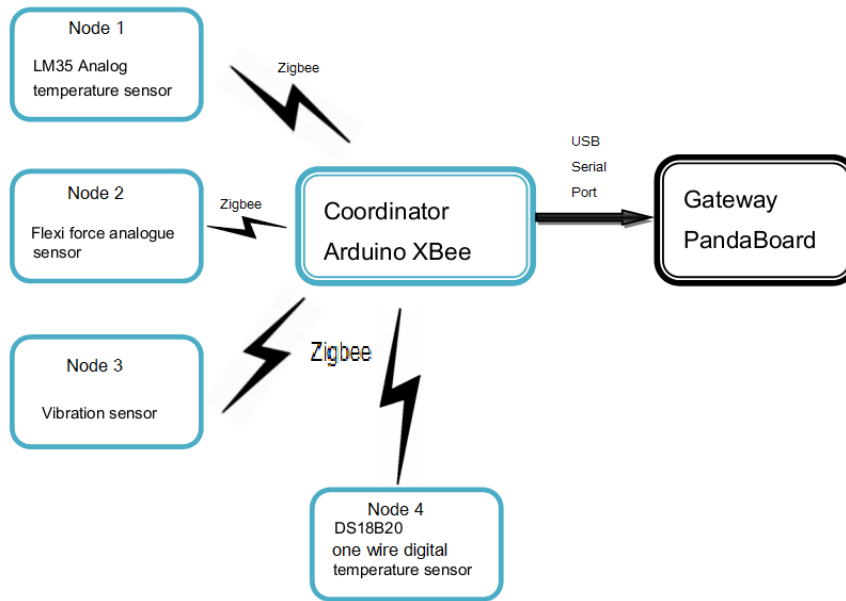


Figure 4.2 ZigBee WSN model

Data Transmission Side:

In data transmission side we use XBee Modules for sensor nodes configured as ZigBee router device. In our project we use four sensor nodes, three of them are analog sensors nodes and one is digital sensor node. Node 1 is for LM35 Analog Temperature sensor, Node 2 is Flexiforce Pressure sensor, node3 is for vibration sensor and node 4 is digital temperature sensor.

The purpose of these nodes is to read the sensor data and to transmit the data to coordinator node in the receiver side. For reading the sensor data these nodes should connect to their respective individual LilyPad Arduino microcontroller sensor board which is explained in 4.4.1 while preparing the remote sensor board. Now we have to design and upload program for each individual sensor board with respect to our design specification of ZigBee WSN. The design specification is same for all the nodes which is explained in section 4.3.1 to write a program we use Arduino IDE.

Data receiver Side:

In the receiver side, the main purpose is to receive the sensor data, process the data. To receive the sensor data we use coordinator node which is XBee module configured as ZigBee API coordinator device connected to gate way which is either PC or panda board. To process the data we use Arduino mega 2560 microcontroller board to see all the transmitted with respect the design specification of coordinator data, the design specification is. We use Arduino IDE.

4.3.1 Design specifications of our ZigBee WSN model

Data Transmitter Side

During the data transmission of remote sensor node, each remote sensor node in the network should transmit the sensor data by satisfying the two conditions below:

- Each remote sensor node should send the data periodically, i.e., every 10 seconds, along with its source address, node id, node type, and transmission sequence number for each transmission.
- Within these 10 seconds, if there is any change in sensed data, meaning that an environmental change is detected, it will also send the data to coordinator. The payload should be made as short as possible to save energy.
- When there is no data transmission, the remote sensor node should sleep to save the battery power.

In order to operate the nodes in the above explained fashion, we developed an algorithm to have periodic transmission, event communication, and sleeping mode. The data is transmitted by using ZigBee Transmit request API Frame type protocol, ZigBee receive packet API Frame type protocol is used in our algorithm. The flow chart of the sensor node data transmission algorithm is shown Figure 4.3. By using Arduino IDE we developed algorithm, C programming language used to write the code.

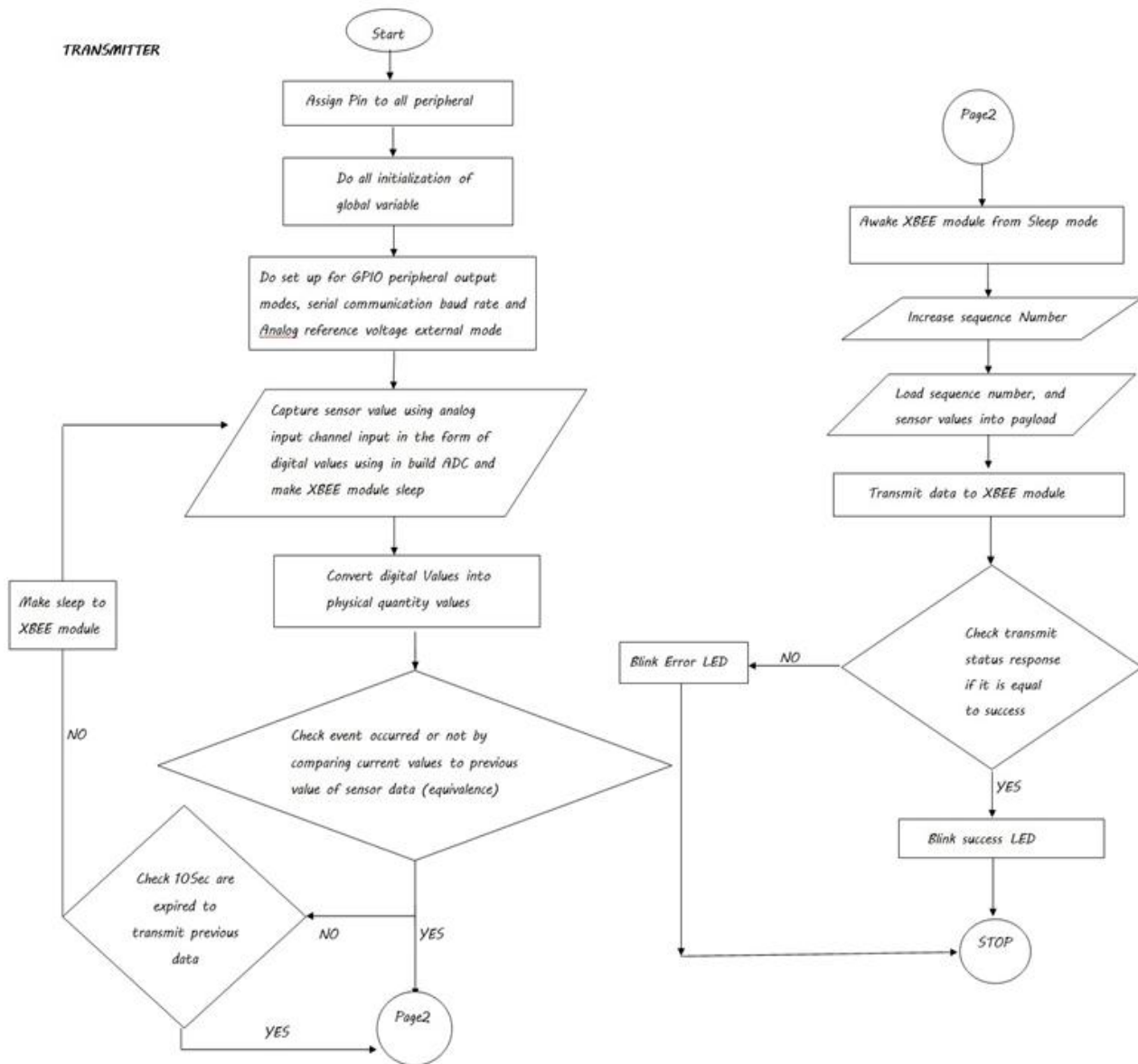


Figure 4.3 Sensor nodes flow chart data transmission algorithm

At the side of the data transmitter, the sensor nodes are present that help in converting the physical quantities like the temperature, pressure, vibration etc. to the electrical parameters such as voltage.


```
// read the sensor for Temperature Value
sensorValue = analogRead(sensorPin);
sensorValue = map(sensorValue, 0, 1023, 0, 255);
currentData=(sensorValue);
if(timerCounter==10)
{
  tseqnumber=tseqnumber+1;
  digitalWrite(sleepPin, LOW); // For Wake up from sleep mode
  digitalWrite(GLed, HIGH); // To indicate Xbee is awake
  sendData(); // Send data to Xbee with TX Request
  previousData=currentData; // Store current value to compare in next capture
  timerCounter=0;
}else if(currentData != previousData) // Check whether data change takes place or no
{ // If yes then Wake Up
  tseqnumber=tseqnumber+1;
  digitalWrite(sleepPin, LOW); // For Wake up from sleep mode
  digitalWrite(GLed, HIGH); // To indicate Xbee is awake
  sendData(); // Send data to Xbee with TX Request
  previousData=currentData; // Store current value to compare in next capture
  timerCounter=0;
}
}else{ // Go to Sleep
  digitalWrite(sleepPin, HIGH); // Go to Sleep
  digitalWrite(GLed, LOW); // Indicate Xbee Sleeping
}
}
delay(1000);
timerCounter++;
}
```

Figure 4.4 code for reading the sensor data, to have periodic, event transmission

Reads the value from analog pin. The received analog voltage is converted to the digital forms of the values that are used in processing the data. In order to convert the data from analogue form into the digital form, the ADC Arduino board is used. The conversion logic for every individual sensor changes as per sensor specification provided by data sheet. We have set the time counter = 10 to have periodic communication for every 10 seconds then adds the sequence number for each 10 second transaction. To have event communication occurs comparing the previous sensor value with the current sensor value.

The packets are generated with all the required specifications along with the sensed actual data of the sensor. This packet is then transmitted to the module with the help of XBee. We used transmit request API frame type used to transmit the data to coordinator node. By using code in below figure shows declaring the pay load size, according specification data should contain 8bytes of data to carry sensor data, sequence number the unit for sensor value and node id. source address of the coordinator node is given to for packet transmission. The transmission request is sent to coordinator node carries the node source address, pay load and size of pay load.

```
uint8_t payload[] = {0,0,0,0,'C','1'};  
XBeeAddress64 addr64 = XBeeAddress64(0x0013a200, 0x4079cf68);  
ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));  
ZBTxStatusResponse txStatus = ZBTxStatusResponse();
```

Figure 4.5 Code for transmitting the packet to data receiver

So it becomes possible to cross check whether the transmitted packet is correct or not by using ZigBee transmit status response frame type.

This particular packet is collected and work on this data is done according to the requirements and then, it is transmitted serially to the computer where this data can be used. For processing of this data, the information that is required from the received packets is extracted and they change their representation which can be read by the humans very easily in order to modify it.

Data receiver side

As we know we are using API mode communication the data received to the coordinator node is in the form of HEXA characters, so it is necessary to process the data. The Design specifications at receiver side are

- Display the source address of sensor nodes. Sequence number is to be added for each received packet and display it on the serial monitor of PandaBoard gateway or PC
- Identify the node id ,node type and display it on serial port monitor on the serial monitor of PandaBoard gateway or PC
- Display the sensor data values on the serial monitor of PandaBoard gateway or PC.
- Display checksum values of each received packet on the serial monitor of PandaBoard gateway or PC

In order to satisfy the above conditions we build an application by using and ZigBee request response API frame type and ZigBee request I/O sample response frame type protocol. The main implementation about how this system will be working on the receiver side can be explained with. The implementation of this algorithm is discussed clearly in while preparing the coordinator nodes in implementation of our WSN model.

4.4 Implementation of our ZigBee WSN model

In implementation of ZigBee WSN first we will prepare our XBee modules in XBee API mode for both data receiver side and data Transmitter side. Secondly the preparation of constant 3.3 volts power supply jack is which is needed for sensor boards, and preparation of sensor boards to know how sensors are connected to LilyPad Arduino microcontroller board and to our XBee module, then by using programming language C, we build an application sketch for each sensor node on Arduino IDE and upload it to the board. Lastly we will prepare coordinator node which explains how the Arduino board 2560 micro controller board is connected to XBee module and we build an application sketch on Arduino IDE and upload it to the board. Here is the list of components used to build our WSN as shown in below table.

Serial no.	Components list	Quantity	Purpose
1	XBee s2	5	Used to transmit the sensor information between the nodes
2	XBee Explorer	1	Used to configuring the XBee Modules by using XCTU application
3	LilyPad XBee	4	Used to mount remote XBee module
4	LilyPad Arduino	4	Consists of AT mega 326 processor, used to connect sensor and for sensor programming.
5	FTDI basic break out	1	Used to connect LilyPad Arduino to upload the Arduino sketch of sensors
6	Arduino mega 2560	1	Use to processing the data to connect coordinator node
7	XBee Shield	1	Used to connect XBee coordinator node with to Arduino mega 2560
8	9 volts power supply	4	Used to give power supply to the remote sensor nodes
9	UTC1117 VDC	4	3.3 voltage regulator
10	10 μ F	8	Used in the remote sensor node circuit to regulate the 3.3v constant power supply
11	Breadboard	4	Used to connect sensors and the LilyPad Arduino while preparing the remote sensor boards
12	Hookup wires	Few	used while connecting the remote node sensor board
13	Resistors 1K Ω , 4.7 K Ω	1	Used in remote node circuits

Table 4.1 List of components used to build our ZigBee WZN module

4.4.1 Preparing the XBee Modules for Data Transmitter & Receiver Side

By using X-CTU application we configure our XBee modules to have ZigBee communication between the data transmitter side and data receiver side. In our project we use four XBee modules at data transmitter side and one at data receiver. First we prepare XBee module for data receiver side, it is easy to know the receiver address, while configuring the XBee modules for data transmitter side. Here are the lists of components used to configure XBee modules.

4.4.1.1 Preparing coordinator API node

Take one XBee Module and make a note of its address which is found on the backside of module and our module address is (00132A00, 4079CF68). The instructions to configure the coordinator API node are:

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

- 1) Connect XBee explorer to USB port of computer via USB cable and then Mount XBee S2 module on the top of it.
- 2) Select the XCTU application and run it.
- 3) On the window of the application, select the component and examine the baud, flow control data bits and then run the query.

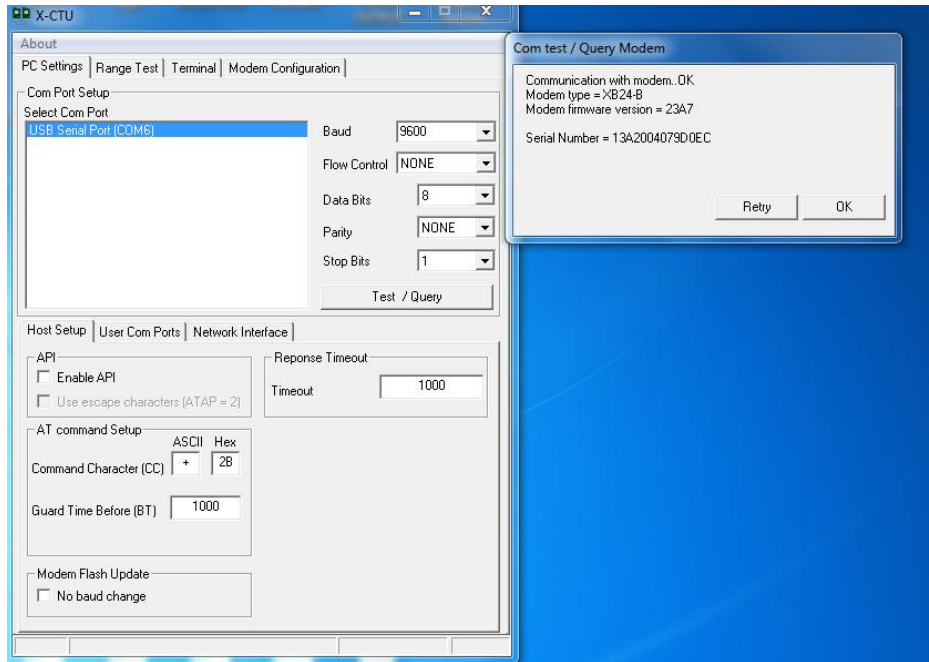


Figure 4.6 Modem firmware version of remote XBee module

- 4) The contest/query modem consists of the modem type as firmware version as shown in Figure 4.6 along with the serial number as 64 and 16 which is the device address.
- 5) Select the tab modem configuration and click the 'read' button to see the default values of the XBee module then set the parameters as shown in the Figure 4.7.

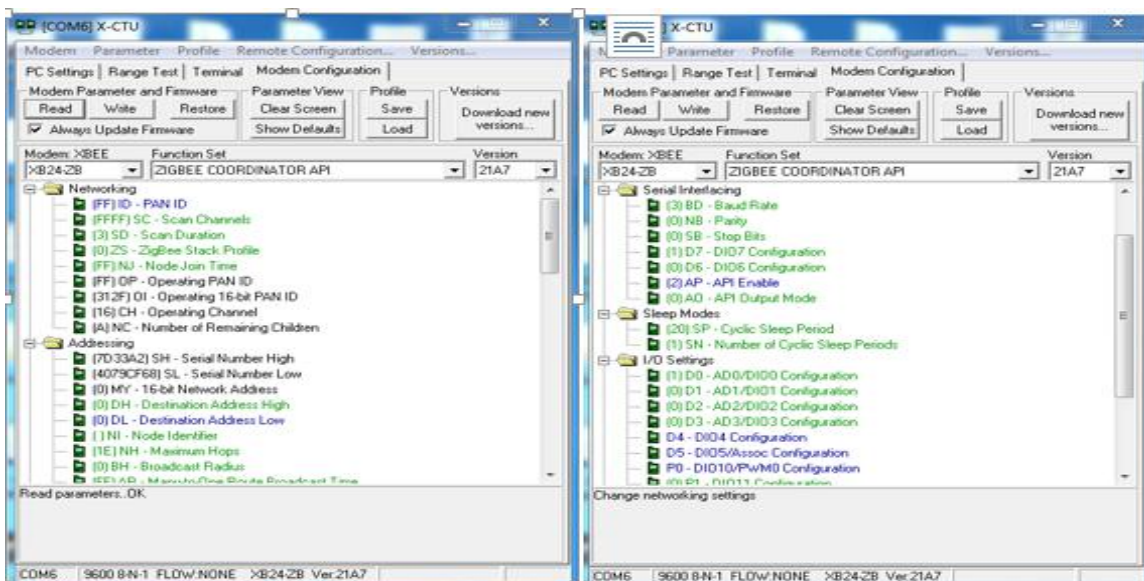


Figure 4.7 Modem configurations for coordinator API

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

- 6) Select the tab modem configuration and click the 'read' button to see the default values of the XBee module then set the parameters as shown in the Figure 4.7. The PAN ID and scan channels should be the same because the nodes in the network will identify with PAN ID
- 7) Return back to the PC settings and select the option to enable API checkbox.
- 8) Then again come to the modem configurations and click the 'write' button.
- 9) The following message is received after configuration as shown in Figure 4.8
- 10) Now remove the XBee module and label it as coordinator node

Note: label the configured XBee module with coordinator node to avoid confusion with other XBee modules

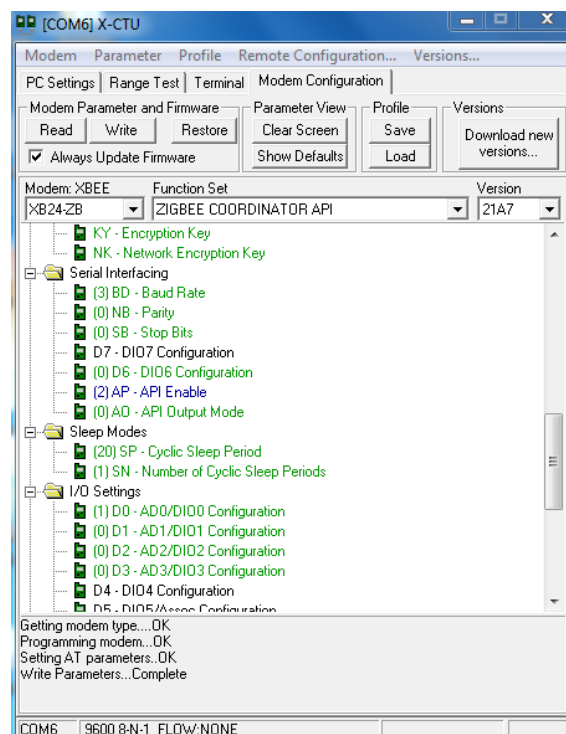


Figure 4.8 Successful modem configuration message for coordinator XBee module

4.4.1.2 Preparing the XBee module for data transmitter side

Now we will configure our XBee module as ZigBee router API device which is used as a remote sensor node in our WSN and helps in transmitting the data too coordinator.

The instructions to configure the XBee module as remote sensor node are:

- 1) Connect XBee explorer to USB port of computer via USB cable and then Mount XBee S2 module on the top of it.
- 2) Select the XCTU application and run it.

- 3) On the window of the application, select the component and examine the baud, flow control data bits and then run the query.

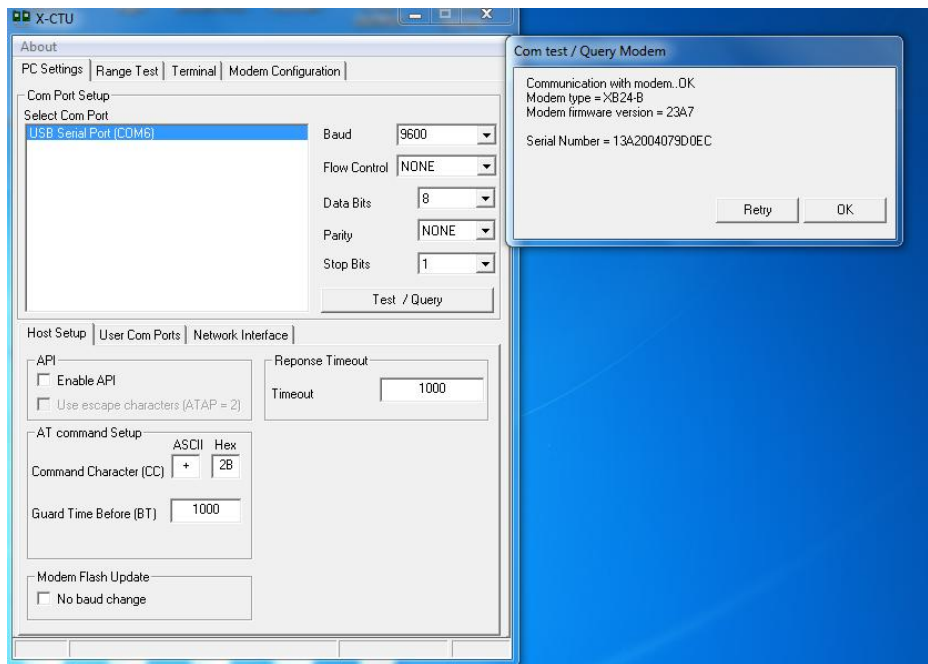


Figure 4.9 Modem firmware version of remote XBee module

- 4) The contest/query modem consists of the modem type as firmware version as shown in Figure4.9 along with the serial number as 64 and 16 node.
- 5) Select the tab modem configuration and click the 'read' button to see the default values of the XBee module then set the parameters as shown in the Figure 4.10.
- 6) Make sure that the PAN ID and SC are same as what we used while configuring the coordinator node. In our WSN Model PAN ID is FF and SC is FFFF. The DH, DL is the 64 bit address and 16bit address of coordinator device.
- 7) Return back to the PC settings and select the option to enable API checkbox.
- 8) Then again come to the modem configurations and click the 'write' button.
- 9) The following message is received after configuration as shown in Figure4.11.

NOTE: label the configured XBee module with Node1 node to avoid confusion with other XBee modules.

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

Similarly follow the same steps for configuring the other three XBee modules and label them as Node 2, Node 3 and Node 4.

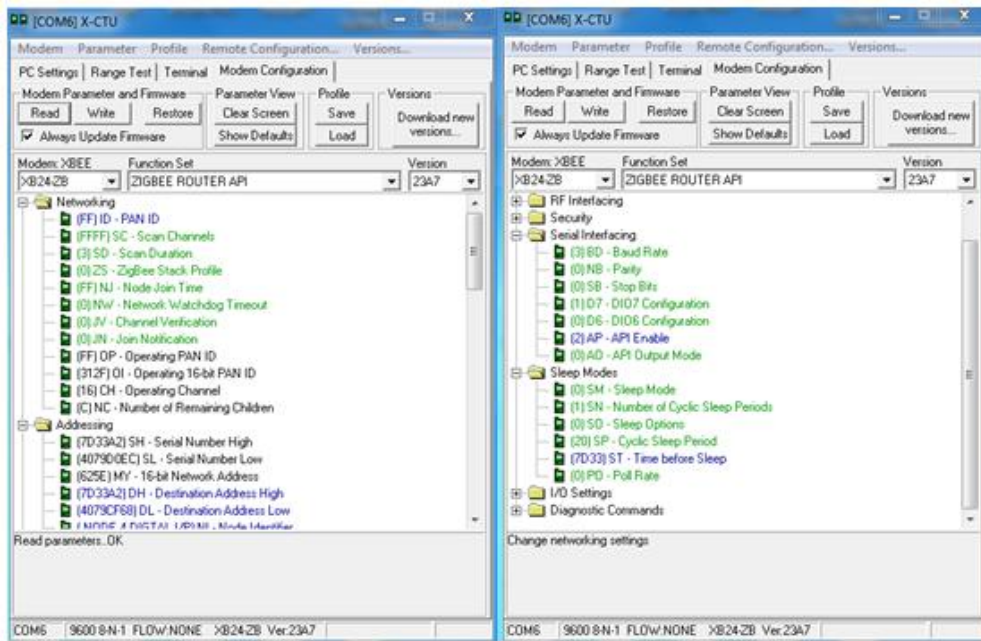


Figure 4.10 Mode configuration parameter settings for XBee remote Module with parameters



Figure 4.11 Successfully modem configuration message for remote XBee module

4.4.2 Preparing the Remote Sensor Boards for Data Transmitter Side

The following steps are followed by every node as a precautionary method for checking the final circuit:

- 1) The breadboard is set up with the wires of 3.3V with the voltage regulator that we use, i.e. UTC1117 voltage converter, and the circuit is shown in Figure 4.12. This regulator consists of three pins, which are for input, output, and ground. Get the ground along with the blue rails going along the breadboard. And get the output power of 3.3V to both red rails.

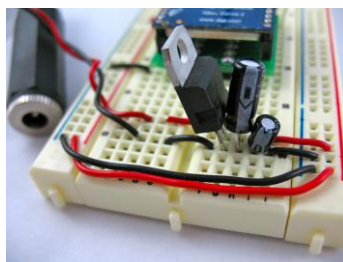


Figure 4.12 Constant 3.3 volts power jack breadboard [36]

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

- 2) The red wire is to be soldered to the power jack's center pin, and similarly, the black wire is connected to the outer pin that is longer.
- 3) Join the power jack's red wire with the help of the breadboard so that it can be connected to the input pin of the voltage regulator. Similarly join the black wire to the regulator.
- 4) Connect the output pin of the regulator to the breadboard's power rail with the help of red wire. Connect the ground pin to the ground rails.
- 5) To remove the low frequency noise coming from the wall power supply and to prevent the noisy power reaching the radio and its signal interface, we have to decouple the power supply by using $10\ \mu\text{f}$ capacitors as shown Figure4.13.

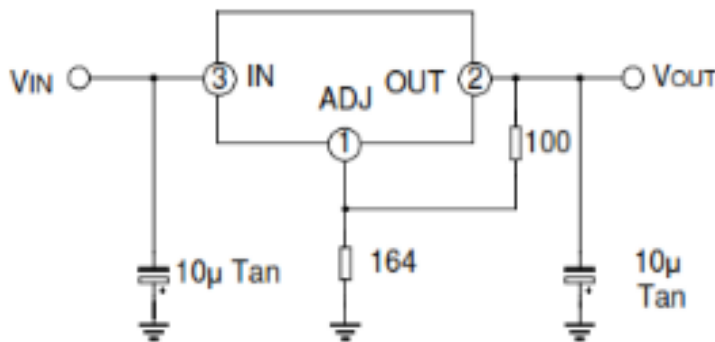


Figure 4.13 Decoupling the circuit voltage [59]

4.4.2.1 Preparing node1 LM35 analog temperature sensor board

The lm35 analog sensor board will act a node1 at data transmitter side in our WSN module. The main purpose is to measure temperature values, gives the analog output for the input, in the form of $^{\circ}\text{C}$. The output voltage differs to $10\text{mV} / ^{\circ}\text{C}$. The temperature range of LM35 can operate in between -50° to $+140^{\circ}\text{C}$. The circuit diagram to connect LilyPad Arduino micro controller board is shown in the Figure 4.14 which is done on the bread board.

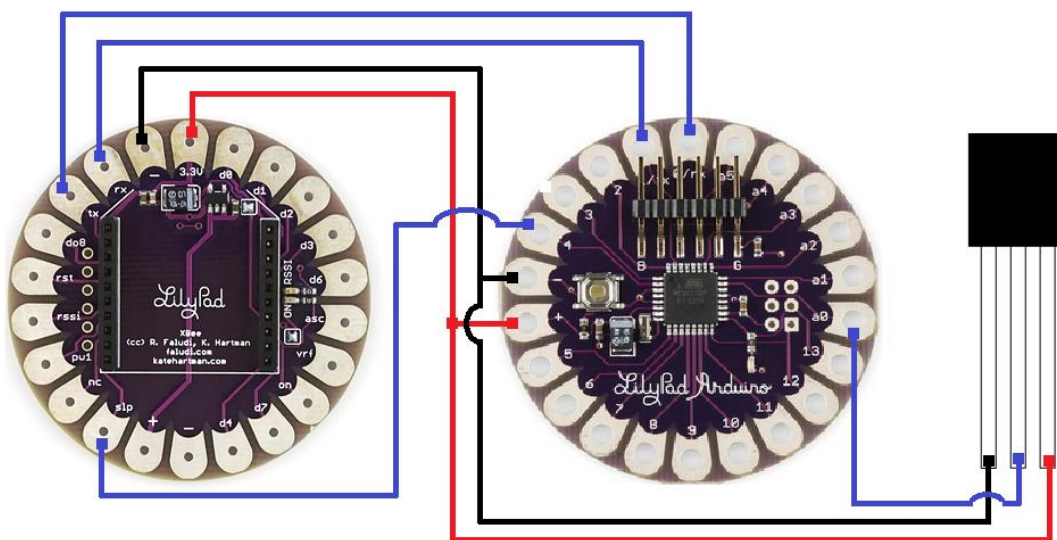


Figure 4.14 LM35 Analog temperature sensor circuit with respect to LilyPad Arduino

If we see the sensor from top view, the left first pin represents 'Vin', middle pin is output voltage and last right pin is ground. By using bread board the Vout pin of sensor is connected to the A0 pin of LilyPad Arduino to read the sensor data, The Txx pin, Rxx pin of LilyPad Arduino are connected to 'Rxx' pin, 'Txx' pin of LilyPad XBee to exchange the data between the two boards. The pin connections between the sensor, LilyPad Arduino, and LilyPad XBee is show in the table.4.2.power up the circuit by connecting constant 3.3volt pin 1 of LilyPad XBee to constant 3.3 output voltage of power jack circuit and pin 10 of LilyPad XBee to ground .

LilyPad XBee	LilyPad Arduino	breadboard
Txx pin 3 for data out	Rxx pin 0	
Rxx pin 2 data in	Txx pin	
+ve 3.3 volts pin 1	+ve pin	
Pin 10 gnd	-ve pin	
Slp pin 9 used to make XBee sleep and awake	Pin 4	
	Pin A0 analog pin reads analog data from Vout pin sensor	Vout pin
	Pin5	+ve pin red Led
	Pin 6	+ve pin of green led
	Pin 7	+ve pin of yellow led

Table 4.2 Pin connections of Im35 sensor with respect to LilyPad Arduino and LilyPad XBee

After making all the connections on bread board, take XBee module which was labelled as node 1 connect on the top of LilyPad XBee. With this our LM 35 remote sensor board looks like as shown in the Figure 4.15.

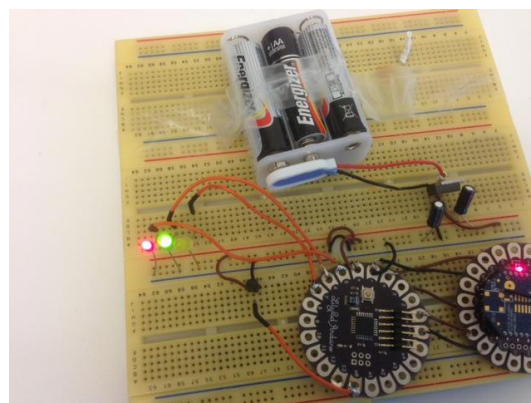


Figure 4.15 Node 1, Im35 Analog temperature remote sensor

Build & Upload the sketch to sensor node 1 LilyPad Arduino microcontroller board.

As we discussed during design algorithm for data transmitter side, the functionality of each node is same but differs in converting the analog sensor values to real data values .the conversion of sensor analog output value to real data value is explained in below code as shown in figure 4.16

```
sensorValue = analogRead(sensorPin);  
sensorValue = map(sensorValue, 0, 1023, 0, 255);  
currentData= (sensorValue);
```

Figure 4.16 Code to convert sensor data to real data for Im35 analog sensor

The analog pin A0 of LilyPad Arduino microcontroller board reads the sensor value and converts it into digital values for 10bit resolution ADC. Map the digital values between 0 to 255.store the sensor value to current and this current data is transfer with our algorithm.

Now open the Arduino IDE terminal on Computer make necessary changes in converting the sensor values to real data values in our algorithm and upload the code to LilyPad Arduino board. We have to make sure that we choose correct board and the serial port .to upload the code we need to connect serial port pin of Arduino to Ftdi basic board using USB cable. The complete code for node 1 is referred in attachment file which is named Node 1 Data Transmission.

4.4.2.2 Preparing Node 2 Flexi Force Pressure Sensor Board

Now we are preparing node 2 flexi force pressure sensor is also an analog sensor the output of this sensor is in the form of voltage. These are mostly used for calculating calculating the forces with no disturbance caused to the test of dynamics. They can measure dynamic as well as static forces. As the force applied increases, the resistance decreases. As per the data sheet the external Vref is 4.5 volts, the pressure value range varies from 0 to 110 newton. Digital value range is (0, 1023).By using bread board connect fliforce pressure sensor to LilyPad Arduino and LilyPad XBee is shown in Figure 4.13 .the pin connections are explained in Table4.3.

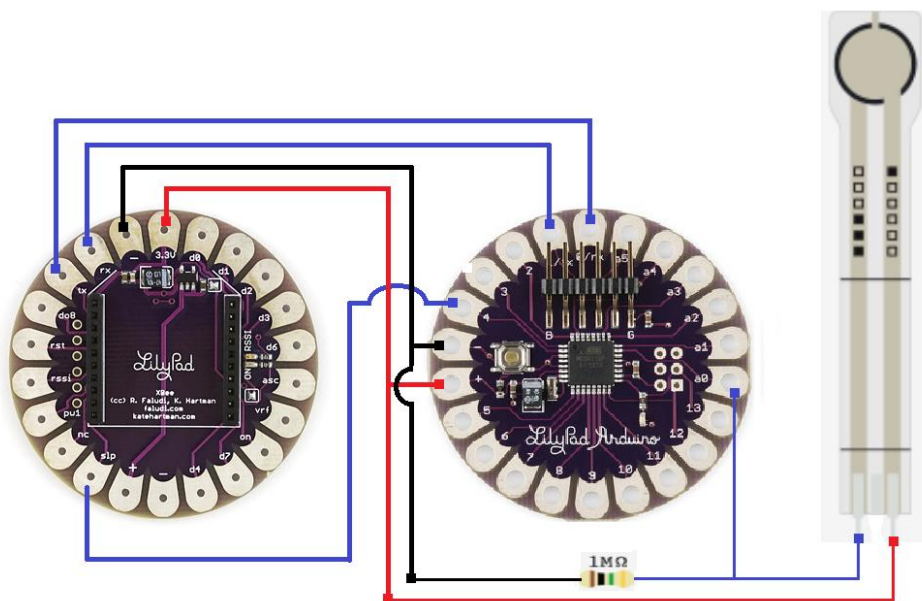


Figure 4.17 Node 2, flexi force pressure sensor circuit with respect LilyPad Arduino

LilyPad XBee	LilyPad Arduino	breadboard
Txx pin 3 for data out	Rxx pin 0	
Rxx pin 2 data in	Txx pin	
+ve 3.3 volts pin 1	+ve pin	
Pin 10 gnd	-ve pin	
Slp pin 9 used to make XBee sleep and awake	Pin 4	
	Pin A0 analog pin reads analog data from Vout pin sensor	Vout pin
	Pin 11	+ve pin red Led
	Pin 12	+ve pin of green led
	Pin 13	+ve pin of yellow led

Table 4.3 Node 2, Flexiforce pressure sensor pin connections with respect to LilyPad Arduino, LilyPad XBee and bread board

power up the circuit by connecting constant 3.3volt pin 1 of LilyPad XBee to constant 3.3 output voltage of power jack circuit and pin 10 of LilyPad XBee to ground.

After making all the connections on bread board, take XBee module which was labelled as node 2 connect on the top of LilyPad XBee. With this our node 2 flexiforce pressure sensor board looks like as shown in the Figure4.18

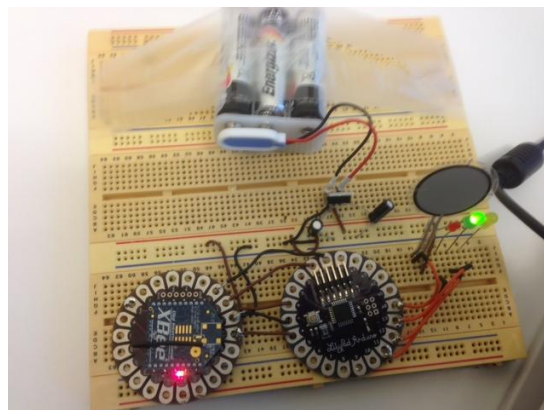


Figure 4.18 Node 2 flexi force pressure sensor

Build & Upload the sketch to sensor node 2 LilyPad Arduino microcontroller board.

As we discussed during design algorithm for data transmitter side, the functionality of each node is same but differs in converting the analog sensor values to real data values. The conversion of sensor analog output value to real data value is explained in below code as shown in figure 4.19

```
sensorValue = analogRead(sensorPin);  
  
sensorValue = ((110.0/1023.0)*sensorValue);  
  
currentData = (sensorValue);
```

Figure 4.19 Code to convert sensor data to real data for node 2 pressure sensor

Now open the Arduino IDE terminal on Computer make necessary changes in converting the sensor values to real data values in our algorithm and upload the code to LilyPad Arduino board. We have to make sure that we choose correct board and the serial port. To upload the code we need to connect serial port pin of Arduino to Ftdi basic board using USB cable. The complete code for node 2 is referred in attachment file which is named as Node 2 Data Transmission

4.4.2.3 Preparing node 3 vibration sensor board

Now we are using vibration sensor to prepare node 3 which is also an analog sensor. These are used for measuring, displaying, and then investigating the displacement, velocity, and acceleration. They can be used as a single instrument or also in the combination with the system of data acquisition. These vibration sensors are available in various formats. They are available as the elements used for sensing which are raw, transducer package or otherwise as the instrument or the system consisting the features like the calculating, rare or local display and storing data. Use bread board to the circuit diagram with respect to LilyPad Arduino is shown in the Figure 4.20 and the pin connections of sensor with respect to LilyPad Arduino, LilyPad XBee and explained in Table 4.4. Power up the circuit by connecting constant 3.3 volt pin 1 of LilyPad XBee to constant 3.3 output voltage of power jack circuit and pin 10 of LilyPad XBee to ground.

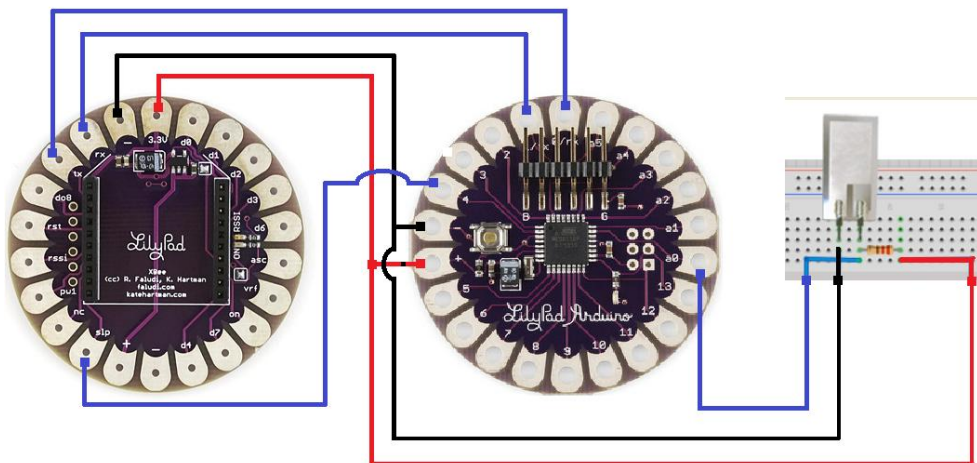


Figure 4.20 Node 3, vibration sensor circuit with respect LilyPad Arduino

LilyPad XBee	LilyPad Arduino	breadboard
Txx pin 3 for data out	Rxx pin 0	
Rxx pin 2 data in	Txx pin	
+ve 3.3 volts pin 1	+ve pin	
Pin 10 gnd	-ve pin	
Slp pin 9 used to make XBee sleep and awake	Pin 4	
	Pin A0 analog pin reads analog data from Vout pin sensor	Vout pin of sensor
	Pin 11	+ve pin red Led
	Pin 12	+ve pin of green led
	Pin 13	+ve pin of yellow led

Table 4.4 Node 3 Pin connections of vibration sensor with respect to LilyPad Arduino and LilyPad XBee

After making all the connections on bread board, take XBee module which was labelled as node 3 connect on the top of LilyPad XBee. With this our node 3 vibration sensor board looks like as shown in the Figure 4.21.

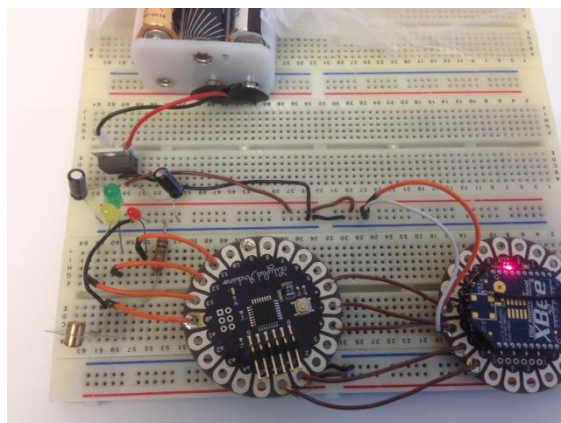


Figure 4.21 Node 3, vibration sensor

Build & Upload the sketch to node 3 LilyPad Arduino microcontroller board.

As we discussed during design algorithm for data transmitter side, the functionality of each node is same but differs in converting the analog sensor values to real data values .the conversion of sensor analog output value to real data value is explained in below code as shown in figure 4.12

```
sensorValue = analogRead(sensorPin);  
  
sensorValue=( (330.0/1023.0)*sensorValue);  
  
currentData=(sensorValue);
```

Figure 4.22 Code to convert sensor data to real data for node 3 vibration sensor

Now open the Arduino IDE terminal on Computer make necessary changes in converting the sensor values to real data values in our algorithm and upload the code to LilyPad Arduino board. We have to make sure that we choose correct board and the serial port .to upload the code we need to connect serial port pin of Arduino to Ftdi basic board using USB cable. The complete code for node 3 is referred in attachment file which is named Node 3 Data Transmission.

4.4.2.4 Preparing node 4 one wire digital temperature sensor board

In preparation of node 4 we used one wire digital temperature sensor and the value ranges from -55C to 125C it allows 2.7to 5vls of input power supply circuit diagram with respect Arduino microcontroller board is shown in the Figure 4.23

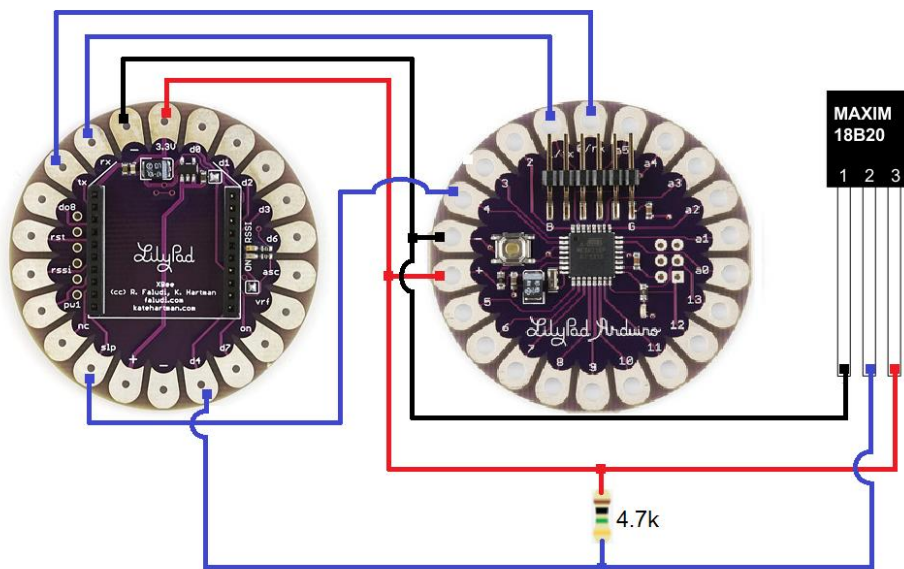


Figure 4.23 Node 4, one wire digital temperature sensor circuit with respect to LilyPad Arduino

In this circuit, we connect 4.7 Ohms pull-up-resistor it has one digital pin which is connected to digital input pin 9 of LilyPad Arduino. The pin connections with respect to LilyPad Arduino and LilyPad XBee are explained in the table 4.5. Power up the circuit by connecting constant 3.3 volt pin 1 of LilyPad XBee to constant 3.3 output voltage of power jack circuit and pin 10 of LilyPad XBee to ground.

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

LilyPad XBee	LilyPad Arduino	breadboard
Txx pin 3 for data out	Rxx pin 0	
Rxx pin 2 data in	Txx pin	
+ve 3.3 volts pin 1	+ve pin	
Pin 10 gnd	-ve pin	
Slp pin 9 used to make XBee sleep and awake	Pin 4	
	Pin 9 Digital pin	Digital pin pin
	Pin5	+ve pin red Led
	Pin 6	+ve pin of green led
	Pin 7	+ve pin of yellow led

Table 4.5 Node 4, one wire digital temperature sensor pin connections

After making all the connections on bread board, take XBee module which was labelled as node 4 connect on the top of LilyPad XBee. With this our node 4 one wire digital sensor board looks like as shown in the Figure 4.24.

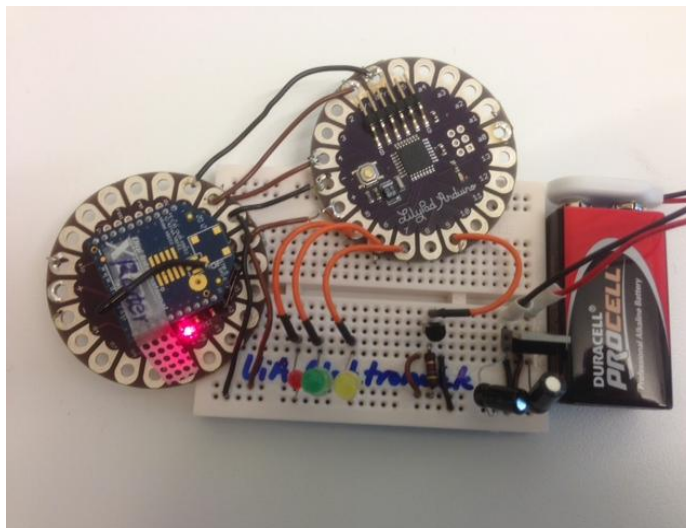


Figure 4.24 Node 4, one wire digital temperature sensor

Build & Upload the sketch to node 4 LilyPad Arduino microcontroller board.

Now open the Arduino IDE terminal on Computer make necessary changes in converting the sensor values to real data values in our algorithm and upload the code to LilyPad Arduino board.

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

As node 4 is a one wire digital temperature sensor we used one wire library file provided in examples of the Arduino IDE for converting the real sensor data. We have to make sure that we choose correct board and the serial port. To upload the code we need to connect serial port pin of Arduino to Ftdi basic board using USB cable. The complete code for node 4 is referred in attachment file which is named as Node 4 Data Transmission.

4.4.3 Preparing coordinator node for data collection and processing the data

As we know the remote sensor nodes will transmit the data in the form of XBee API transmit request frame format, the output of the received node on coordinator side is always ASCII or HEXA characters which is not readable for human beings and the serial port terminals of various operating systems. In order to overcome this situation we have to process the data, for this we have to connect coordinator XBee node to the Arduino mega2560 using XBee shield and Dout pin of XBee shield is connected to rx3 pin on Arduino mega2560 as shown in the following Figure 4.25.

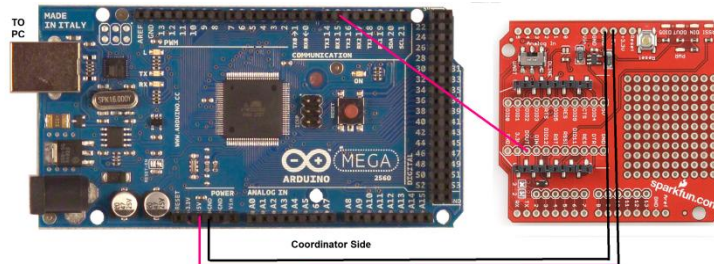


Figure 4.25 Coordinator Arduino XBee

As per our design specification of our ZigBee WZN model we developed an algorithm using Arduino IDE and the specifications are:

- Display the source address of sensor nodes.
- Sequence number is to be added for each received packet and display it on the serial monitor of PandaBoard gateway or PC
- Identify the node id ,node type and display it on serial port monitor on the serial monitor of PandaBoard gateway or PC
- Display the sensor data values on the serial monitor of PandaBoard gateway or PC.
- Display checksum values of each received packet on the serial monitor of PandaBoard gateway or PC

We used ZigBee receive request API frame type, I/O data sample Rx indicator API frame type to receive the data from sensor nodes and to process the data. The data receiver algorithm is explained in a flow chat as shown in the Figure 4.26. The code is attached as an attachment file named as Data receiver and processing.

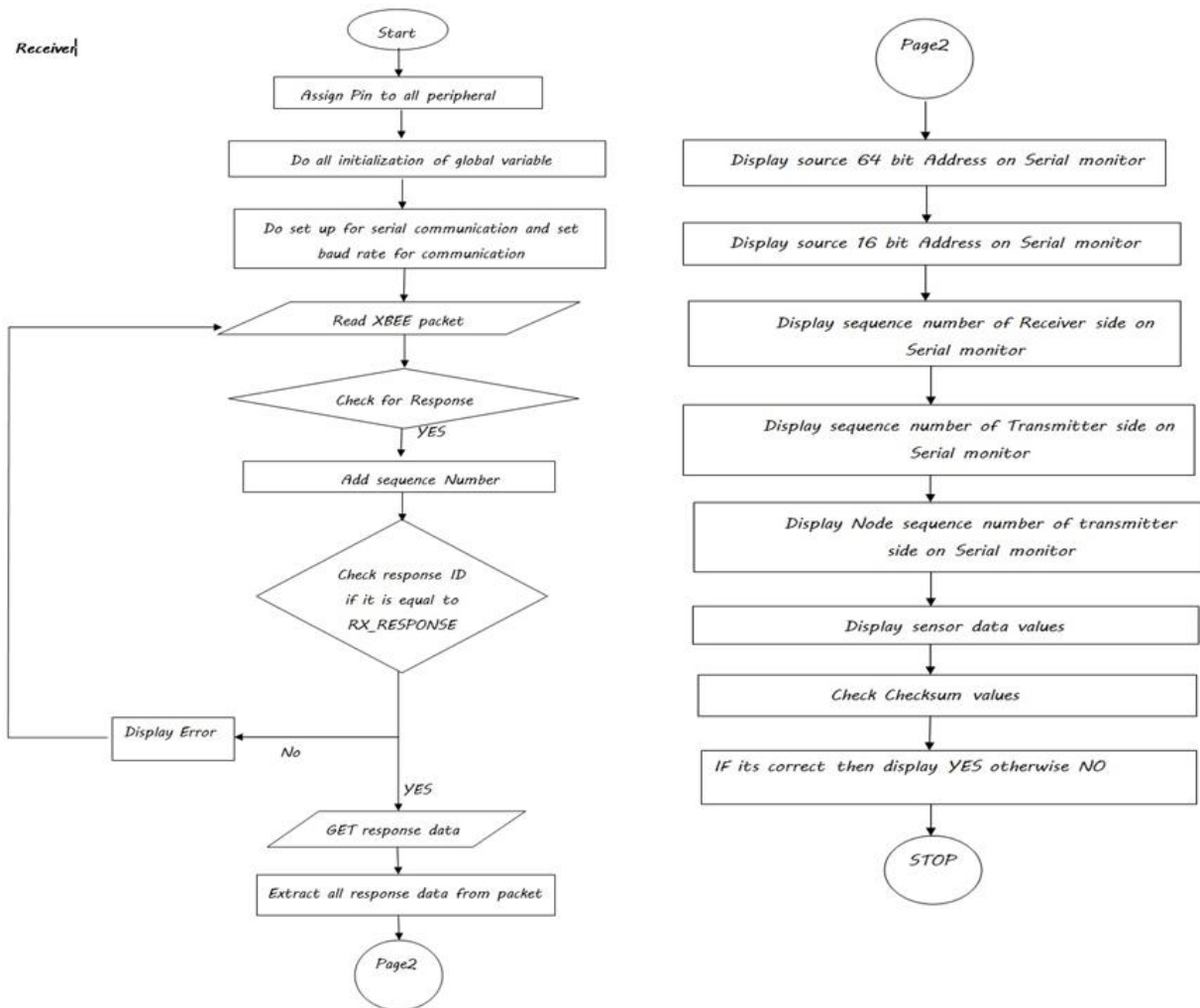


Figure 4.26 Coordinator sensor data processing algorithm

By using USB cable the coordinator Arduino mega 2560 is connected to USB port of PandaBoard.

4.5 Data Storage and Transmitting to Remote Server

In this part, we focus on the storing data to the database and sending this data to the remote server. To achieve these goals, three problems are needed to be solved. The first problem is to design and create a database in both local and remote side. The second one is how we sniffing the data for the coordinator node and then analyse it. The last problem is to transmit the data from the local gateway to the remote server.

We achieve these three goals in one program and flow chart is shown in the Figure 4.27. The whole program codes can be found in attachment which is called "sniffer_xbeeRe".

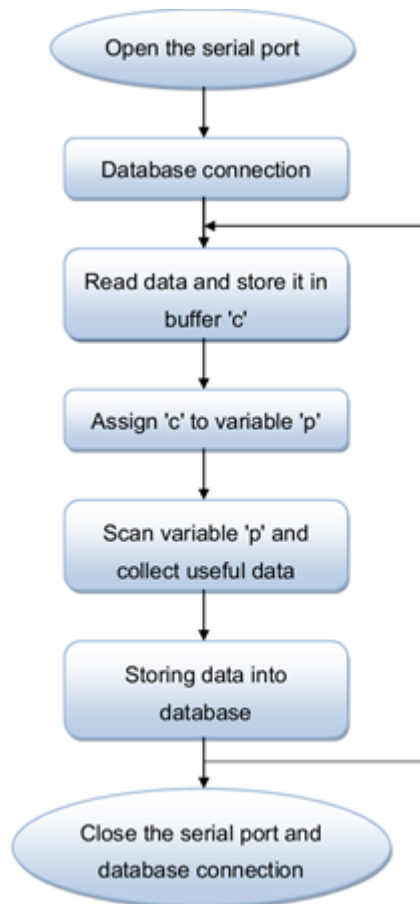


Figure 4.27 Flow chart of data base architecture

The serial port is opened first and then database connection will be established. After that, the program will enter a loop which includes sniffing the data and storing it. In this loop, the first step is reading the data from the serial port and stores it for further use. After that, the program will scan the stored data and collect useful data from it. The last step is to store the collected data into the database.

4.5.1 Data base architecture and design

In this system, every node has one sensor. That means we just get only one value from each node. Thus, we can use a very simple table which has only two fields to store the data. These fields are Note Id and Value. This is basic request from the database but it's enough for future use. In our project, we have 3 different sensors now which are temperature, pressure and vibration. For

distinguish different value from different type sensor, we need a new field in the database which is Data Type. After that we find that we may have many boats which will carry so many sensors on it. Then we need to add another field to specify which boat is the owner for the sensor. This new field is named Boat id. In the end, we need to know when we get the data. Thus we use the field Date to declare the timestamp for each message. This timestamp shows the record time based on the clock of database.

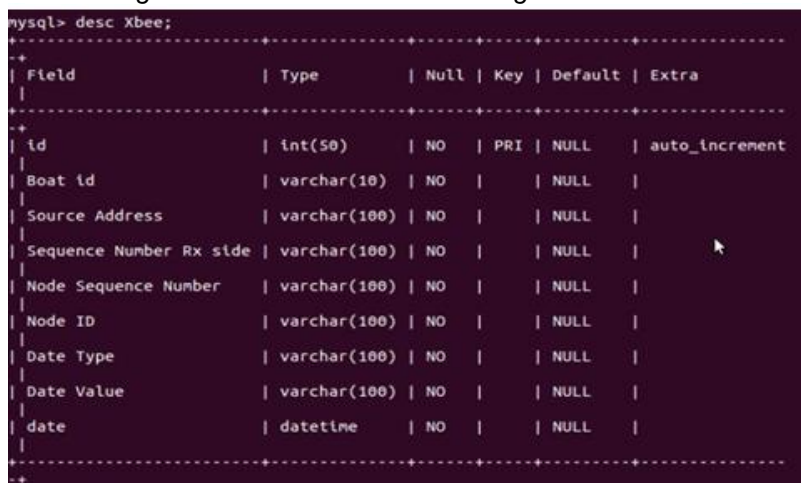
For now, we have five fields in one table. All this fields are used to help us read the data and show a brief state of a boat. However, we don't have anything for confirming the communication part. If there is any message missing or dropped, we can't read it from the database. Thus we add two more fields to achieve this. The first field is Node Sequence Number. This parameter is from every node. It starts from one and continuously increases in every message. It shows how many messages are send from this node. The second field named Sequence Number Rx side. This parameter comes from the coordinator note. This number will be recounted after we restart the coordinator while the Node Sequence Number is continuously increased.

In the end, we find that in the XBee system, the note name is very complicacy which is composed with 20 alphabet and number. So we put a simple name like note 1 to every note. At the same time, we also keep the original name for double check.

After all, the database has 8 fields which is Boat id, Source Address, Sequence Number Rx Side, Note Sequence Number, Note ID, Date Type, Date value and Date. Therefore, the database table is created by the following codes shown in Figure 4-28 and the architecture of table is shown below in Figure 4-29. The codes are written in MySQL language.

```
CREATE TABLE IF `Xbee` (  
  `id` int(50) NOT NULL PRIMARY KEY auto_increment,  
  `Boat id` varchar(10) NOT NULL  
  `Source Address` varchar(100) NOT NULL,  
  `Sequence Number Rx side` varchar(100) NOT NULL,  
  `Node Sequence Number` varchar(100) NOT NULL,  
  `Node ID` varchar(100) NOT NULL,  
  `Data Type` varchar(100) NOT NULL,  
  `Data Value` varchar(100) NOT NULL,  
  `date` datetime NOT NULL,  
);
```

Figure 4.28 Codes used in creating data base



```
mysql> desc Xbee;
```

Field	Type	Null	Key	Default	Extra
id	int(50)	NO	PRI	NULL	auto_increment
Boat id	varchar(10)	NO		NULL	
Source Address	varchar(100)	NO		NULL	
Sequence Number Rx side	varchar(100)	NO		NULL	
Node Sequence Number	varchar(100)	NO		NULL	
Node ID	varchar(100)	NO		NULL	
Date Type	varchar(100)	NO		NULL	
Date Value	varchar(100)	NO		NULL	
date	datetime	NO		NULL	

Figure 4.29 Architecture of database table in PandaBoard

Both local and remote database table is designed in the same architecture.

4.5.2 Data sniffing

In this part, we are facing two main problems. The first problem is that every node will send their data continuously. We need to distinguish the specific message in a data flow one by one that we can ensure the whole data is collected. At beginning, we try to use the length of the data to cut the message. For example, we cut the first 150 bytes as one message and store it in a variable. But this is not the right solution. Because we have 3 kinds of sensors and the data length from each sensor are different. Even the same sensor will send data in different length. For example, when the temperature increased for 5 to 15, the data length will increase 1 byte. After that, we use identifier to signify the end of one message. We use the end symbol '\0' to show that the message is end and then store it in the buffer because this symbol will appear in the end of the message by default. But we failed again in the test. We still can't recognize the end of the message and the program will cut the data randomly. Then we try the same method but use another symbol to test again. In this time, we add a symbol '!' in the end of every message manually. In this way, data is read from the serial port by every byte and stored in the buffer. If the identifier '!' is appeared, the program will stop read and enter the next phase.

The second phase is that we need to analysis message and collect the useful data to storing in the database. Since we can use identifiers to cut the message, we use the same method to collect the data. In one message, there are 6 data which we need to collect. We insert the symbol ':' in the beginning of the data while the end of data is the newline symbol '\n'. The data between these two identifiers will be collected and stored in different variables. In this way, the program will collect the useful data and store it. The main codes of this part are shown below.

```
read(sd, &c, 1);
p[j] = c;
j++;
if (c == '!') {
    p[j] = '\0';
    sscanf(p, "%[^:]:%[^\\n]*[^:]:%[^\\n]*[^:]:%[^\\n]*[^:]:%[^\\n]*[^:]:%[^\\n]*[^:]:%[^\\n]*",
           &SA[0], &SN[0], &NID[0], &type[0], &val[0], &Csum[0]);
}
```

Figure 4.30 Codes for data sniffing

Two functions 'read' and 'sscanf' are used here. In the function read, 'sd' declares the source of the data, 'c' is the buffer and 1 means read 1 byte every time. The effect of this function is read the serial port 'sd', and stores the data into the buffer 'c' by each one byte. Then, we store the data into a variable 'p'. If the identifier '!' appears, the program will be entered the next phase and run the function 'sscanf'. In this function, the variable 'p' will be scanned. The function will ignore the data until the first identifier ':' and then store the data into a variable after that. When the next identifier '\n' appears, the storing is stopped and waiting for the next beginning identifier. In this way, useful data will be collected and waiting for store in to the MySQL database.

4.5.3 Data storing

While storing data, we need to insert the data into 2 database tables at the same time. By using the MySQL api functions, the MySQL database is connected by using the following codes.

```
mysql_init(&mysql);
mysql_options(&mysql, MYSQL_READ_DEFAULT_GROUP, "your_prog_name");
if (!mysql_real_connect(&mysql, "127.0.0.1", "root", "uia", "cursoWasp", 0, NULL, 0)) {
    fprintf(stderr, "Failed to connect to database: Error: %s\n", mysql_error(&mysql));
    exit(-1);
} else {
    fprintf(stderr, "Connected to database: return: %s\n", mysql_error(&mysql));
}
```

Figure 4.31 Codes for database connection

In the third line, the codes declare the host IP, name, password and which database will be used. The IP '127.0.0.1' means that it is a local host. It should be changed when the connection is between the gateway and the remote server.

In this program, it connects to local and remote host at beginning, and then run the loop of data sniffing. Otherwise, if the connection phase is included in the sniffing loop, it will be repeated every cycle and wastes system consumption.

After connecting the database, the data which is collected before is inserted into database table by using following codes.

```
printf(sql, "insert into Xbee values (null,A,'%s','%s','%s','%s','%s','%s',now());"
        | SA, SN, NSN,NID, type, val);
if (mysql_query(&mysql, sql) != 0) {
    fprintf(stderr, "Failed:%s\n", sql);
} else {
    fprintf(stdout, "Succeeded:%s\n", sql);
}
```

Figure 4.32 Codes of insert data

In these codes, the function of the first two lines is to create the order of insert the data into the database table Xbee. In the third line, `mysql_query (&mysql, sql)` means that the local MySQL connection which is set before is used.

4.5.4 Remote Server

This server is done by the group 'Marine Data Collection based on Embedded System with Wired and Wireless Transmission'.

They use their own laptop as remote server. The machine model is ASUS N43SL with an Intel Core i5 processor, 4GB 1333 MHz RAM and 500GB storage. And the operating platform is Windows 7 Professional, which can support us to install all the software we need and work smoothly.

As we introduced in Figure 4.1, both embedded system and remote server have a MySQL database. Thus, similar as MySQL database in embedded system, they install MySQL (version 5.5.31 for Windows 7) to receive and store real-time data, which is sent from embedded system.

Since the default setting of access control in MySQL only allow the access from local IP address 127.0.0.1, we need to make the database can be accessed from any IP address. By entering command `update user set Host = % where User = 'root'`.

4.5.5 Data Transmission to Remote Server

In this part, the data which is collected before will transmit to the remote side. We have two ways to achieve this goal. The first one is real-time transmission. After the data is collected, the data will be stored in the local database which is created in the panda board. At the same time, we send the date to the remote database with the same format through Internet or 3G network. Since networks are not reliable every time and some package will be lost some time, we upload the whole local database to an ftp server for backup. This upload will be executed one time for each day.

To insert the data into remote side, a new database connection is created by the following codes.

```
mysql_init(&re_connection);
if (!mysql_real_connect(&re_connection, ReHOST, ReUSERNAME, RePASSWORD, ReDATABASE, 0, NULL, CLIENT_FOUND_ROWS)){
    fprintf(stderr, "Failed to connect to Redatabase: Error: %s\n", mysql_error(&mysql));
    exit(-1);
} else {
    fprintf(stderr, "Connected to Redatabase: return: %s\n", mysql_error(&mysql));
}
```

Figure 4.33 Codes of remote database connection

Since the IP address of remote server is changing, the IP address is declared before running this program. Since the insert order is made for the local database part, it can be used for the remote side by the following code.

```
if (mysql_query(&re_connection, sql) != 0) {
    fprintf(stderr, "Failed:%s\n", sql);
} else {
    fprintf(stdout, "Succeeded:%s\n", sql);
}
```

Figure 4.34 Codes of insert data to remote database

The codes 'mysql_query (&re_connection, sql)' determine the re_connection is used in this time. And the following code will show the inserting is succeeded or not.

5 Experiments and Testing Results

In this chapter we present the testing results of our designed WSN in three different scenarios which are shown below:

1. To connect WSN to gateway PC.
2. To process and store data on gateway PandaBoard.
3. To test on boat and integrate with CAN-bus system.

The first and second scenario is tested in UIA campus. For the first test, one digital sensor and three analog sensors are used to build the WSN. Collected data is transmitted to the gateway PC to check the results. In the second test, PandaBoard is used to be the gateway. Meanwhile data will be processed, filtered and stored in database. The results show that system requirements are successfully achieved. The third scenario is the most important one which is a real condition experiment. The experiment takes place on a leisure boat which is moving between two cities. After processing the data on gateway PandaBoard, data will be stored in local and remote databases at the same time. Furthermore, we have run our system with the CAN-bus system in one gateway Pandaboard at the same time. The result shows that two systems are integrated successfully.

5.1 Testing WSN at UIA

Four remote sensor nodes are placed in a college lab which the area is around 10 square meters. The coordinator node is connected to PandaBoard which acts as a gateway is placed in the same room. The whole setup can be shown in Figure5-1

Precautions to be taken before testing the network:

- 1) Check whether all the connections are done and all the components are placed according to the circuit layout.
- 2) Check there is no short circuit between the pin connections.
- 3) Check all the nodes properly mounted to respective remote sensor board.
- 4) Check whether the program is successfully uploaded to the respective LilyPad Arduino.
- 5) Sensor board Run Arduino and check the results.
- 6) Connect the 3.3v power jack to the circuit.
- 7) Connect the coordinator to the panda board and check whether the receiver program is uploaded to Arduino mega2560 coordinator.



Figure 5.1 WSN Testbed in college lab

5.1.1 Testing Procedure

Testing of our WSN can be done at both sides.

Test cases and results:

1. Check whether the received unprocessed output is similar to the ZigBee receiving packet frame type. Each new message starts with 7E.
2. Check whether the data is received is processed or not.
3. Check the received data is as per our data transmitter algorithm.
4. At coordinator side if we power off X-BEE module and see the output. There will be no output on serial monitor.
5. At the coordinator side if we power off, then no data transmission take place and we can observe that Red LED will glow which indicate data transmission not successful.
6. For every successful transmission we can see the yellow LED will glow.
7. When remote node in sleep mode the green LED will not glow when it is awake it will glow.
8. By connecting the individual remote sensor node we can observe the processed and unprocessed output.
9. If we see the processed data and un processed data on coordinator node.
10. If the data is successfully stored in gate way database and in remote server.

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

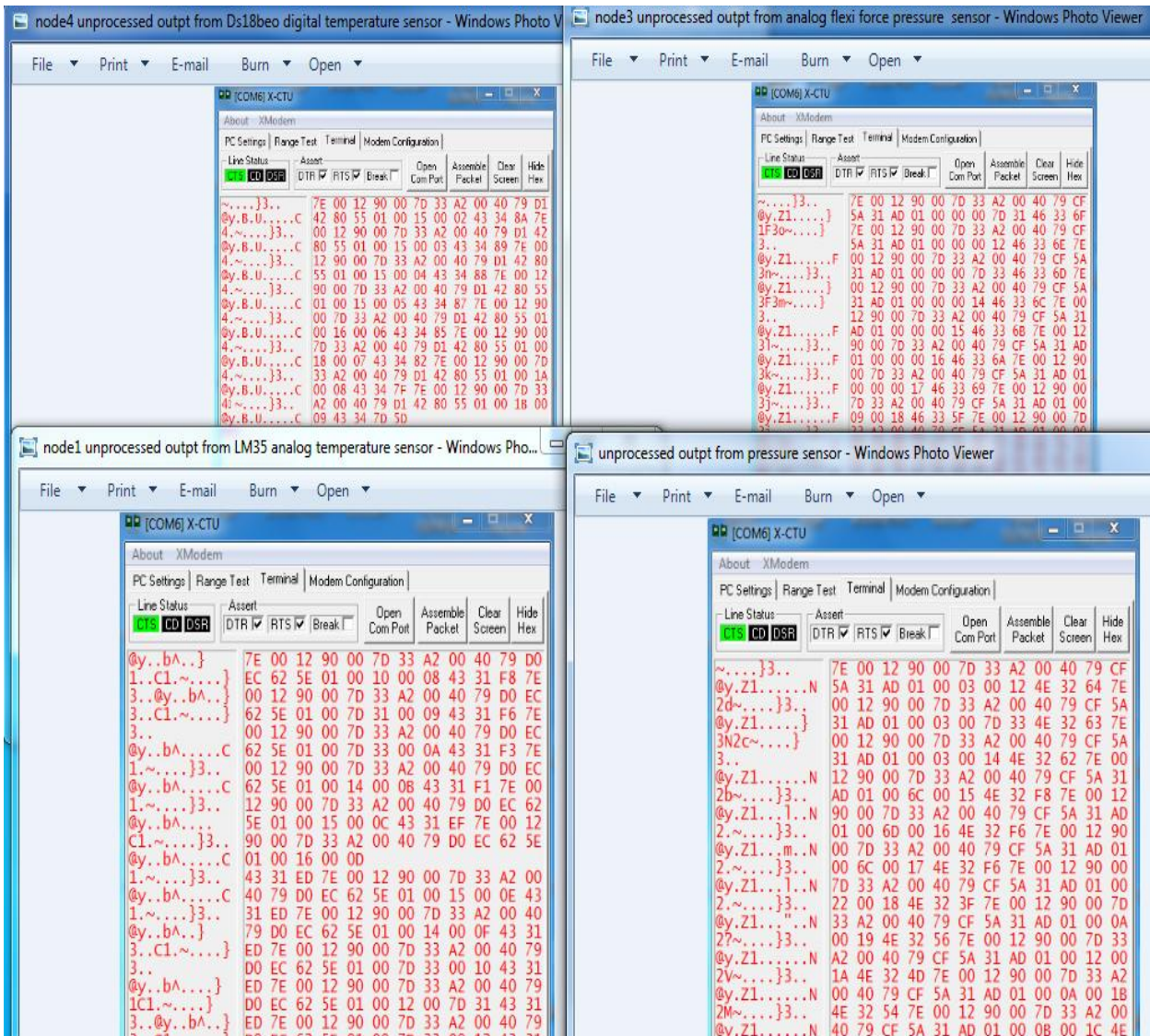


Figure 5.3 Unprocessed data of an individual node received at coordinator node

```

COM9
#Source Address      : 0013A200 4079D142 (8055)
Sequence Number Rx side : 6
Node sequence number  : 9
Node ID              : 4
Data Belongs to      : DS18xxx
Data Values          : 22C
Check Sum            : 82 YES!

#Source Address      : 0013A200 4079D0EC (625E)
Sequence Number Rx side : 7
Node sequence number  : 39
Node ID              : 1
Data Belongs to      : Temperature
Data Values          : 16C
Check Sum            : D9 YES!

#Source Address      : 0013A200 4079D0EC (625E)
Sequence Number Rx side : 8
Node sequence number  : 40
Node ID              : 1
Data Belongs to      : Temperature
Data Values          : 16C
Check Sum            : D8 YES!

#Source Address      : 0013A200 4079D0EC (625E)
Sequence Number Rx side : 9
Node sequence number  : 41
Node ID              : 1
Data Belongs to      : Temperature
Data Values          : 16C
Check Sum            : D7 YES!

#Source Address      : 0013A200 4079D0EC (625E)
Sequence Number Rx side : 10
Node sequence number  : 1
Node ID              : 3
Data Belongs to      : Vibration
Data Values          : 0F
Check Sum            : A YES!

COM9
#Source Address      : 0013A200 4079D0EC (625E)
Sequence Number Rx side : 9
Node sequence number  : 41
Node ID              : 1
Data Belongs to      : Temperature
Data Values          : 16C
Check Sum            : D7 YES!

#Source Address      : 0013A200 4079D0EC (625E)
Sequence Number Rx side : 10
Node sequence number  : 10
Node sequence number  : 1
Node ID              : 3
Data Belongs to      : Vibration
Data Values          : 0F
Check Sum            : A YES!

#Source Address      : 0013A200 4079D142 (8055)
Sequence Number Rx side : 11
Node sequence number  : 2
Node ID              : 2
Data Belongs to      : Pressure
Data Values          : 6N
Check Sum            : 90 YES!

#Source Address      : 0013A200 4079D142 (8055)
Sequence Number Rx side : 12
Node sequence number  : 3
Node ID              : 2
Data Belongs to      : Pressure
Data Values          : 109N
Check Sum            : 28 YES!

#Source Address      : 0013A200 4079D142 (8055)
Sequence Number Rx side : 13
Node sequence number  : 4
Node ID              : 2
Data Belongs to      : Pressure
Data Values          : 14N
Check Sum            : 86 YES!
    
```

Figure 5.4 Processed data of an individual node received at coordinator node

5.2 Testing With Gateway PandaBoard

In order to store the data which is collected by WSN, a database is created in the PandaBoard. In this test, the PandaBoard acts as the gateway. The coordinator node is connected with PandaBoard through USB serial port. The data will be processed, filtered and stored in database after receiving by the gateway PandaBoard. The test procedure is as follows:

1. Select the code file and move it to the target folder.
2. Confirm the IP address of remote server and modify it in the code file.
3. Go to the directory of the code file and run the following commands to compile the code and generate the running file: `gcc -o sniffer_xbeeRe -L/usr/include/mysql sniffer_xbeerRe.c -L/usr/lib/mysql-lmysqlclient -Wall`. The name of running file is `sniffer_xbeeRe.c`
4. Connect the coordinator to the USB port of Panderboard.
5. Determine the port ID of the coordinator by the following commands: `ls /dev/ttyUSB*`. The port ID is `dev/ttyACM0`.

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

6. Run the program and start testing by the following commands: ./sniffer_xbeeRe.c ACM0

The data received in PandaBoard is shown below in Figure 5.5.

```
#Source Address      : 0013A200 4079D142 (8055)
Sequence Number Rx side : 3
Node sequence number  : 26
Node ID              : 2
Data Belongs to      : Pressure
Data Values           : 0N
Check Sum             : 7E YES!

SNO.: 26
SA: 0013A200 4079D142 (8055)
Nid: 2
Type: Pressure
Val: 0N

Succeeded:insert into Xbee values(null, 'A', ' 0013A200 4079D142 (8055)', ' 3', ' 26', ' 2', ' Pressure ', ' 0N
',now());
```

Figure 5.5 Received coordinator data stored in database of gateway

In this figure, the first seven lines display the message which is received. From line eight to twelve, the processed and filtered data is shown. The last line shows the result of the storing data. The records which is stored in MySQL database is shown in Figure 5.6 below.

```
| 944 | A      | 2013-05-21 22:33:48 |
| 0013A200 4079CF5A (31AD) | 2 | 34
| 1 | Temperature | 16C
| 945 | A      | 2013-05-21 22:33:52 |
| 0013A200 4079CF5A (31AD) | 3 | 35
| 1 | Temperature | 17C
| 946 | A      | 2013-05-21 22:33:53 |
| 0013A200 4079D142 (8055) | 1 | 25
| 2 | Pressure    | 0N
| 947 | A      | 2013-05-21 22:34:39 |
| 0013A200 4079CF5A (31AD) | 2 | 40
| 1 | Temperature | 17C
| 948 | A      | 2013-05-21 22:34:46 |
| 0013A200 4079D142 (8055) | 3 | 26
| 2 | Pressure    | 0N
| 949 | A      | 2013-05-21 22:34:50 |
| 0013A200 4079CF5A (31AD) | 4 | 41
| 1 | Temperature | 17C
| 950 | A      | 2013-05-21 22:34:56 |
| 0013A200 4079D142 (8055) | 5 | 27
| 2 | Pressure    | 0N
| 2013-05-21 22:35:00 |
+-----+-----+-----+-----+-----+
950 rows in set (0.01 sec)
```

Figure 5.6 Viewing the inserted data in MySQL database of gate way

From the screen shoots above, we can see that the data from node 2, the pressure sensor, is stored in the record 948. In this testing, everything goes well. The communication between the

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

sensor and coordinator nodes is stable and accurate. The period of processing and storing data is short and almost no delay in the storing period.

5.3 Testing WSN on Leisure Boat

In this scenario, the testing place is on a leisure boat named 'Marex 320' which is shown in Figure 5.7. The boat is moving from Fevik to Arandal and returning. The date is 16th of May. Three sensor nodes, coordinator node and the gateway PandaBoard are all placed on boat. The pictures of these sensors and PandaBoard are shown below in Figure 5.7

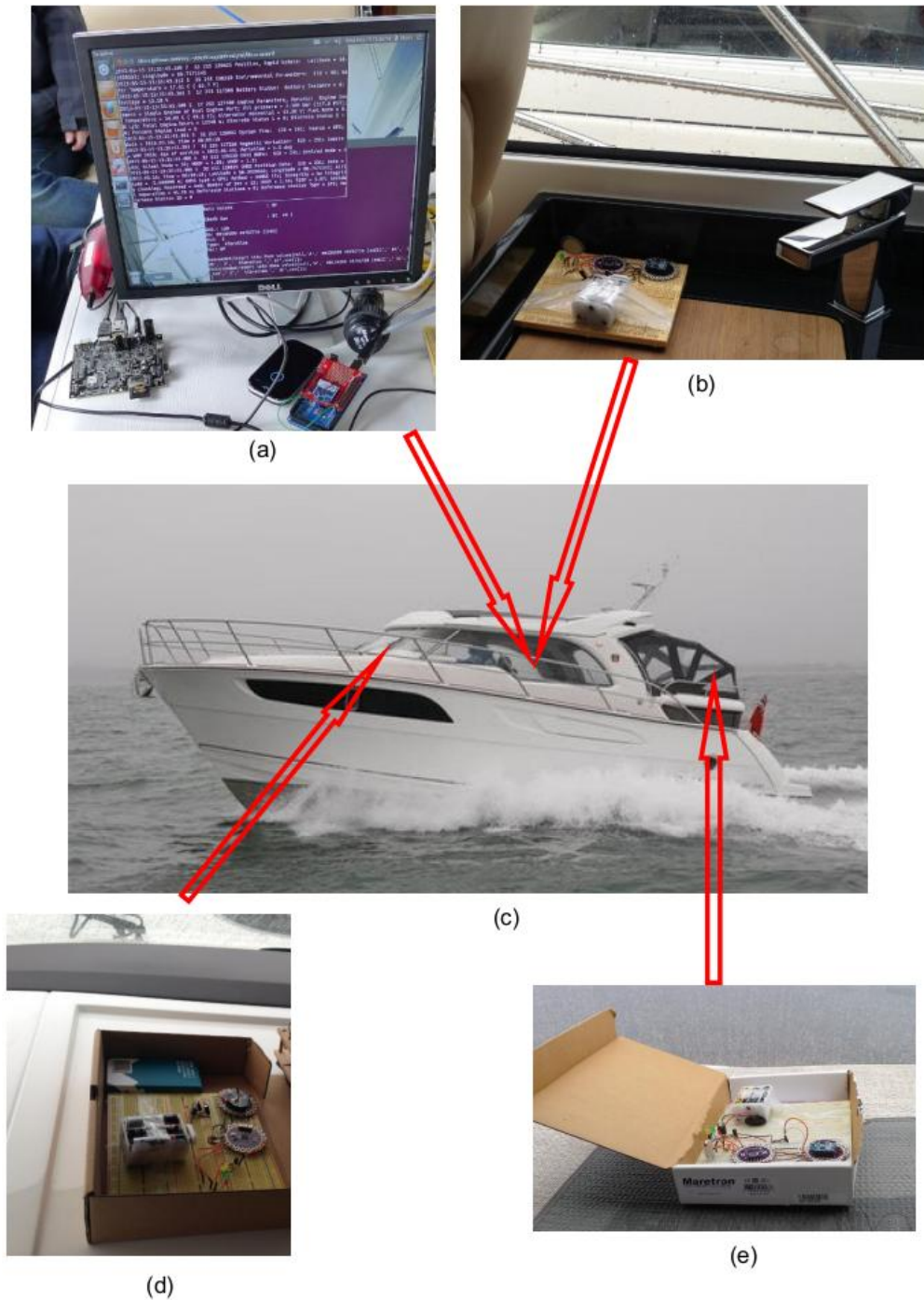


Figure 5.7 WSN in leisure boat

Data Collection and transmission for leisure time boats based on Arduino WSNs and LTE.

Picture (c) shows the outlook of leisure boat 'Marex 320'. Coordinator node, PandaBoard and the 4G router is displayed in picture (a) while picture (b) presents the Flexi force analog sensor. Hardware which shows in these two pictures is placed in the middle part but two sides of the boat. Picture (d) shows the LM35 analog temperature sensor is placed in front of the boat just behind the windshield. The vibration sensor is placed in tail part of the boat which is shown in picture (e).

While the testing is executing out of Grimstad, the remote server is placed in the UIA campus in Grimstad. The gateway PandaBoard is connected through Wi-Fi with the LET router. Then LET router connects to the remote server through Internet.

In the course of this experiment, the speed of leisure boat is changing all the time. The boat is shaking while the speed is high. Emergency brake is executed once when the boat is moving fast. In all these situations, system runs well and records data continuously. The result of test is shown below.

```
bbxm@bbxm-desktop: ~
| 2 | | | Pressure | 0N
| 895 | A | | 2013-05-16 10:56:03 | | 509 | 16
| | | | 0013A200 4079CF5A (31AD) | | |
| | 1 | | | Temperature | 11C
| 896 | A | | 2013-05-16 10:56:03 | | 510 | 621
| | | | 0013A200 4079CF58 (2AE2) | | |
| | 3 | | | Vibration | 0F
| 897 | A | | 2013-05-16 10:56:05 | | 511 | 432
| | | | 0013A200 4079D142 (8055) | | |
| | 2 | | | Pressure | 0N
| 898 | A | | 2013-05-16 10:56:13 | | 1 | 462
| | | | 0013A200 4079D142 (8055) | | |
| | 2 | | | Pressure | 0N
| 899 | A | | 2013-05-16 11:00:45 | | 2 | 43
| | | | 0013A200 4079CF5A (31AD) | | |
| | 1 | | | Temperature | 11C
| 900 | A | | 2013-05-16 11:00:46 | | 3 | 692
| | | | 0013A200 4079CF58 (2AE2) | | |
| | 3 | | | Vibration | 4F
| 901 | A | | 2013-05-16 11:00:50 | | 4 | 694
| | | | 0013A200 4079CF58 (2AE2) | | |
| | 3 | | | Vibration | 7F
| 902 | A | | 2013-05-16 11:00:54 | | 5 | 695
| | | | 0013A200 4079CF58 (2AE2) | | |
| | 3 | | | Vibration | 0F
| 903 | A | | 2013-05-16 11:00:55 | | 6 | 463
| | | | 0013A200 4079D142 (8055) | | |
| | 2 | | | Pressure | 0N
| 904 | A | | 2013-05-16 11:00:58 | | 7 | 44
| | | | 0013A200 4079CF5A (31AD) | | |
| | 1 | | | Temperature | 11C
| 905 | A | | 2013-05-16 11:01:01 | | 8 | 697
| | | | 0013A200 4079CF58 (2AE2) | | |
| | 3 | | | Vibration | 0F
| 906 | A | | 2013-05-16 11:01:02 | | 9 | 464
| | | | 0013A200 4079D142 (8055) | | |
| | 2 | | | Pressure | 0N
| | | | 2013-05-16 11:01:06 | | |
```

Figure 5.8 Data stored in PandaBoard gateway on leisure boat

In these screenshots, Figure 5.8 is recorded in local database while the Figure 5.9 shows the remote side. Since the timestamp shows the insert time and is based on different hardware, it cannot display the delay of remote transmit. During the test, we found that the delay of remote transmit is too short to perceive mostly. However, the Internet connection is not always stable and fine, the delay sometimes lasts for 2 or 3 seconds.

In the local database, the message 897 is recorded while this message is missing in the remote side. The reason is that the Wi-Fi connection is disconnected and the gateway disconnect from the remote server. After that, the program is restarted. Thus this message is just recorded in local database. And then two databases record message normally.

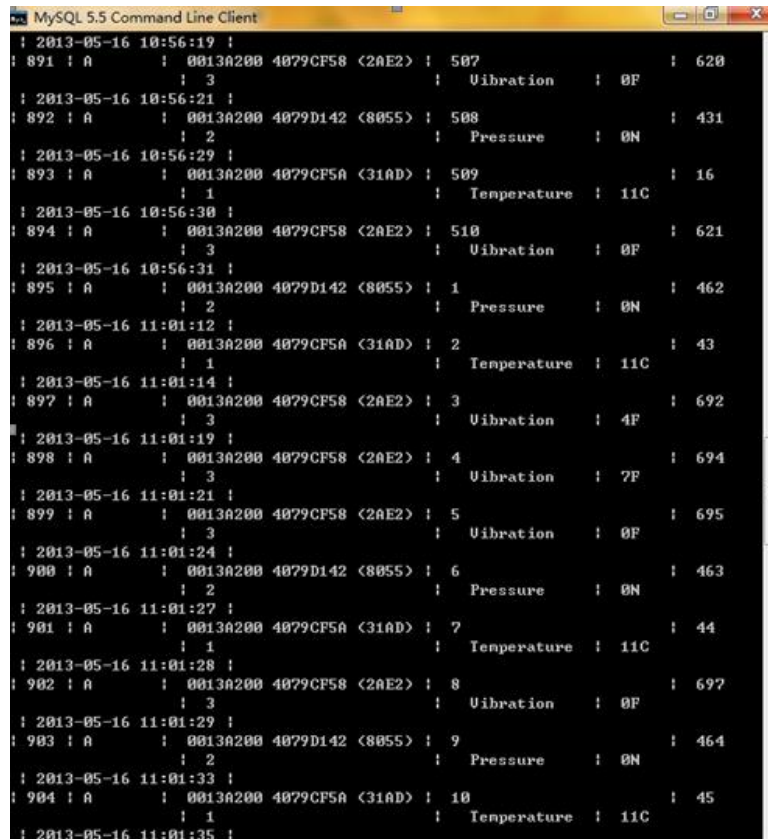


Figure 5.9 Data stored in remote server in UIA campus

During the test, two systems are running at the same time. The CAN-bus system is recording engine parameter and other data while WSN system is storing data from three sensors. The result is shown in Figure 5.10.

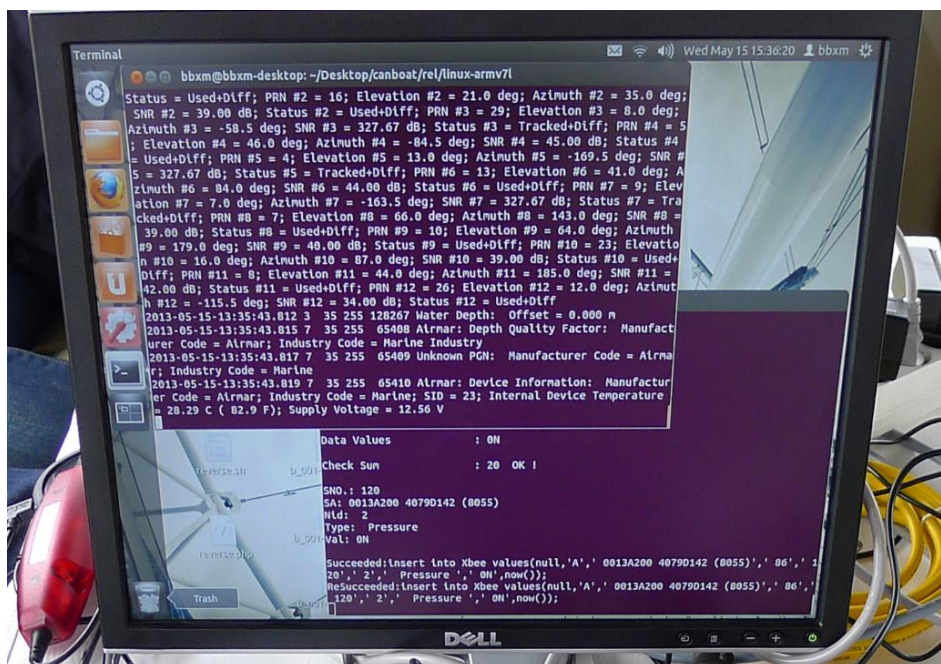


Figure 5.10 Result for two systems

In Figure 5.10, two windows are shown in the screen. The left-upper window is the result for CAN-bus system. The other one shows the result of WSN system. A message which is from the pressure sensor is received and stored in both local and remote databases. The result shows that two systems are running together and integrated successfully.

6 Discussions:

6.1 Bug of First Message

In the test above, a bug of reading data is found. Mostly, data comes in the right format. But in the beginning of the whole process, some message is received in wrong format. Normally, seven different kinds of data are included in one message. But in these abnormal messages, one or more data is missed. Since the coordinator node transmits message continuously and the program is started during the transition. I assumed that incomplete message is received and stored.

To solve this problem, I add identifier '#' at the beginning of the message. If the message is received without the symbol '#', the program will ignore this message and read the next integral message. The changed part of codes is shown below.

```
read(sd, &c, 1);
for(c=#; c != '!' ; j++)
{
    read(sd, &b, 1);
    p[j] = b;;
}
```

Figure 6.1 Modified codes of sniffing data

In this code, the message is read by byte to byte. If the byte '#' is received, the program will read the following part of the message and store it in the variable 'p' until the ending identifier '!' is coming. In this way, we can ensure that the incomplete message is ignored.

However, we found that this bug still existed in the tests, and the message is coming in this way which is shown in Figure 6.2 below.

In this screen shot, 2 or 1.5 messages are coming in the wrong format. There is just one starting symbol while two ending identifiers come out. The first message just includes the result of check sum while the second message lost half of it. I have tested more than 20 times and the results show that all these incorrect messages are randomly received. In some test, two source address values come in one message and nothing else. While the content of the message is random, this kind of incorrect message is just appearing in the beginning.

Since the incorrect message is random, the message length, content of message, and the number of messages are all uncertain, I don't know how to filter this incorrect message in the program. However, I found that after I restart the coordinator and run the program, everything goes well and no incorrect message is received. After that, I tested 10 times in this and no more incorrect message comes.

```
bbxm@bbxm-desktop:~/Downloads$ ./sniffer_xbeeFOR ACM0 127.0.0.1
Connected to database: return:
resultttt:#          : 6 YES!
SNO.:
SA: 6 YES!
Nid: 01
Type:
Val:*****2*****hK**
Succeeded:insert into Xbee values(null, 'A', ' 6 YES!', '', '', '01', '*****2*****hK**',now());
resultttt:
          : 1
Data Belongs to      : Temperature
Data Values          : 17C
Check Sum            : 6 YES!
SNO.: 17C
SA: 1
Nid: 6 YES!
Type:
Val:*****2*****hK**
Succeeded:insert into Xbee values(null, 'A', ' 1', ' Temperature ', ' 17C', ' 6 YES!', '', '*****2*****hK**',
now());
```

Figure 6.2 Incorrect message

I assumed that the reason is that there is some data laying in the buffer on the coordinator. Before running the program, the coordinator is receiving data and sending it to PandaBoard. When we start to read data from the coordinator, some data which is stored in the buffer comes first. After that, the correct message is coming. Since we restart the coordinator, the buffer is refreshed and only correct message will be send.

6.2 Unstable MySQL remote connection

In one test, the program is struck after successfully inserting the value into local database. The reason is that the network is unstable. In my program, the remote database is connected first. Then the program runs a sniffer loop to read the data, scan it and insert it into database. If the network is down while the program runs the sniffer loop, the remote connection is disconnected and will never be connected again. Thus the program is waiting to insert the value into remote database forever.

The solution is that move the remote connection phase into the sniffer loop and disconnected it while the loop is end. The flow chart of this solution is shown below in Figure 6.3.

In this way, the program connects the remote database every time. The system consumption is increased and the period of each sniffer loop is increased greatly. But we can ensure that the remote connection is stable while the data is inserted into the remote server.

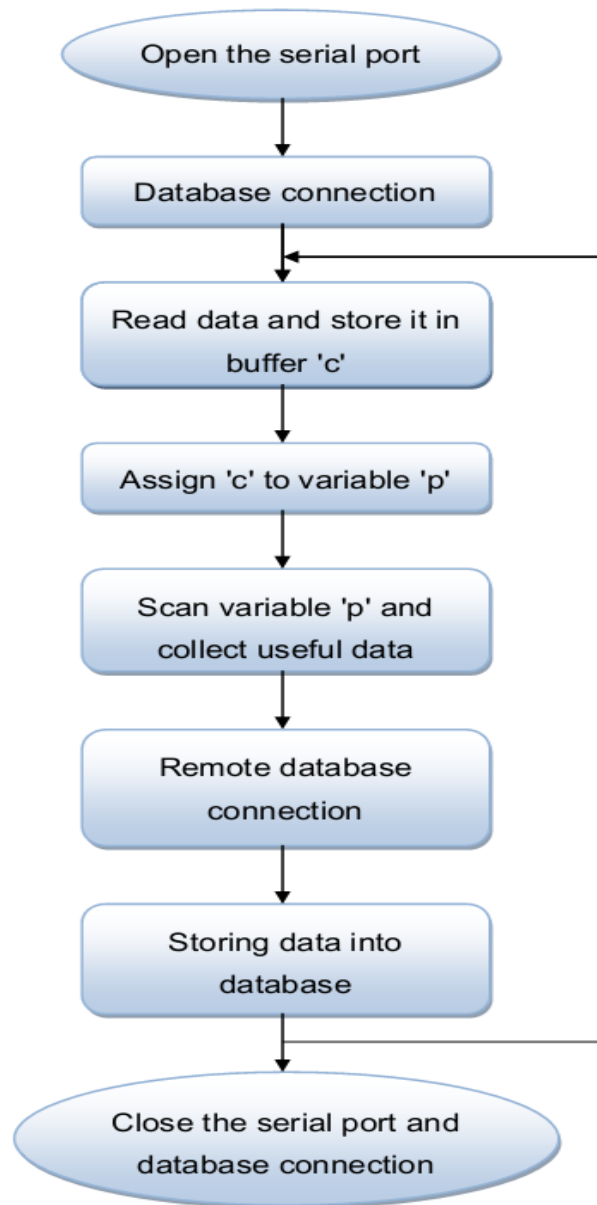


Figure 6.3 Modified flow char of the program

7 Conclusion and Feature work

In this project, we have successfully implemented a ZigBee WSN testbed and connected it to a gateway computer. Four different nodes are used in the network. XBee API mode is used for communication and data transmission payload is optimized thus the quality of communication is increased. We adjust sleep for sensor nodes to reduce power consumption and increase lifetime. We have developed software to collect and filter the data from the ZigBee testbed. Furthermore, we can store the data in local and remote database with the same software. At last, we have tested both in UIA compass and on a real boat. We are satisfied with the achieved results, as the design requirements are successfully achieved.

For future work, we need to improve the program to refresh the buffer in coordinator, so that we don't need to restart it manually. Furthermore more kinds of sensor nodes can be added to the WSN and the number of sensor node could be increased.

At last, the network security needs to be concerned. Although the transmitted data is encrypted by AES algorithm, network protection is still needed for prevent remote hacking.

8 References

- [1] J. Camp, J. Robinson, C. Steger, and E. Knightly. Measurement driven deployment of a two-tier urban mesh access network. In *Proc. ACM MobiSys 2006*, Uppsala, Sweden, June 2006.
- [2] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In *Proc. Mobicom2005*, August 2008
- [3] P. Bahl, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, , and B. Zill. Dair: A framework for managing enterprise wireless networks using desktop infrastructure. In *Proc. the 4th ACM Workshop on HotTopics in Networks (HotNets-IV)*, November 2007
- [4] S. Kim, S. Pakzad, D. Culler, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Proc. Fourth International Conference on Information Processing in Sensor Networks (IPSN'07)*, April 2007.
- [5] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proc. Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2004.
- [6] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proc. OSDI2006*, 2006.
- [7] Nicolas Burri, Pascal von Rickenbach, and Roger Wattenhofer. Dozer: ultralow power data gathering in sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 450{459, New York, NY, USA, 2007. ACM.
- [8] SOSUS, sound surveillance system. <http://www.globalsecurity.org/intell/systems/sosus.htm>. [Online; accessed April 25, 2013].
- [9] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88{97, New York, NY, USA, 2008. ACM.
- [10] Igor Talzi, Andreas Hasler, Stephan Gruber, and Christian Tschudin. PermaSense: investigating permafrost with a WSN in the Swiss Alps. In *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*, pages 8{12, New York, NY, USA, 2007. ACM.
- [11] Gilman Tolle, Joseph Polastre, Robert Szewczyk, David Culler, Neil Turner, Kevin Tu, Stephen Burgess, Todd Dawson, Phil Buonadonna, David Gay, and Wei Hong. A microscope in the redwoods. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 51{63, New York, NY, USA, 2005. ACM.
- [12] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. In *Proceedings of the 14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, April 2006.
- [13] Geo_ Werner-Allen, Konrad Lorincz, Je_ Johnson, Jonathan Lees, and Matt Welsh. Fidelity and yield in a volcano monitoring sensor network. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*, pages 381{396, Berkeley, CA, USA, 2006. USENIX Association.
- [14] Stefan Poslad, *Ubiquitous Computing: Smart Devices, Environments and Interactions*, Wiley, 2009
- [15] Won-Suk Jang, William M. Healy, Miroslaw J. Skibniewski, Wireless sensor networks as a part of a web-based building environmental monitoring system, *Automation in Construction* Volume 17, Issue 6, August 2008, Pages 729-736

- [16] Intel Corporation. Intel®PXA270 Processor. Electrical, Mechanical, and Thermal Specification, April 2008.
- [17] X-CTU Configuration & Test Utility Software technical support <http://www.digi.com/support/eservice/login.jsp> 2007.
- [18] [Puzar05] M.Puzar et al, "SKiMPy: A Simple Key Management Protocol for Ad hoc networks in Emergency and Rescue Operations," 2005
- [19] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 58.
- [20] Wireless Sensor Network Testbed 2.0: A New Service Oriented Architecture by Prof. Dr. L. Thiele: 2008,Pg. 2
- [21] Wireless Sensor Network Testbed 2.0: A New Service Oriented Architecture by Prof. Dr. L. Thiele: 2008,Pg. 3
- [22] Wireless Sensor Network Testbed 2.0: A New Service Oriented Architecture by Prof. Dr. L. Thiele: 2008,Pg. 1
- [23] <https://www.sparkfun.com/products/9266> (accessed on 25th may 2013)
- [24] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 117.
- [25] <http://archlinuxarm.org/platforms/armv7/pandaboard> (accessed on May 26, 2013)
- [26] XBee Wireless Sensor Networks for Temperature Monitoring Vongsagon Boonsawat, Jurarat Ekchamanonta, Kulwadee Bumrungkhet, and Somsak Kittipiyakul School of Information, Computer, and Communication Technology Sirindhorn International Institute of Technology, Thammasat University, Pathum-Thani, Thailand 12000 vongsagon@gmail.com, {popo_jj, just_a_pear}@hotmail.com, somsak@siit.tu.ac.th. Pg 1.
- [27] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 118.
- [28] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 119
- [29] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 120
- [30] <http://hibp.ecse.rpi.edu/%7Econnor/education/breadboard.pdf> (accessed on May 26, 2013)
- [31] <http://www.electronics123.com/s.nl/it.A/id.3204/.f> (accessed on May 26, 2013)
- [32] <https://www.sparkfun.com/products/8937> (accessed on May 26, 2013)
- [33] <https://www.sparkfun.com/products/8687> (accessed on May 26, 2013)
- [34] <https://www.sparkfun.com/products/11061> (accessed on May 26, 2013)
- [35] <https://www.sparkfun.com/products/8687>(accessed on May 26, 2013)
- [36] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 146.
- [37] (<http://www.globalsecurity.org/intell/systems/sosus.htm>) [Accessed April 25, 2013]
- [38] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 25
- [39] Matthias Dyer, Jan Beutel, Thomas Kalt, Patrice Oehen, Lothar Thiele, Kevin Martin, and Philipp Blum. Deployment support network - a toolkit for the development of WSNs. In EWSN '07: Proceedings of the 4th European Workshop on Sensor Networks, pages 195{211. Springer, 2007

- [40] Geo_rey Werner-Allen, Patrick Swieskowski, and Matt Welsh. MoteLab: a wireless sensor network testbed. In IPSN '05: Proceedings of the 4th inter-national symposium on information processing in sensor networks, page 68, Piscataway, NJ, USA, 2005. IEEE Press.
- [41] Vlado Handziski, Andreas K opke, Andreas Willig, and Adam Wolisz. TWIST: a scalable and recon_gurable testbed for wireless indoor experiments with sensor networks. In REALMAN '06: Proceedings of the 2nd international workshop on multi-hop ad hoc networks: from theory to reality, pages 63{70, New York, NY, USA, 2006. ACM.
- [42] Emre Ertin, Anish Arora, Rajiv Ramnath, Vinayak Naik, Sandip Bapat, Vinod Kulathumani, Mukundan Sridharan, Hongwei Zhang, Hui Cao, and Mikhail Nesterenko. Kansei: a testbed for sensing at scale. In IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks, pages 399{406, New York, NY, USA, 2006. ACM.
- [43] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 57.
- [44] X-CTU Configuration & Test Utility SoftwareUser's GuideOnline support: <http://www.digi.com/support/eservice/login.jsp> Phone: (801) 765-9885. 2008 Pg. 2
- [45] X-CTU Configuration & Test Utility SoftwareUser's GuideOnline support: <http://www.digi.com/support/eservice/login.jsp> Phone: (801) 765-9885. 2008 Pg 3
- [46] X-CTU Configuration & Test Utility SoftwareUser's GuideOnline support: <http://www.digi.com/support/eservice/login.jsp> Phone: (801) 765-9885. 2008 Pg 4
- [47] X-CTU Configuration & Test Utility SoftwareUser's GuideOnline support: <http://www.digi.com/support/eservice/login.jsp> Phone: (801) 765-9885. 2008 Pg 7
- [48] X-CTU Configuration & Test Utility SoftwareUser's GuideOnline support: <http://www.digi.com/support/eservice/login.jsp> Phone: (801) 765-9885. 2008 Pg 10.
- [49] MySQL Reference Manual;Copyright c 1997-2000 TcX AB, Detron HB and MySQL Finland AB. 2008. Pg 2.
- [50] MySQL Reference Manual;Copyright c 1997-2000 TcX AB, Detron HB and MySQL Finland AB. 2008. Pg 3
- [51] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 124
- [52] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 125
- [53] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 129
- [54] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 130
- [55] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg. 131
- [56] Building Wireless Sensor Networks by Robert Faludi, Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Dec 2010. Pg.132
- [57] <http://www.omappedia.com/wiki/PandaBoard> and <http://beagleboard.org/hardware-xM> (accessed on May 28, 2013)
- [58] (www.digi.com) (accessed on May 28, 2013)
- [59] http://www.datasheetcatalog.org/datasheets2/98/98580_1.pdf (accessed on May 28, 2013)