WIRELESS SENSOR MESH NETWORKS FOR REAL TIME LIQUID LEVEL MONITORING SYSTEM

By

José Luis Marrero-Ramos

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In

MECHANICAL ENGINEERING

UNIVERSITY OF PUERTO RICO

MAYAGÜEZ CAMPUS

May, 2012

Approved by:

_____	_____
Pedro Quintero, PhD Professor of Mechanical Engineering Member, Graduate Committee	Date
_____	_____
Ricky Valentín, PhD Professor of Mechanical Engineering Member, Graduate Committee	Date
_____	_____
Yi Jia, PhD Professor of Mechanical Engineering President, Graduate Committee	Date
_____	_____
Omar Colón, PhD Interim Director of Mathematical Department Representative of Graduate Studies	Date
_____	_____
Gustavo Gutierrez Director of Mechanical Engineering Chairperson of the Department	Date

ABSTRACT

Sensor networks are an indispensable requirement for a multiple applications, from the home appliances to the most ground breaking technology. They are now used in many industrial and civilian application areas, including industrial process monitoring and control, machine health monitoring, environment and habitat monitoring, healthcare applications, home automation, and traffic control. This thesis presents a wireless sensor mesh network and the application in liquid level sensing. The sensor network consist of a suite of nodes for data sensing, a coordinator node to establish a network and receive data, and a central server to process the data and host it to internet. Two different prototypes were built. In the first prototype central server was based on a personal computer with a seamless XBee RF module configured to operate with ZigBee mesh network standard. The second prototype was based on Arduino open source microcontroller with a Wi-Fi Arduino microcontroller shield and a seamless XBee RF module configured to operate with ZigBee mesh network standard.

Test runs demonstrated the capabilities of both prototypes to measure liquid level of multiple tanks, and host data online through a web host Pachube. Our work demonstrated the feasibility of real time liquid level monitoring for wireless mesh networks.

RESUMEN

Las redes de sensores son un requisito indispensable para múltiples aplicaciones, desde los electrodomésticos a la más revolucionaria tecnología. Estas se usan en muchas áreas de aplicación industrial y civil, incluyendo el monitoreo y control de procesos industriales, monitoreo preventivo de maquinaria, monitoreo ambiental, aplicaciones de atención médica, domótica y control de tráfico. En este documento se presenta una red de sensores inalámbricos y su aplicación en la detección de nivel de agua. La red de sensores consiste en un conjunto de nodos para obtener datos sensoriales, un nodo coordinador para establecer una red y recibir datos, y un servidor central para procesar los datos y presentar en internet. Dos prototipos se construyeron. El primer prototipo central en primer está basado en una computadora personal con un módulo de radio frecuencia llamado XBee, configurado para funcionar en el estándar de protocolo inalámbrico ZigBee. El segundo prototipo se basa en un microcontrolador Arduino de código abierto con un módulo inalámbrico de internet y un módulo de radio frecuencia llamado XBee, configurado para funcionar en el estándar de protocolo inalámbrico ZigBee

Las pruebas realizadas demostraron la capacidad de ambos prototipos para medir el nivel de agua de varios tanques, y presentar los datos de forma online a través de un web de alojamiento llamado Pachube. Nuestro trabajo demostró la factibilidad para monitorear el nivel de agua en tiempo real para redes de malla inalámbrica.

SOLI DEO GLORIA


Ad Experent Omnes

ACKNOWLEDGEMENTS

Thank you God for guiding me throughout the years of my life and for giving me the strength and courage to finish this degree. I would like to thank my beautiful family, my dad José L. Marrero and my mom Milagros Ramos and my brothers Luis Jose Marrero and Jose Ricardo Luis Marrero. I would have never started, continued and finished without your support and love.

I would like to express my deepest gratitude to my advisor Yi Jia ,PhD for his steadfast support and invaluable assistance and knowledge. Thank you for your guidance, friendship, knowledge and support through the years of my bachelor and master degree and for all that support when creating my company. A special thanks to Ricky Valentín, PhD for giving me motivation and confidence during my research and give me the opportunity to participate in multiple research and associations' during my undergraduate and graduate degree. Thanks to Pedro Quintero, PhD, PE for all your support when creating and developing my thesis and my hardware interface.

Finally I would like to thank in the most profound way all my good friends at the UPRM who have made the completion of this thesis possible. Thank you Phillip Rivera, Jose Otaño, Carlos Morales, Armando Cora, Abdiel Aviles, Luis Ortiz and many others for your friendship, knowledge, support, coffee, and countless midnights hours of companionship.

To all of you, I dedicate this work

TABLE OF CONTENTS

LIST OF TABLES

TABLE OF FIGURES

TABLE OF ABBREVIATIONS

1. ADC – Analog to Digital Converter
2. API – Application Programming Interface
3. AT- Attention
4. ATBD – Attention Baud Data
5. ATDL – Attention Destination Low
6. ATID – Attention Personal Area Network (PAN) Identification
7. DIN – Data In
8. DIO – Digital Input/Output
9. DO – Digital Output
10. DOUT – Data Out
11. GND – Ground
12. GUI – Graphic Unit Interface
13. IEE – Institution of Electrical Engineers
14. MAC - Media Access Control Address
15. PAN – Personal Area Network
16. PCB – Printed Circuit Board
17. RF - Radio Frequency
18. RFID - Radio Frequency Identifications
19. RGB - Red Green and Blue (Peripheral)
20. RSSI - Received Signal Strength Indicator
21. RTLLM – Real Time Liquid level Monitoring System

22. RX - Receive

23. TRBS – Transmit Receive Broadcast Slot

24. TX - Transmit

25. UART - Universal Asynchronous Receiver-Transmitter

26. VCC - Voltage of the Common Collector

27. WN – Wireless Node

28. WSN - Wireless Sensor Network

29. LLSN - Liquid level Sensor Network

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Sensor networks are an indispensable requirement for a multiple applications, from the home appliances to the most ground breaking technology. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance. They are now used in many industrial and civilian application areas, including industrial process monitoring and control, machine health monitoring, environment and habitat monitoring, healthcare applications, home automation, and traffic control. The availability of low-cost and low power sensor networks has made place for more flexible system applications. This thesis presents the steps to a wireless sensor network (WSN) and the application in liquid level sensing. The work presented serves as a basis for future implementations in sensor network, monitoring different physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants.

There are multiple solutions for liquid level monitoring, the commonly used instruments maintain contact with the liquid to monitor and are vastly expensive when wireless. From water transportation to fuel transportation there is a need of measuring the liquid level inside them. Today, most storage tanks are made of opaque materials or are not visible to prevent the growth

of bacteria and are kept closed so as not to prevent contamination or insects. This makes more difficult the way to measure the level of the liquid. This approach is suited for mostly all applications of liquid level measurement without coming in contact with the liquid. Our solution has several advantages, low power, centralized server, operating range over 300 feet, compact, easy to deploy and has great energy durability, over the most out off-the-shelf instrumentation. This module solves the same problem in different scenario including outdoor/ indoor. Also, our system takes advantages of its inherent collaboration among devices to monitor multiple environments at the same time without the use and the cost of multiple computers or end devices.

1.2 APPLICATIONS

1.2.1 INDUSTRIAL

Business organizations can benefit from liquid level sensor to automate multiple processes to improve their processes from chemical to water supply. Industrial cooling systems like chillers and cooling towers can be monitored on temperature, humidity, and liquid level in order to gather important information to improve cooling processes.

1.2.2 HOME AUTOMATION

Home appliances can adjust, react and collaborate by being aware of the liquid level in cisterns as well as other appliances and home equipment that uses the water. This can make household owners to be aware the house status.

1.2.3 TRANSPORTATION

Out there exist multiple transportation ways for liquids, from hazardous chemicals to gasoline, crude oil, liquid food etc. Every day there are transportation in trains, water transports, and cars; and everyday there is more safety requirements and monitoring requirements that can be easily provided by sensors. Liquid level sensor detects any change or disparity in liquid level and it can prevent big leaks that can cause great loses or disasters.

1.2.4 RIVER DAMS

There are several uses for river dams, from storing water, to a great energy supply for towns or villages. The liquid level provides a guide in how much energy they can provide or what action to take in order to save water for dry seasons. Knowing liquid levels in different spots of the preceding rivers can predict the course of the energy supply for an entire village.

1.3 MOTIVATION

There is a rapid global development that comes with a computational era. In order to continue the improvement of the previous technology there is a need to combine it to an age where wireless devices work together. Technology has been evolving into a stage where wireless sensor networks are enabling us to solve more challenging problems in more difficult environments. Wireless sensor mesh networks are clusters of nodes that interact with each other to share gathered information, execute commands and most importantly to communicate. In this enormous gamma of applications and purposes of wireless sensor networks, this research focuses on sensor networks that determine water lever from three different tanks.

1.4 MARKET EXPECTATIONS

IDTechEx firm develops various reports on RFID (Radio Frequency Identification) and related technologies. The term Active RFID relates and incorporates many technologies including Real Time Systems, Ubiquitous Sensor Networks and Active RFID with ZigBee (XBee), RuBee, Ultra Wide Band and Wi-Fi. Their latest report on June 2011 [4] analyses the technologies, the market and related issues. IDTechEx has constructed a ten year forecast in which they state that the active RFID market will grow to over 10 times its present size by 2021. They predict that the active RFID market "will rise from 13% of the total RFID market in 2010 to 25% in 2020", which translates to a $6.02 billion market.

1.5 PROPOSED SOLUTION

Build a basic, low cost interface structure for liquid level sensing based on wireless mesh networks. This WSN can be modeled based on any applications influenced on several need in the market. The hardware infrastructure will be built upon the Maxstream XBee RF module and any needed off-the-shelf components. Future decisions must be based on system expectations and current adoption of the hardware products. Our focus is the creation and implementation of a basic algorithm that can have innovative applications and uses for the existing interface structure.

1.6 OBJECTIVE OF THIS RESEARCH

The main objectives of this thesis are:

- Network Architecture Design

- Physical sensing principle of liquid level monitoring
- Hardware analysis- sensor, microcontroller and wireless communication units
- Software implementation and design
- Prototype demonstration system
    - Development and implementation of the liquid level sensor system
    - To validate the software middleware using analog sensors as a functional system
- The development of an algorithm for the interconnection of this WSN nodes that can be used for any related application
- To investigate the reliability of XBee based on wireless mesh networks versus other approaches.

1.7 DESIGN CONSTRAINTS

Starting by defining research constraints in order to define problems and looks for feasible solutions. Our main goal is to create real time liquid level monitoring system based on a wireless mesh network. The requirements for this system are:

- Low-cost
- Low-power
- Mesh networking
- Centralized location server
- Small as possible node hardware
- Use of analog sensor

- Use standardized technologies

Other requirements were drawn upon several iterations of the design and a specific application as the motivation in mind.

1.8 CONTRIBUTIONS

The major contribution can be summarized as follows:

- Built a prototype of real time liquid level monitoring system for 3 water tanks based on wireless mesh networks using off-the-shelf technology
- Implemented software for XBee graphing algorithm for three wireless nodes and one wireless node.
- Implementation of a compact printed circuit board for the system end nodes.
- Implemented a graphing algorithm to display sensor data online
- Conducted experiment with our prototype demonstrating that the system works efficiently

1.9 THESIS OUTLINE

Chapter 1 dealt with the introduction of the research and its purpose on wireless mesh network for liquid level monitoring. The research applications as well as the objectives of the given were defined. The design constraints and distribution were reviewed. In Chapter 2 we explore the technological areas related to liquid level technologies and wireless sensor networks up to date. The presentation and review of the wireless modules and board used were documented. Chapter

3 explains network architecture designs and an overview over wireless nodes and central server as part of the architecture. Chapter 4 takes us to the Network Hardware Design. It is described the sensing element including specifications and sensing principles. The sensor node fabrication is discussed as well as the power consumption of given node. Chapter 5 consists of the Software Design. It introduces us to the wireless module configurations and pairing parameters. A description of the data management system software is presented and the on line hosting process. Chapter 6 presents the results of the experiments conducted on a working implementation. Both prototypes are described and the given results. Finally Chapter 7 presents the conclusions of our work over the thesis research.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter presents a review of related technologies to liquid level sensing and wireless sensor networks technologies. The elements that compose a wireless network are explained to obtain a view of the new systems nowadays. A technological review on our wireless module and its protocol are presented including the wireless module board used in some experiments. A description of the technology used as central server is provided including wireless module and its specifications.

2.2 LIQUID LEVEL SENSING TECHNOLOGIES

The principle in liquid level monitoring is to employ a mechanic device or electric sensor to obtain the measurement in order to gather the water behavior along time; the level measurement can be either continuous or point values. Continuous sensing determines the exact amount of substance while the point-level sensors only indicate whether the substance is above or below the sensing point. During the beginnings there were used different mechanical devices, as the science developed the selection criteria to monitor the level expanded as well as the application constrains. Some of the criteria are temperature, pressure or vacuum, chemistry, dielectric

constant of medium, density (specific gravity), acoustical noise, vibration, mechanical shock, tank or bin size and shape.

Some types of sensors are:

- *Ultrasonic*- a non-contact level sensor that emit an ultrasonic signal and looks for a return signal and uses that time to calculate the level values.

Figure 2-1 Ultrasonic level sensor [12]

- *Mechanic Float*- uses a float, raised or lowered by liquid level, to change the position of a cable or rod. This control the position of a level display clock or a resistor in order to obtain analog or digital readings.

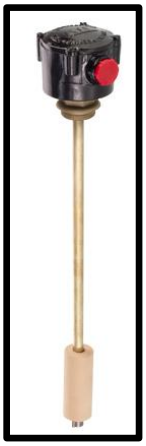


Figure 2-2 Continuous Output Level Transducer [13]

- *Magnetic Float*- the principle behind this is using a permanent magnet sealed inside the float. As the float containing the magnet rises or goes down, the magnet control an internal metal piece that changes the resistance value, giving analog readings for the giving liquid level.

Figure 2-3 Magnetic Float Level Transmitter [14]

- *Capacitance*- uses a capacitance circuit to measure difference in capacitive values as the water rises or goes down.

Figure 2-4 Capacitive Level Sensor

- *Optical Sensor*- uses infrared light to measure distance from sensor to water.

Figure 2-5 Compact optical liquid level switch[15]

- *Pressure sensors*- Pressure sensors are available with a variety of reference pressure options: gauge (psig), absolute (psia), differential (psid), and sealed (psis). All use a force-summing device to convert the pressure to a displacement, but that displacement is then converted to an electrical output by any of several transduction methods. The most common are strain gauges, variable capacitance, and piezoelectric.

2.2.1 PHYSICAL PRINCIPLE OF LIQUID LEVEL SENSING

Since pressure is a very important characteristic of a fluid field there have been developed multiple mechanical elements to measure pressure. Most of these devices are traditional pressure mechanisms that convert the pressure to physical movement. To analyze, manipulate and understand this data this physical movement is transduced to electrical or other input.

In order to obtain the liquid level of any tank we would be obtaining the gauge pressure. This pressure is measured relative to an absolute reference pressure, which would be defined in a

manner convenient to the measurement. The relation between an absolute pressure and gage pressure will be:

$$P_{gauge} = P_{abs} - P_0 \qquad (1)$$

Po is a reference pressure. In this project we will be using the local atmospheric pressure as a reference pressure at the tanks.

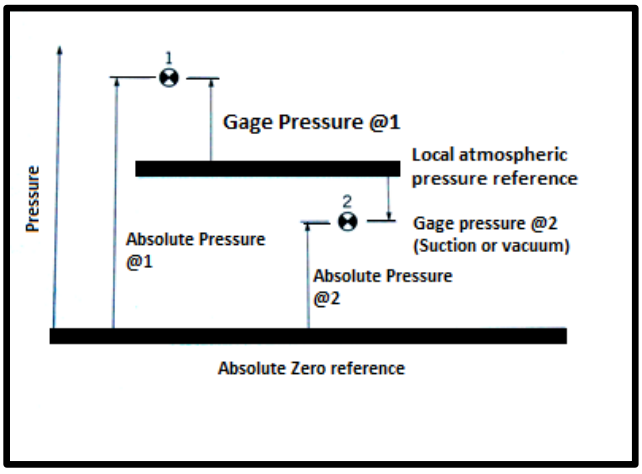


Figure 2-6 Graphical Representation of Gage and Absolute Pressure

2.2.2 FLUID COLUMN

The force that generates pressure can be caused by any weight or mass. Therefore, the weight of the fluids' mass generates pressure. A column of fluid generates pressure proportional to the density of the fluid and the vertical height of the column. The pressure at a given depth is independent of the area of the column and the shape of the container.

The weight (W) of the fluid in the container is distributed over the area (A) of the base. At the bottom of the container:

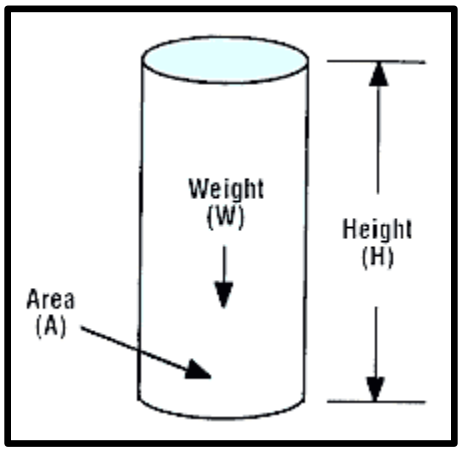


Figure 2-7 Liquid Container

```
P = W/A;
because W = mg and P = mg/A
Mass (m) = density (r) × volume (V),
so P = rVg/A
V = Ah,
so P = rAhg/A, and
P = rgh
```

Thus, P is independent of A and dependent, r (density), h (height of column), and g (acceleration of gravity).

2.2.3 CLOSED CONTAINER

Closed liquid containers are often blanketed by pressurized gas. Blanketing using inert gas is used for the storage of materials in case there could react with components of the air, oxygen or humidity.

Nitrogen is usually used as an inert gas. An inner container atmosphere not only provides quality assurance but it also reduces the danger of explosions. For containers blanketed with inert gas, pressure monitoring therefore is of major importance.

Differential pressure is used here to make sure the superimposed gas pressure is not reflected in the measurement.

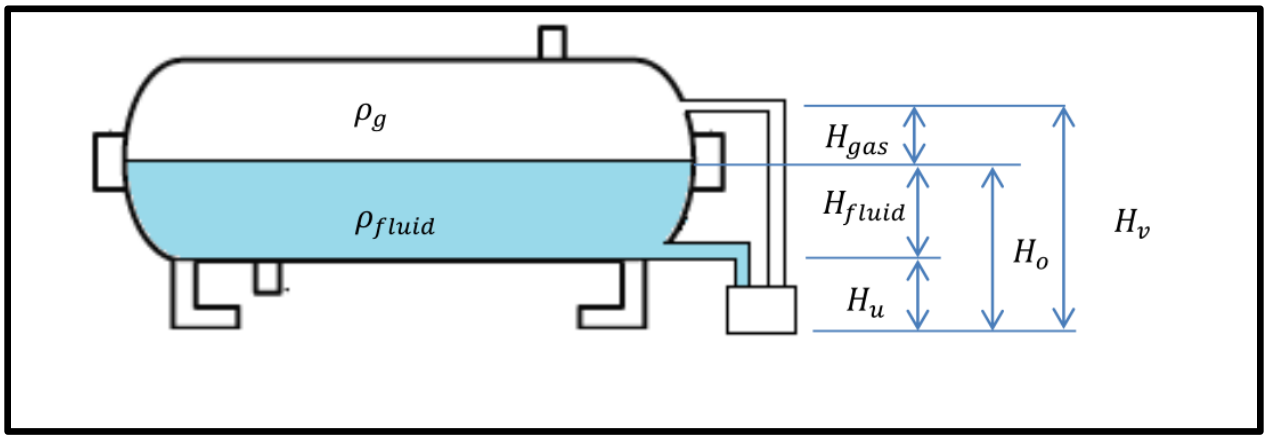


Figure 2-8 Closed Container

Given:

$H_{fluid} + H_{gas}$= meter

H_v= meter

H_u= meter

ρ_{fluid}=kg/m^3 (density of stored liquid)

ρ_g=kg/m^3 (density of gas (N) above the stored liquid)

ρ_v=$\rho_g = kg/m^3$ (density compensation side = gas)

G=9.81 m/s^2 (gravitational acceleration)

P_u(at lower level Hu)

P_o(at lower level Ho)

Calculation:

$$\Delta P_o = \left[H_o * \rho_{fluid} * g + (H_v - H_o) * \rho_g * g\right] - H_v * \rho_v * g$$

2.3 WIRELESS SENSOR NETWORK TECHNOLOGIES

Wireless sensor networks systems are composed of individual embedded systems that are capable of interact with their environment through various sensors, processing information, and communicating this information wirelessly. A sensor node typically consists of three components and can be either an individual board or embedded into a single system:

- **Wireless module** – possess the communication capabilities and the programmable memory where the application code resides. A mote usually consists of a microcontroller, transceiver, power source, memory unit, and may contain a few sensors. A wide variety of platforms have been developed in recent years including Mica2 [2], Cricket [2], Iris[2], Telos [2], SunSPOT [2], lmote2 [2], Wi232 [2], LT2510 FlexRF® [2], XBee [8], ZigBee[5].

Figure 2-9 Laird Technologies LT2510 FlexRF® 2.4 GHz Wireless Module

Figure 2-10 ZigBee / 802.15.4 Modules WIRELESS MCU IEEE802.15.4 TRNSCVR

- **Sensor and Sensor Board** – is the base of the in-place operation. Contains the wireless module and give place for multiple types of sensors.
- **Programming board**- sometimes embedded on the wireless module, they provides multiple interfaces for connecting different modules to a network or locally to a PC/laptop. They could also be programmed over the radio.

Table 1 Wireless Modules [2]

Mote type	CPU speed (MHz)	Prog. mem. (kB)	RAM (kB)	Radio freq. (MHz)	Tx. rate (kbps)
WeC	8	8	0.5	916	10
rene	8	8	0.5	916	10
rene2	8	16	1	916	10
dot	8	16	1	916	10
mica	6	128	4	868	10/40
mica2	16	128	4	433/868/916	38.4 kbaud
micaz	16	128	4	2.4 GHz	250
Cricket	16	128	4	433	38.4 kbaud
EyesIFX	8	60	2	868	115
TelosB/Tmote	16	48	10	2.4 GHz	250
SHIMMER	8	48	10	BT/2.4 GHz[a]	250
Sun SPOT	16–60	2 MB	256	2.4 GHz	250
BTnode	8	128	64	BT/433–915[a]	Varies
IRIS	16	128	8	2.4 GHz	250
V-Link	N/A	N/A	N/A	2.4 GHz	250
TEHU-1121	N/A	N/A	N/A	0.9/2.4 GHz	N/A
NI WSN-3202	N/A	N/A	N/A	2.4 GHz	250
Imote	12	512	64	2.4 GHz (BT)	100
Imote2	13–416	32 MB	256	2.4 GHz	250
Stargate	400	32 MB	64 MB SD	2.4 GHz	Varies[b]
Netbridge NB-100	266	8 MB	32 MB	Varies[b]	Varies[b]

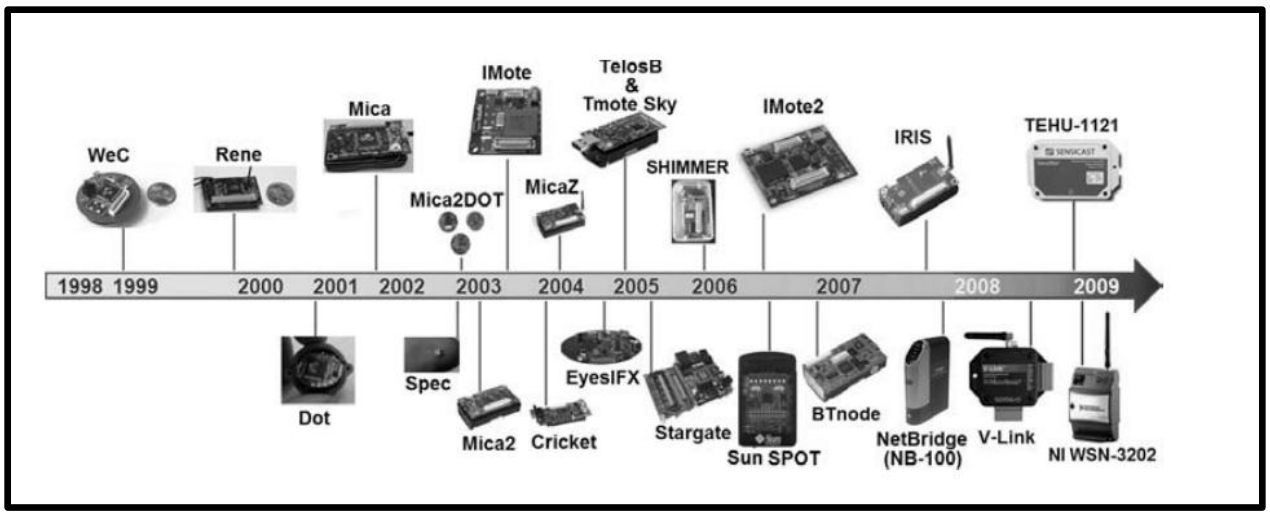


Figure 2-11 Timeline for wireless nodes platform [2]

Table 1 Wireless Modules [2] captures the major characteristics of popular platforms that were designed over the past few years in terms of their processor speed, programmable and storage memory size, operating frequency, and transmission rate. The timeline for these platforms is also shown in Figure 2-11.

2.4 TECHNOLOGICAL BACKGROUND

2.4.1 XBEE MODULE OVERVIEW

One of the communication protocols proposed for this project is the XBee. XBee is a wireless communication module that Digi built to the IEE 802.15.4 standard [17]. The XBee wireless module is a stand-alone, ready-to-use commercial device designed for low-power, low-cost, wireless sensor networks. This module comes from factory to be a wireless serial line replacement. The XBee can be programmed to do other functions like behave as a "wireless wire" where a level transition on an input pin of one module is sent out as the same level

transition on a different module output pin. There is also a special API[1] mode, where the modules accept bytes of data from the host to transmit into the network. These modules are controlled via a serial interface and a predefined API command set. The API is used to perform configurations on the module, to control the hardware on the XBee and to perform several networking tasks.

They operate within the ZigBee[2] mesh networking protocol at the network level or MAC level. The network could grow up to 65,000 wireless nodes unique addresses. This is great for developing wireless control networks and real time sensor networks that need to interact between them. The network also supports point-to-point, point-to-multipoint and peer-to-peer topologies. The modules have an indoor range of up to three hundred feet and outdoors of up to one mile.

2.5 XBEE-PRO 802.15.4 SPECIFICATIONS

- Power output:
- 63 mW (+18 dBm) North American version
- 10 mW (+10 dBm) International version

---

[1] An **application programming interface** (**API**) is a particular set of rules and specifications that software programs can follow to communicate with each other. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers.

[2] The **ZigBee protocol** is built on recent algorithmic research (Ad-hoc On-demand Distance Vector, neuRFon) to automatically construct a low-speed ad-hoc network of nodes. In most large network instances, the network will be a cluster of clusters. It can also form a mesh or a single cluster.

- Indoor/Urban range: Up to 300 ft. (90 m)
- Outdoor/RF line-of-sight range: Up to 1 mile (1.6 km) RF LOS
- RF data rate: 250 Kbps
- Interface data rate: Up to 115.2 Kbps
- Operating frequency: 2.4 GHz
- Receiver sensitivity: -100 dBm (all variants)

For the prototype system presented it is used the XBee-Pro with chip antenna.

2.6 ANTENNA

These radio modules we are using are with two different antennas, in order to take advantage over the different XBee antenna options. The XBee modules currently used in the project are:

2.6.1 WHIP OR WIRE ANTENNA

A single piece of wire connected coaxially from the body of the radio. Offers Omni-dimensional radiation, meaning the maximum transmission distance is the same in all directions when is straight and perpendicular to the module.

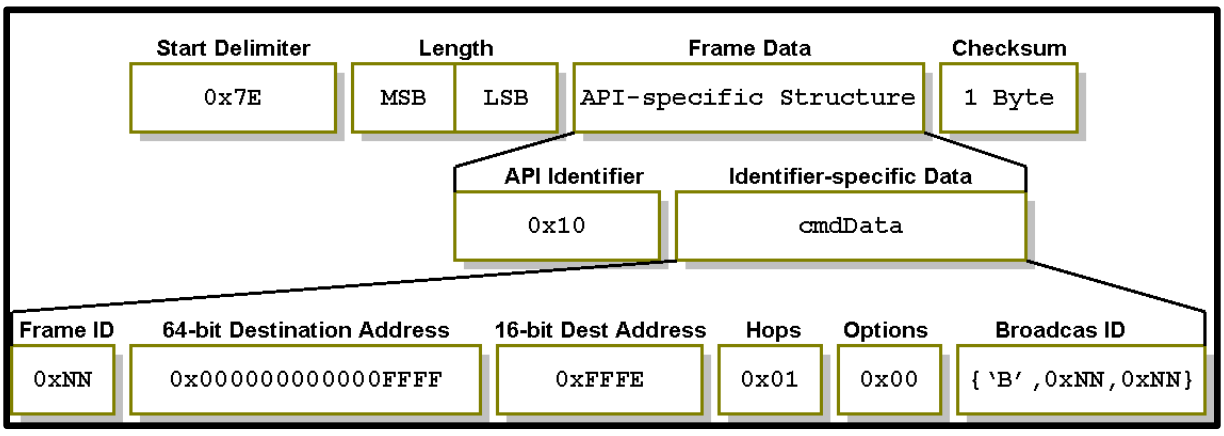
2.6.2 CHIP ANTENNA

This antenna is a flat ceramic chip that is embedded on the body of the XBee. This makes the XBee smaller and sturdier. Chip antennas have a cardioid (heart-shaped) radiation pattern, meaning that the signal is attenuated in many directions.

2.7 API MODE

XBee modules have two modes of operation, command mode and API mode. In this research project we will deal with the XBee on API mode. This mode specifies the application level protocol in which modules and devices talk to each other, be it wirelessly or via serial port. Figure 2-12 shows the structure of the API data frames[3]. Any module can communicate with another module in the network by defining the destination address and other predefined commands. The most common commands are:

- *ATBD*- It is the XBee AT command to change the baud rate. Number of pulses per second, also known as modulation rate.
- *ATID*- It is the XBee AT command to change the Personal Area ID (PANID).
- *ATMY*- It is the XBee AT command to change the address number.
- *ATDL*- It is the XBee AT command to change the destination low address.



[3] Description of the **API data frames** can be found on XBee/XBee Pro product manual, see bibliography [8].

Figure 2-12 General XBee API Frame Structure

2.7.1 XBEE PIN DESCRIPTION

The next table presents the pin assignment for the XBee and XBee-Pro modules. (Low asserted signals are distinguished with a horizontal line above signal name.) Throughout the document we have been giving references about the pin numbers and the use we have been giving to them.

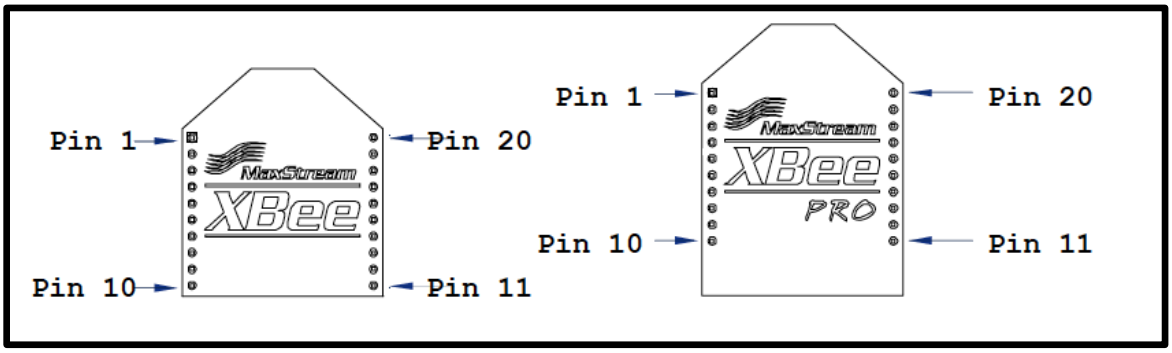


Figure 2-13 XBee and XBee-Pro Modules Pin Localization Diagram

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / **CONFIG**	Input	UART Data In
4	DO8*	Output	Digital Output 8
5	**RESET**	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	$\overline{\text{DTR}}$ / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	**CTS** / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / **SLEEP**	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	**RTS** / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

Table 2 XBee & XBee-Pro Pin ID Table

Design Notes:

• Minimum connections: VCC, GND, DOUT & DIN

• Minimum connections for updating firmware: VCC, GND, DIN, DOUT, RTS & DTR

• Signal Direction is specified with respect to the module

• Module includes a 50k Ω pull-up resistor attached to RESET

• Several of the input pull-ups can be configured using the PR command

• Unused pins should be left disconnected

2.8 MAXSTREAM DIGI XBEE BOARD

The first experiment realized with the XBee was using the MaxStream board. We can see further in the thesis how did we developed a new smaller printed circuit board for the proposed end node, also has the capacity to embed a XBee radio transceiver. As part of the development kit from digi.com (part no. XK-Z11-PD) comes with two different types of boards. We can differentiate them by the peripherals used, Figure 2-14 shows the MaxStream RGB board and MaxStream USB boards, shown in Figure 2-15.

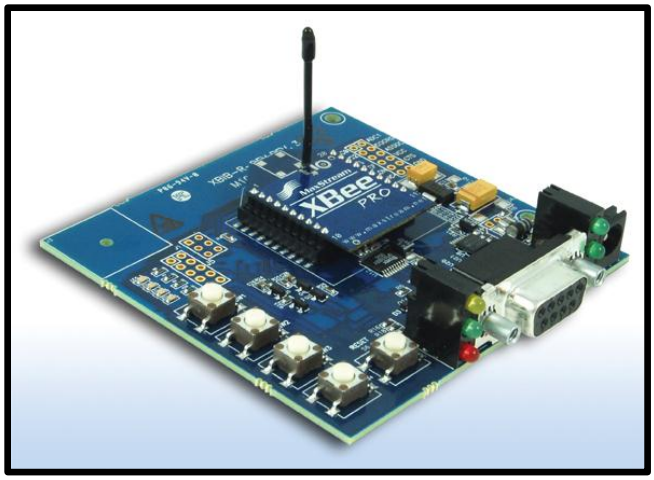


Figure 2-14 MaxStream RGB board

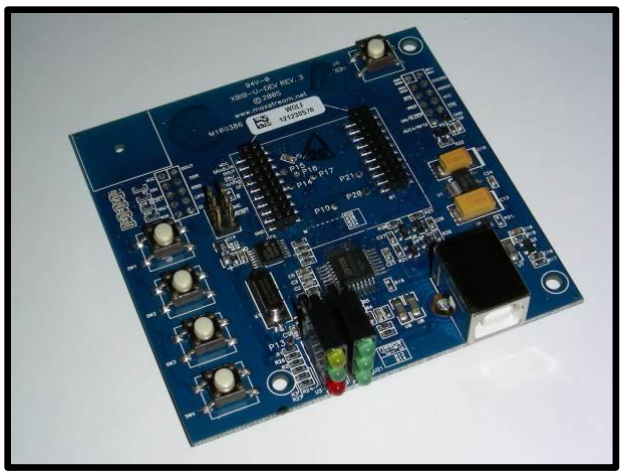


Figure 2-15 MaxStream USB 2.0 board

They are two good boards but are substantially larger. This limits application capabilities around small environments.

2.9 ARDUINO MICROCONTROLLER

Arduino is an open source microcontroller system that is very popular for creating prototypes. The system is flexible and can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights motors and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on wiring) and the Arduino development environment (based on processing). Arduino projects can be stand-alone or they can communicate with software running on a computer.

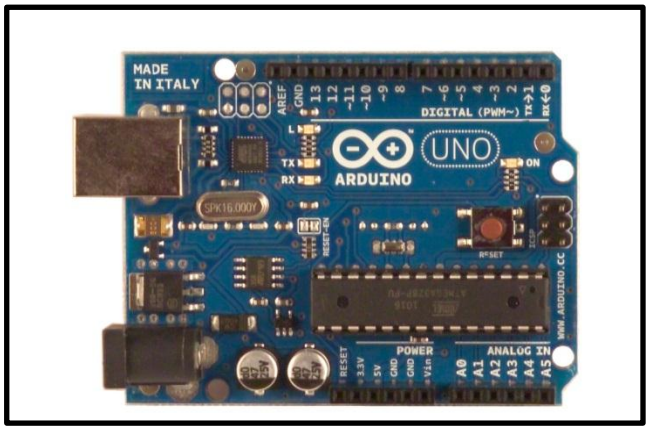


Figure 2-16 Arduino Microcontroller

2.10 WiShield

The WiShield is an add-on board that brings Wi-Fi and true wireless connectivity to the Arduino platform. It is best suited for low data rate applications, such as simple control and sensor applications like liquid level measurement.

Figure 2-17 Arduino Wi-Fi Shield

2.11 WI-FI MODULE FEATURES

- 802.11b Wi-Fi certified
    - 1Mbps and 2 Mbps throughput speeds
- Supports both infrastructure (BSS) and ad hoc (IBSS) wireless networks
- Ability to create secured and unsecured networks
    - WEP (64-bit and 128-bit)
    - WPA/WPA2 (TKIP and AES) PSK
- Low power usage
    - Standby mode: 10mA
    - Transmit: 230mA
    - Receive: 85mA
    - Sleep: 250μA

CHAPTER 3

NETWORK ARCHITECTURE DESIGN

3.1 ARCHITECTURE

A wireless sensor network consists of sensor nodes, interface circuit, power supply, and RF radio module. Compared with many short-range wireless communication protocols such as Bluetooth and Wi-Fi, ZigBee standard is more suitable for liquid level monitoring system and applications in a Multi-Hop Cluster Network. We monitor water with two different network server systems, computer based and Arduino based. Both wireless sensor networks are designed to have two different wireless network architectures as shown in Figure 3-1 and Figure 3-2. Multi-Hop Cluster Network architecture has the advantage to analyze a cluster of information, like a container array, and Multi-Hop Multi-Point to Point Network is designed to reach larger areas, like larger water containers. Each network is composed of a set of sensors nodes to obtain the data, a router node to relay the sensed data and a coordinator node to start the network, control the information, receive data and process all data.

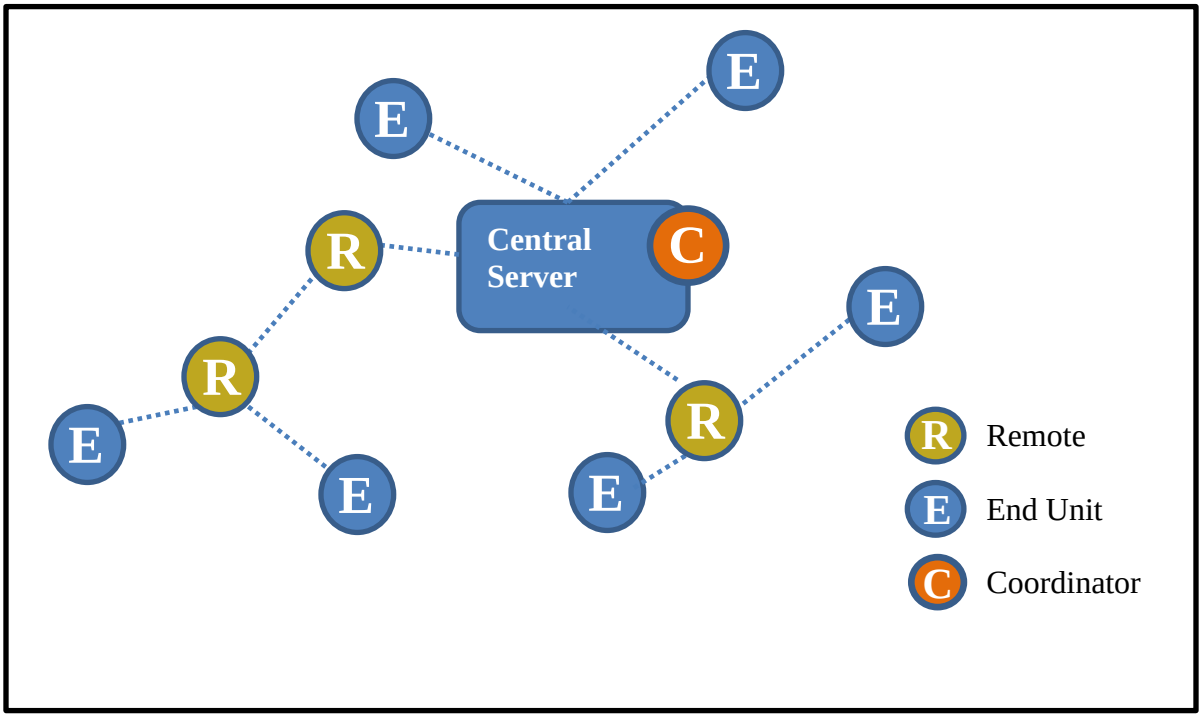


Figure 3-1 Multi-Hop Multi-Point to Point Network

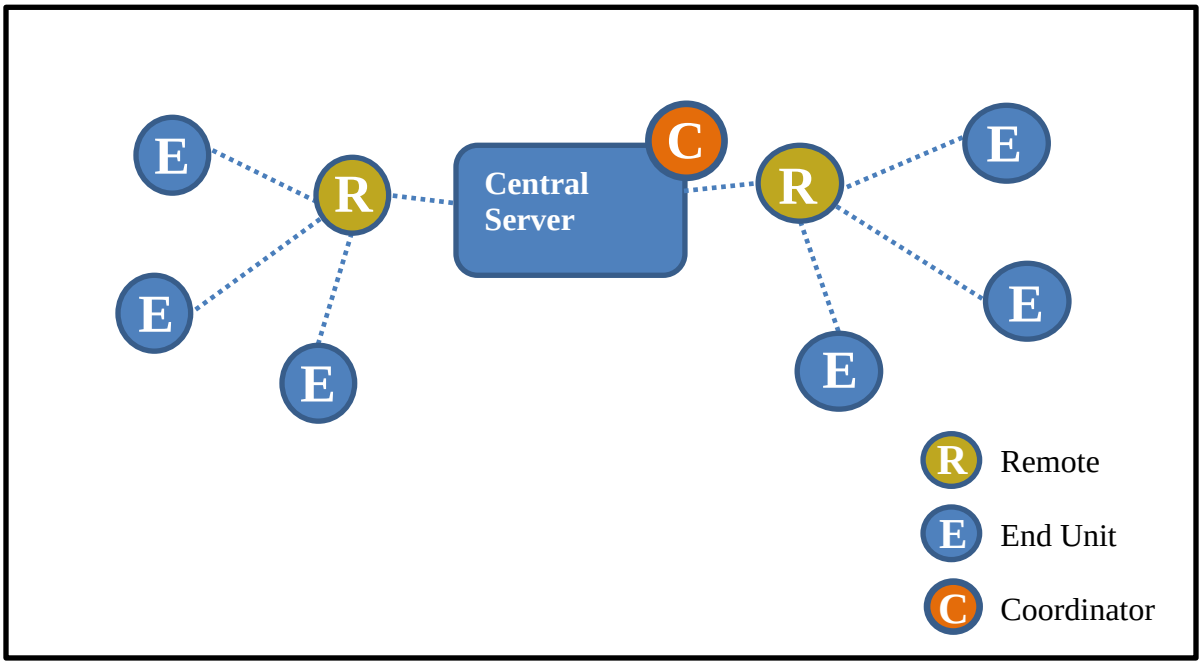


Figure 3-2 Multi-Hop Cluster Network

Software has been developed in order to provide an optimal management of the sensor data, collect distributed data, delivering and analyzing the liquid level for each tank. Since there are two different network systems, the computer based system and the Arduino system; we approach the problem from two different algorithms. Each algorithm contains a set of commands, filters and protocols to deal with the given hardware.

- *Wireless Nodes* – The nodes have several tasks to do besides creating and maintaining the wireless mesh network. Each node receives and performs commands sent by the central server (computer or microcontroller) and return analog values to the central server.
- *Central Server* – The central server software is in charge of managing the coordinator node, configuring the wireless nodes in the network, and calculating digital values in order to graph it. Digital information is used to draw the liquid level in the graphic unit interface.

CHAPTER 4

NETWORK HARDWARE DESIGN

4.1 INTRODUCTION

Our Wireless Network system is a set of interacting and interdependent components forming the core of our liquid level monitoring. WSN nodes usually are composed of a series of basic sensing elements, sensor interface circuits, power supply and radio communication module. We would like to create a hardware that can be able to monitor a physical or environmental condition, to be compact, easy to deploy and has a great energy durability. Our sensing network has the analog pressure transducer component to interact with water physical behavior; interacting according XBee wireless radio, and to finally meeting the Base Station. These stations collect given data to further analysis and display. Base stations are fundamentally in charge of the design calculations, iterations, and manage specific applications to obtain our final product, which in this case is the real time liquid level of multiple tanks.

4.2 SENSING UNIT

The MPX2010 series silicon piezoresistive pressure sensors provide a very accurate and linear voltage output directly proportional to the applied pressure [27]. These sensors house a single

monolithic silicon die with the strain gauge and thin film resistor network integrated. The sensor is laser trimmed for precise span, offset calibration and temperature compensation.

For both our prototype systems our sensing units maintain the same hardware. Compatibility is an advantage that common systems don't have. The end node consists of XBee modules and MPX2010DP pressure transducers [27], as shown in Figure 4-1 . As a sensing unit we employ the MPX2010DP pressure transducer. The MPX2010DP is rated to 10kPa in differential pressure; this is equivalent to just over a meter of water dept. A good sensor with a linear variation for pressure to voltage, but the voltage generated is too low in order to interact correctly with the XBee radio transceiver. When in use, this pressure transducer voltage variation is very small, for a pressure differential of 10kPa it will generate a voltage of only 23 mVdc.

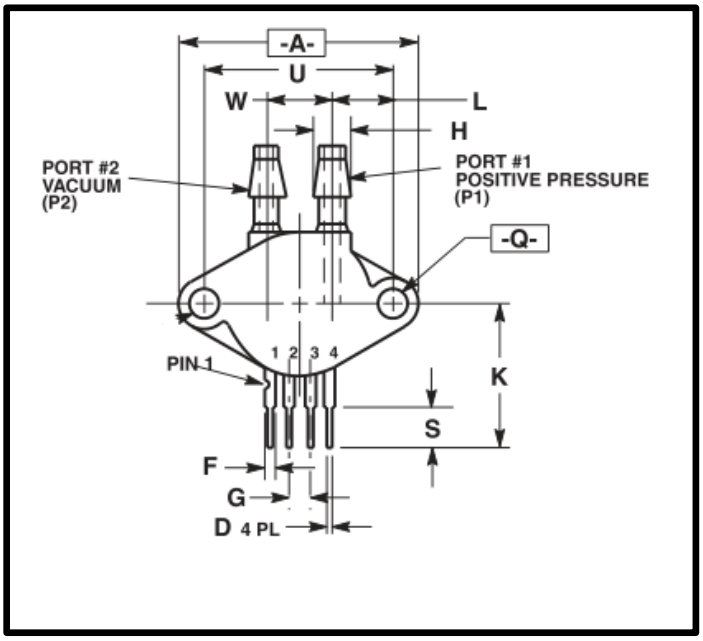


Figure 4-1 MPX2010DP Pressure Transducer Schematic

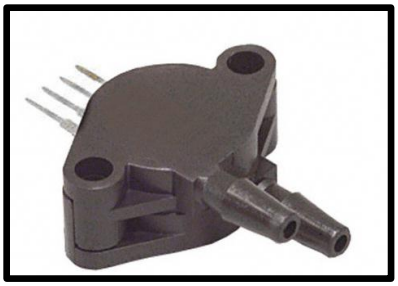


Figure 4-2 MPX2010DP Pressure Transducers

4.2.1 PRESSURE SENSING PRINCIPLE

A semiconductor piezo-resistance dispersion pressure sensor has a semiconductor distortion gauge formed on the surface of the diaphragm, and it converts changes in electrical resistance into an electrical signal by means of the piezo-resistance effect that occurs when the diaphragm is distorted due to an external force (pressure).

A static capacitance pressure sensor has a capacitor that is formed by a static glass electrode and an opposing movable silicon electrode, and it converts changes in static capacitance that occur when the movable electrode is distorted due to an external force (pressure) into an electrical signal [18].

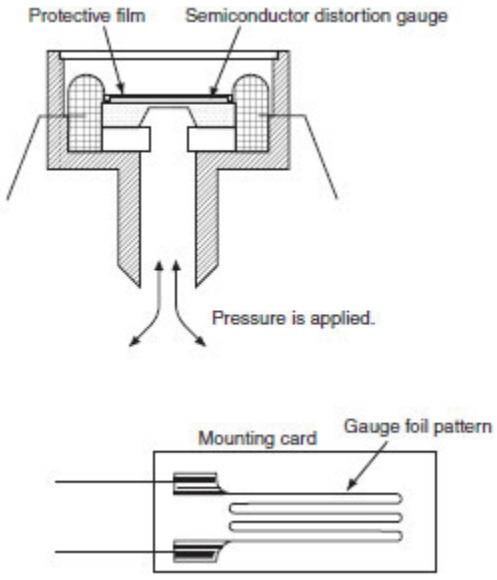


Figure 4-3 Semiconductor Distortion Gage Construction

The distortion gauge operates using the following principle. The kinetic potential of the electrons in a semiconductor changes when the crystal structure of the semiconductor is distorted. This changes the carrier mobility in the semiconductor, resulting in a change in the electrical resistance. In a pressure sensor, external pressure distorts a diaphragm, causing distortion in the resistor section in the gauge. A voltage thus is output that is proportional to the pressure.

Piezo-resistance Effect

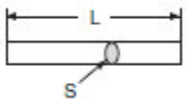


Figure 4-4 Conductor original condition

The mechanical force changes the electrical resistance, we can show by:

$R = \rho * L/S$

When this conductor is stretched the length increases and the cross sectional area S decreases.

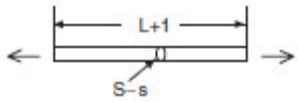


Figure 4-5 Stretched Conductor

After stretched electrical resistance of the above conductor is expressed as:

$R' = \rho * (L + 1)/(S - s)$, $R' > R$

4.2.2 PERFORMANCE

Figure 4-6 Output vs. Pressure Differential for Pressure Transceiver shows the output characteristics of the MPX2O1O series at 25°C. The output is directly proportional to the differential pressure and is essentially a straight line. The effects of temperature on full scale span and offset are very small and are shown under Operating Characteristics. This performance over temperature is achieved by having both the shear stress strain gauge and the thin-film resistor circuitry on the same silicon diaphragm. Each chip is dynamically laser trimmed for precise span and offset calibration and temperature compensation.

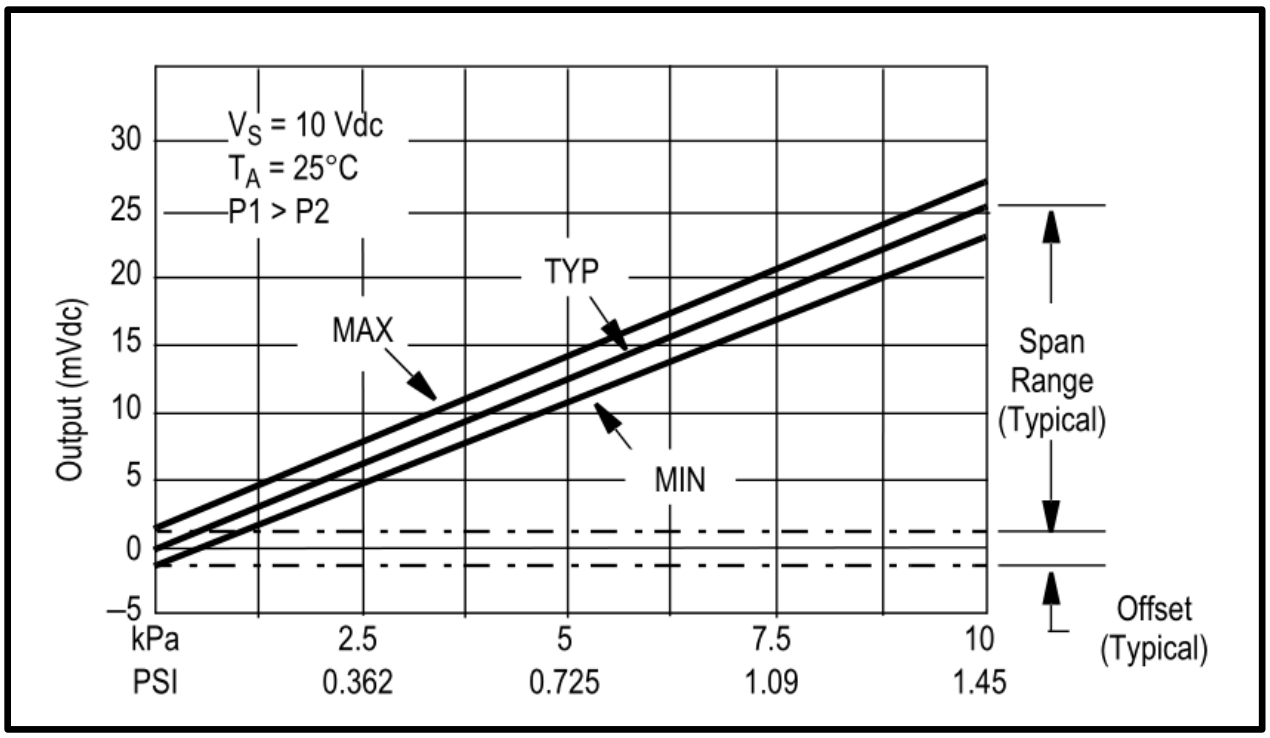


Figure 4-6 Output vs. Pressure Differential for Pressure Transceiver

4.3 SENSOR UNIT INTERFACE - END DEVICE

The network end device is composed of multiple electronic elements, such the sensing unit to the XBee RF module, which gives our system the necessary properties to interact with the working body. For both our prototype systems our end node maintains the same hardware. As we discussed in last section the MPX2010DP the pressure transducer voltage variation is very small, for a 10kPa it will generate a voltage of only 23 mV dc. as we can see on **Error! Reference ource not found.**. In order to deal with this limitation an amplifier circuit was implemented.

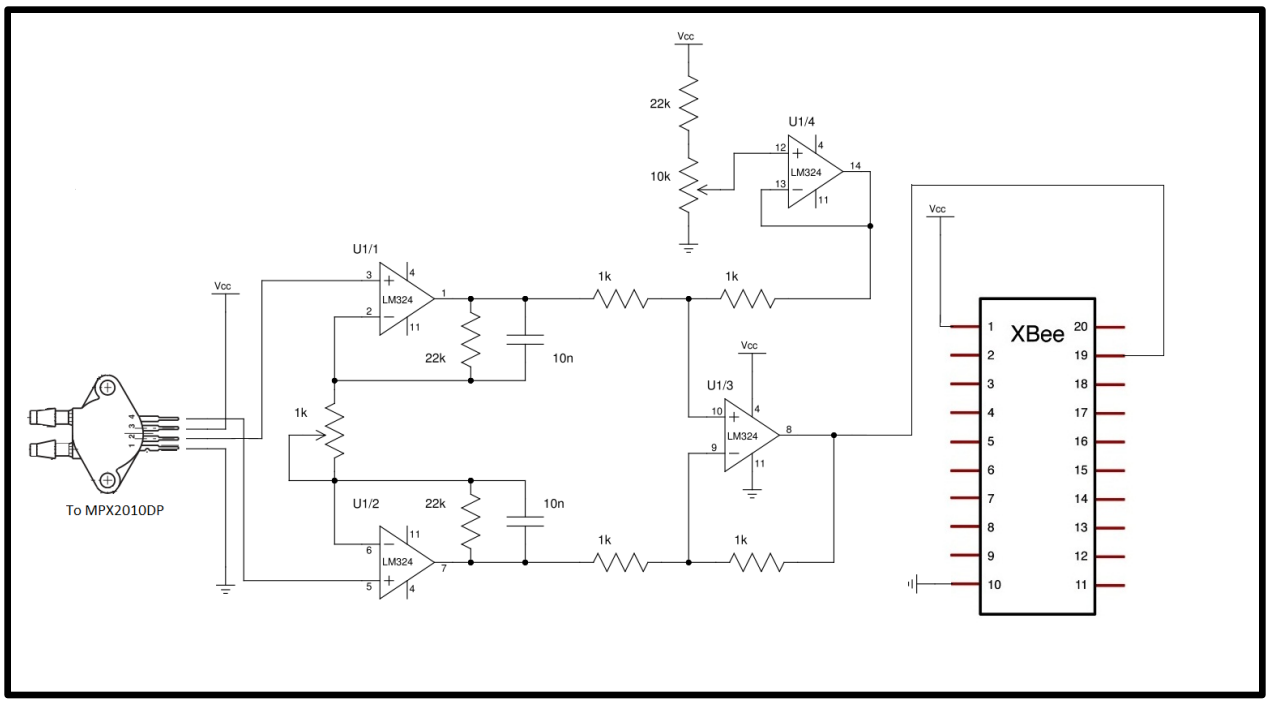


Figure 4-7 Remote Node with Pressure Transducer and amplification circuit

Figure 4-7 shows the hardware for the wireless remote node. It is a simple configuration with minimal interconnection and hardware requirements. The rest of the circuit uses an array of operational amplifiers (LM324 op-amp) to amplify that voltage differential to a higher level in order to couple with the XBee radio transceiver. The XBee obtains sensor data through an I/O, in this figure showed as the XBee pin 19. The analog data obtained from the sensor is then traduced inside the XBee with and analog to digital converter (ADC). This ADC is an XBee internal device that converts a continuous analog quantity of signal into a discrete time digital representation of itself. This provided insulated measurement is then broadcasted as a radio wave out of the XBee to the coordinator node.

This node has 3 special advantages. First, this module is compact and can be deployed within 300 feet indoors up to a mile in outdoors from the coordinator node. This property of this terrain module gives the opportunity to be used in unreachable environmental scenarios. Also the module does not consume a large quantity of energy, giving high living expectancy for the module. The XBee radio transceiver has also a sleeping mode that allows the RF module to enter in a low power state given a certain time rate or a change in port value, in this case pin 19 analog port.

4.4 END NODE PCB DEVELOPMENT

In order to populate all the electronic components and simplify the circuit in a compact environment, a PCB is developed. A toner transfer method [2] is used without having to use UV light and photoresist. PCBs are inexpensive and highly reliable, they do not require a high initial cost, and are very commercial. In this chapter we will discuss the development of our PCB and all materials for development.

EAGLE cad software was used to develop the PCB design. In this software we designed and defined the schematic shapes, pinouts, and part sizes to allow for correct layout in the PCB layout editor. Auto-routing tools of the software determined the location of each active element, of the LM324 IC or component on the PCB. After placement, the routing step adds wires needed to properly connect the placed components while obeying all design rules for the IC.

The final PCB design product consists of a mirrored black-and-white bitmap of the copper pattern of the bottom of the PCB, as we can see on Figure 4-8 and Figure 4-9.

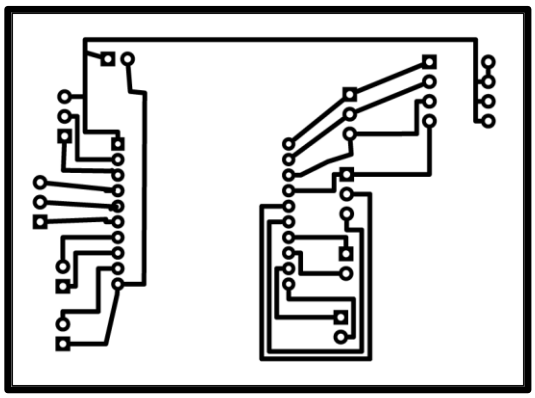


Figure 4-8 XBee node, single XBee Electronic Schematic

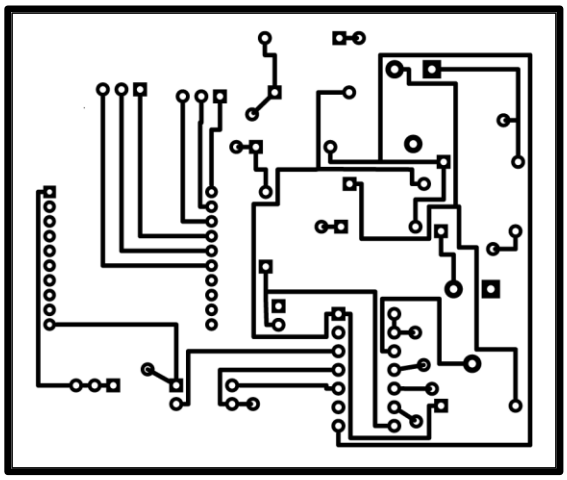


Figure 4-9 XBee Node with Amplification Circuit Electronic Schematic

Theoretically, everything you can print in a 300dpi image can end up as a copper trace, pad or letter on your print. In practice, anything thinner than 2/300 inch (2 pixels in a 300dpi image) can be problematic [2].

4.5 SENSOR NODE FABRICATION

4.5.1 MATERIALS

Materials as used in the development of the PDB with the Toner Transfer technique are presented.

Table 3 PCB Manufacturing Materials

A blank PCB, without a photoresist layer.
PrintWorks Soft Gloss Photo Paper
A laser printer with ordinary toner, 600dpi is recommended.
A clothes iron.
Acetone.
Steel wool.
Paper towel
A toothbrush or similar semi-soft brush
Etchant. The traditional etchant for copper is Ferric Chloride. Here is used Muriatic acid.
Hydrogen Peroxide 3%

4.5.2 PCB MANUFACTURE

Steps in order to develop the PCB by toner transfer method:

- Wash the copper PCB with vinegar (or a stronger acid like HCl), which will return the metal to an un-oxidized state.
    - Removing contaminants from the copper
    - Atmospheric dust
    - Abrasive particles
    - Films from other sources:
        - Solvent residue
        - H2O residue
        - Photoresist or developer residue (if it had a photoresist)
        - Oil
        - Silicone

Standard degrease

- 2-5 min soak in acetone
- 2-5 min. soak in acetone with ultrasonic agitation
- 2-5 min. soak in methanol with ultrasonic agitation

- Cutting design to minimal margins
- Sett Iron to highest setting
- Align toner print to copper board

Figure 4-10 Toner print aligned with Copper Board

- Put hot Iron over the setup. Press and remove every 5 seconds to ensure is heated evenly for 4 minutes.

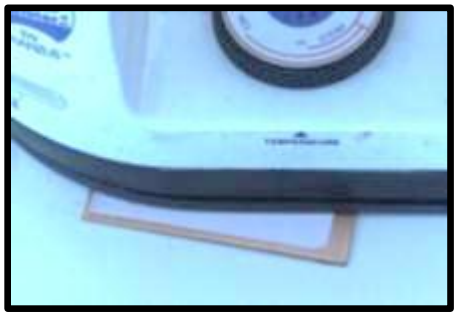


Figure 4-11 Iron Heating Setup

- Drop setup in water for 10 minutes.
- Peel first layer
- Rub with toothbrush until only the toner is on the copper surface.

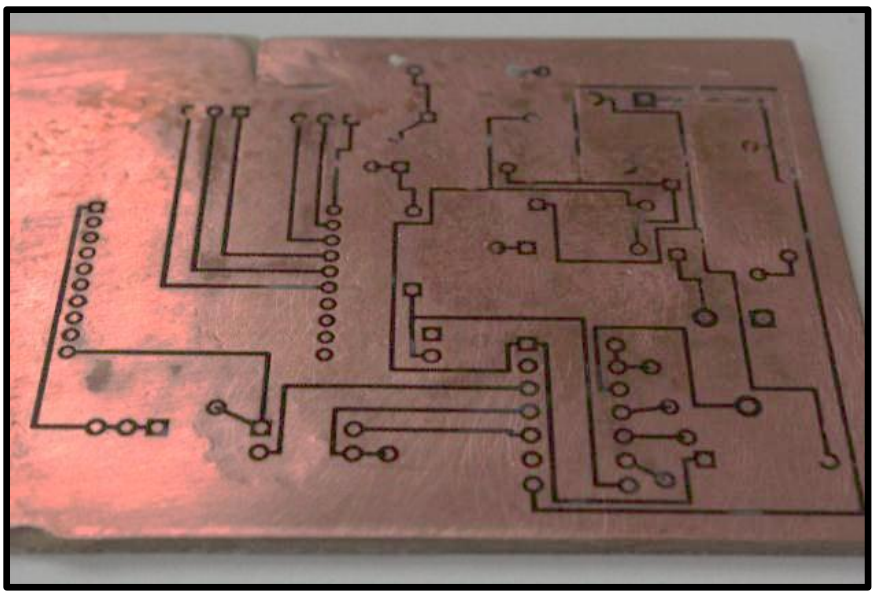


Figure 4-12 Toner Routes Adhered to Copper Plate

- Noncontiguous lines were cover with tape in order to etch and maintain copper route.

- Etch (33% Muriatic Acid, 67% Hydrogen Peroxide) 10 minutes
- After rinsing and drying the PCB remove the toner
- Populate with electronic devices :

Table 4 Electronics Bill of Materials

XBee headers
4 1k resistors
14-pin IC socket
1 LM324 op-amp
3 22k resistors

1k multi turn variable resistor
10k multi turn variable resistor
2 10nF MKT capacitors
1 100nF MKT capacitor
2 LEDs
1 10pin Female Header

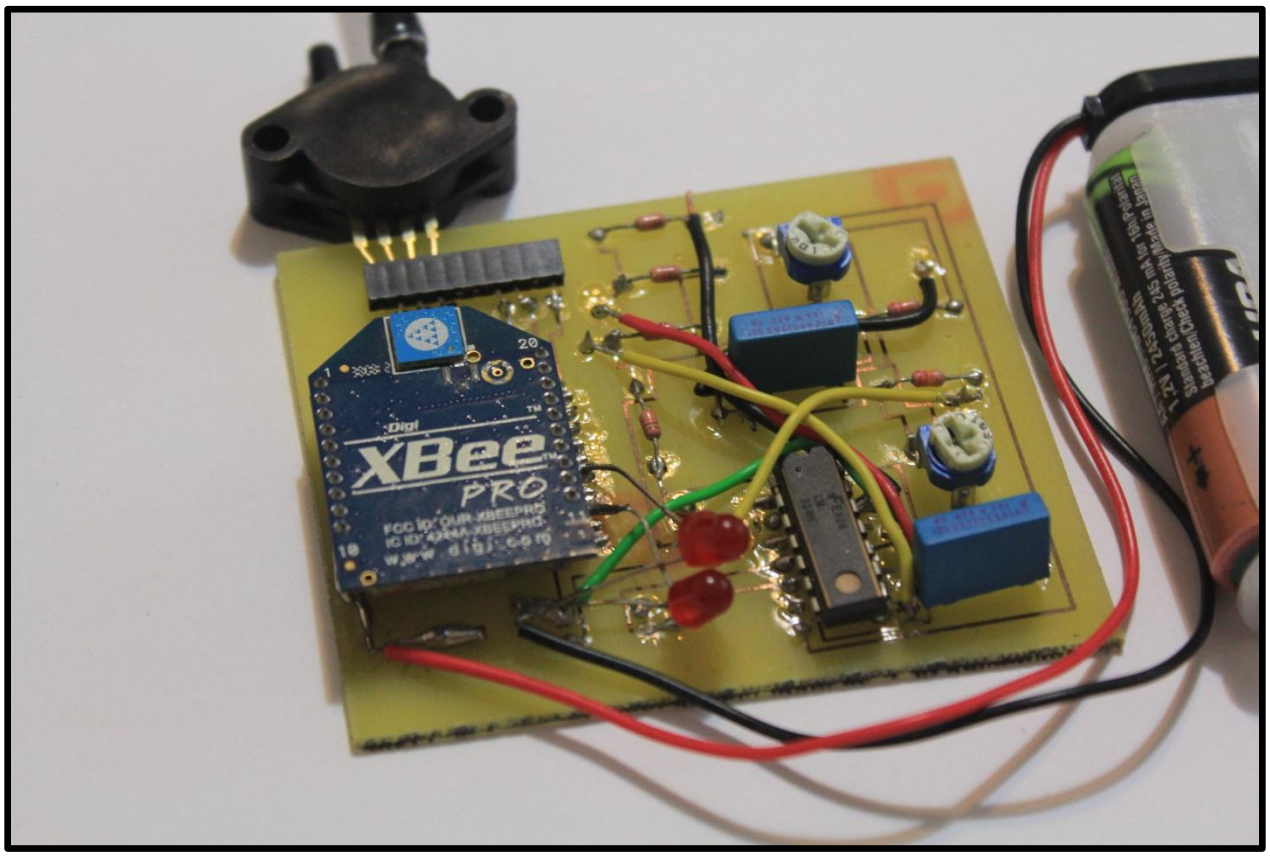


Figure 4-13 Wireless node without solder mask

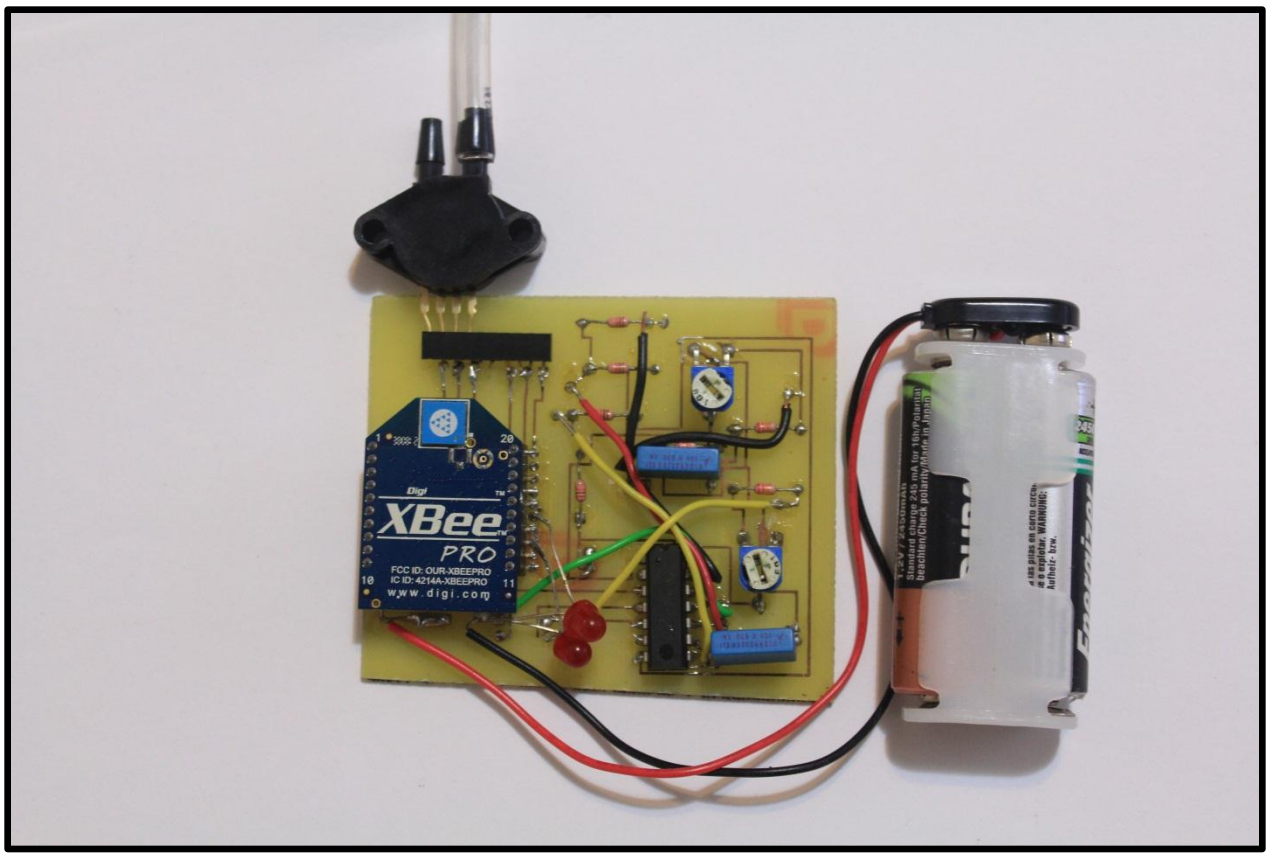


Figure 4-14 Wireless node setup with AA batteries

- After populating the electronic components into the PCB we applied solder mask (nail enamel) in order to protect copper routes.

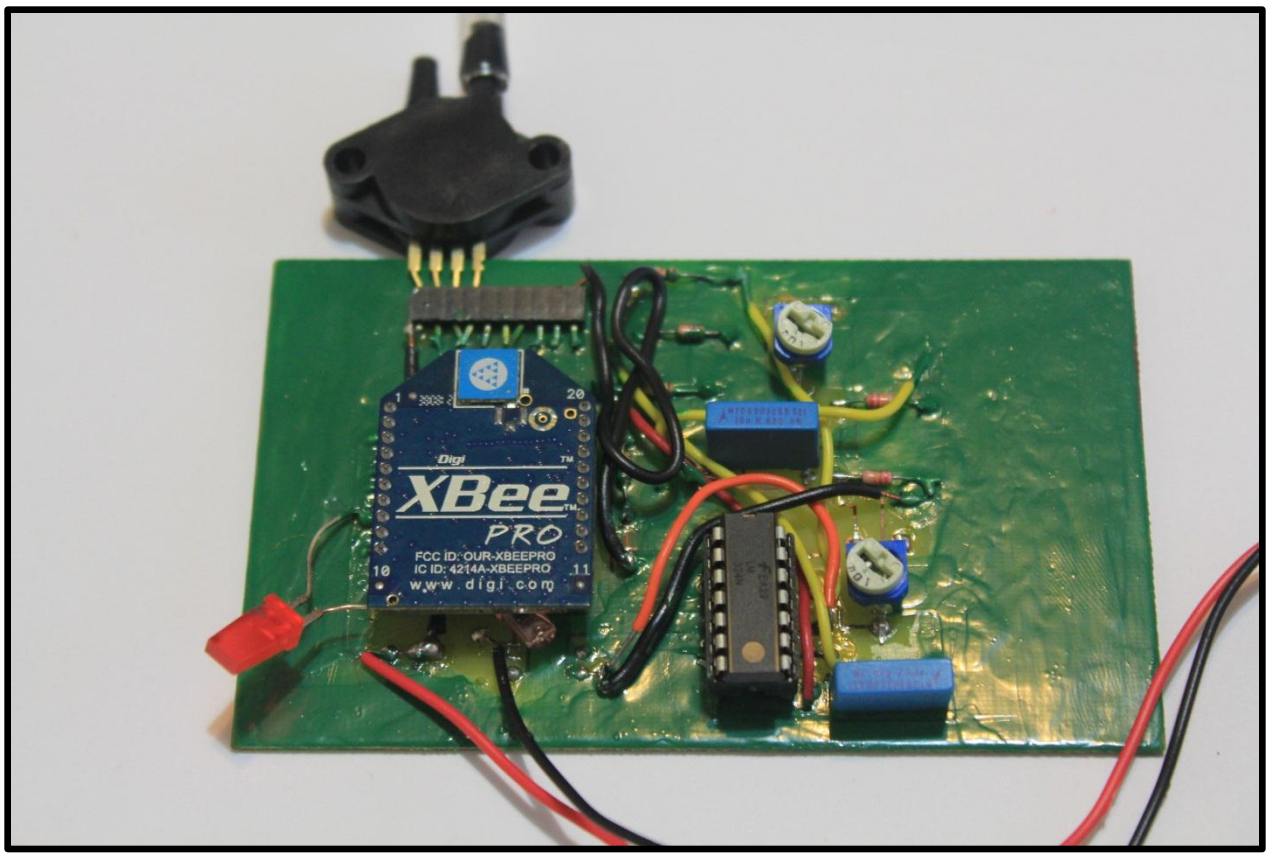


Figure 4-15 XBee wireless node with solder mask

After placing the enamel was verified that the circuit was fully operational. If there is a case where the circuit is damaged by any means in order to access the copper routes must be submerged under Acetone for 30 minutes until weakening of the solder mask, then scrubbed until copper appears. This solder mask provides a permanent protective coating for the copper traces of our printed circuit board (PCB) and prevents solder from bridging between conductors, thereby preventing future short circuits.

4.5.3 POWER CONSUMPTION

There is many variable when estimating the power consumption for the end node. One of the fastest way to estimate the node per consumption under different given scenarios is to do it experimentally. The XBee module consumes about 50mAh at full capacity, also we have to have in mind that this varies depending on the signal packet size, the MPX2010DP pressure transducer consumes about 6mAh, the LM324 operational amplifier consumes 1mAh and the LED is 10mAh.

Based on these numbers we compute their power consumption as 60 mA, assuming led is not constantly on. The amount of power consumed at sleep mode is negligible. To calculate the time spent receiving or transmitting we calculate the time needed to process frames of 160 – 176 bits long at 9600 baud rate. We get a total of 35 ms if a frame is received and resent.

Transmit Rx	22 bytes	176 bits	18.3 ms @ 9600 baud
Receive Packet	20 bytes	160 bits	16.7 ms @ 9600 baud
Full Power			35 ms

At full power the node consumes 60 mA, using 2 AA batteries of 5,000 mAh in about 83.333 hours almost 3.5 days. But this is not the case since most of the time the nodes are at sleep mode. To get realistic numbers we use the time spent transmitting and receiving which is about 35ms. We will call this transmit/receive cycle a Tx/Rx slot. Given a total of 83.333 hours of battery power, we get 8571429 Tx/Rx broadcasted slots per battery life.

83.333 hours x 60 min x 60 sec x 1000 ms / 35 ms = 8571429 Tx/Rx slots

For each server broadcast we can assume that each node spends 3 Tx/Rx slots receiving the broadcast and sending RSS data back to the server. We get 2857143 Tx/Rx broadcast slots (TRBS). This means that each node lasts for c server broadcasts after the battery is drained.

8571429 Tx/Rx slots / 3 per broadcast = 2857143 Tx/Rx broadcast slots (TRBS)

If we set a 10 second time interval between each server broadcast, we get 7936.5 hours of operation for a node which translates to about 330.5 days of operation.

2857143 (TRBS) x 10 second broadcast interval = 7936.5 hours ~ 330.5 days

These figures can improve by increasing the network baud rate and decreasing the broadcast interval. The figures relate to the prototype implemented. We do not claim the correctness of these calculations. They are intended to be used as a guide.

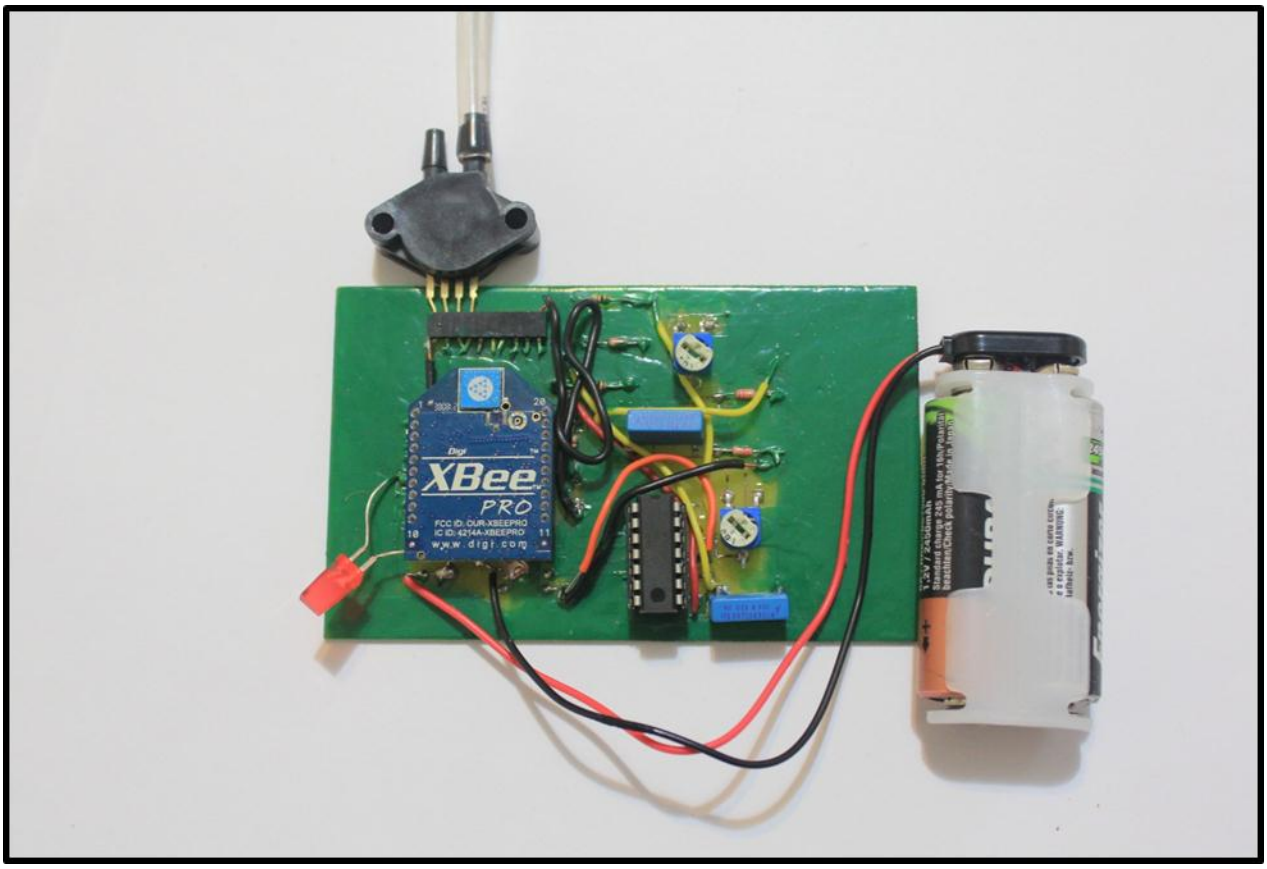


Figure 4-16 Wireless Node with AA Battery

4.6 WIRELESS COMMUNICATION UNITS

The coordinator node typically performs functions such as managing the nodes, collecting and analyzing the data received from sensor nodes, and connecting as a gateway for remote data access. Also propagating request to each node and dealing with the queries. In order to provide an optimal integrated solution for distributed data collecting there were implemented two different types of coordinator nodes. A PC based server and an Arduino based server were used to manage coordinator nodes

Different capabilities can be reaches with these two central nodes. Using the PC as a central server will increment the system overall budget, but also gives the system more managing power over the coordinator radio and all data obtained from it. Using an Arduino will reduce the cost in a great extent, also gives us a very good base and programing language in which we can manage, understand, analyze, and interpret the data to fit our main purpose. Also its more compact, does not consume much energy and easy to manage, some advantages that we cannot achieve using an ordinary computer.

4.6.1 XBee Configuration

Both systems use the XBee radio transceiver to communicate with the end nodes. This means that the radio internal setup is similar to each other's. The internal setup is a modem configuration profile which is composed of multiple parameters that enables the XBee radio transceiver to communicate at a given speed and network address. Also enable a single XBee module to be configured as a coordinator node, radio mode, or as an end node.

4.6.2 PC Base Station – Coordinator Node

As previously mentioned we used a computer to manage, understand, analyze, and interpret the data from the coordinator node. The infrastructure we will be implementing is composed of one XBee radio transceiver connected by an USB to UART serial converter, as shown in Figure 4-17 and Figure 4-18 . The computer interacts with the coordinator node, setting up and searching all available signals from the deployed end nodes. Witch at the time are working gathering sensorial inputs and broadcasting them, values which we will then use for the analytics.

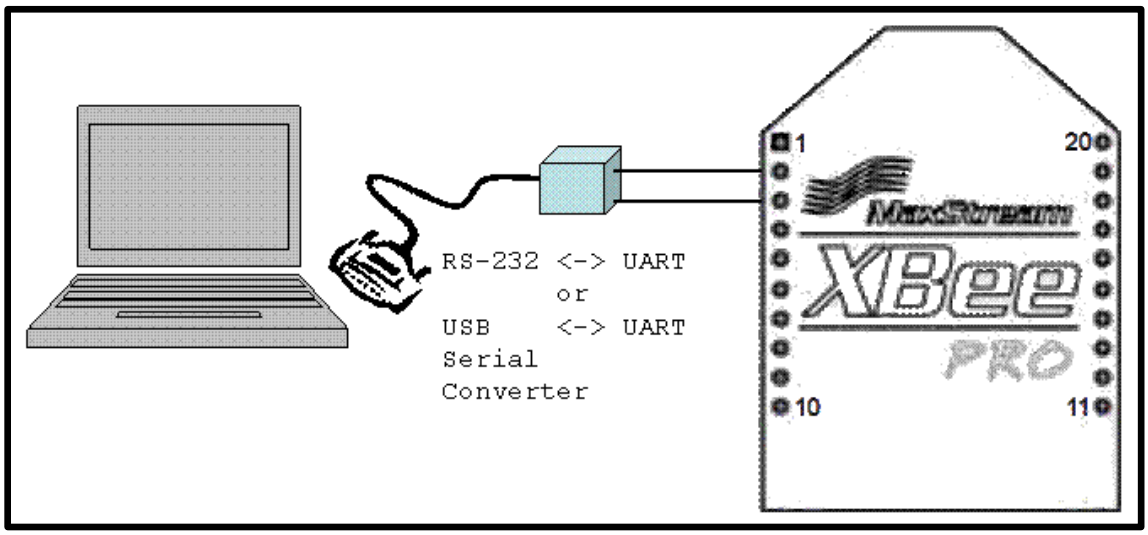


Figure 4-17 Coordinator Node Interface

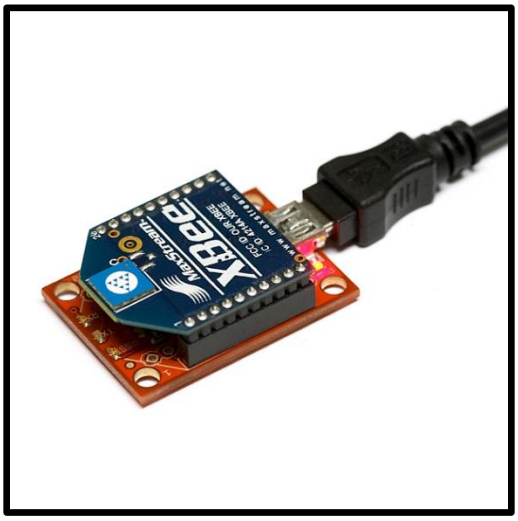


Figure 4-18 XBee Coordinator Node

On Figure 4-17 we can see the basic interface requirements for the serial communication with the PC. They employ the use of the pins 2 (UART data Out) and 3 (UART data In) in order to

manage all the XBee system content and gather the end nodes data. In Figure 4-18 we can see the actual hardware interface used, a XBee break out board the enables the connection with this two UART ports and also power the radio with the computer own energy.

4.6.3 ARDUINO UNO –COORDINATOR NODE

Arduino is microcontroller system that can be used as a stand-alone for many purposes. In our project Arduino brings to the table a number of useful properties that will later enable us to send the data to an internet network. Similar to 4.6.2 above we can use the Arduino microcontroller to manage, understand, analyze, and interpret the data from the coordinator node. But also gives us the advantage to connect to other networks and broadcast the signal to the internet.

The coordinator node, as depicted in Figure 4-20 Arduino Coordinator Node, is composed of an XBee radio transceiver with the Arduino UNO board and a Wi-Fi shield (WISHIELD 2.0).

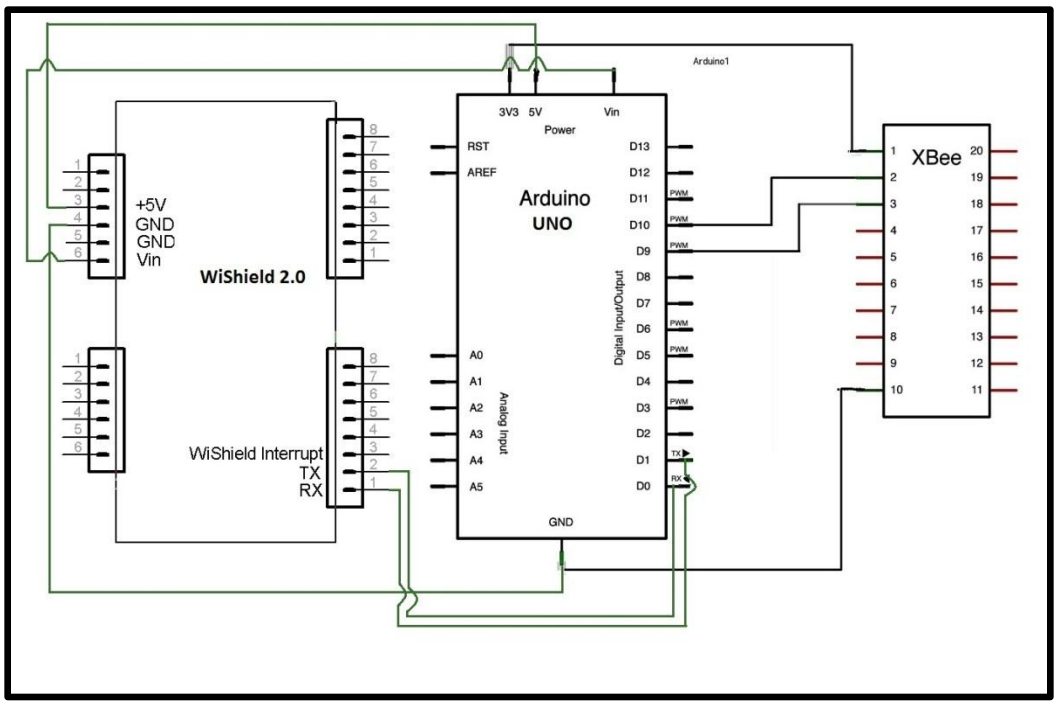


Figure 4-19 Arduino Coordinator Node Schematics

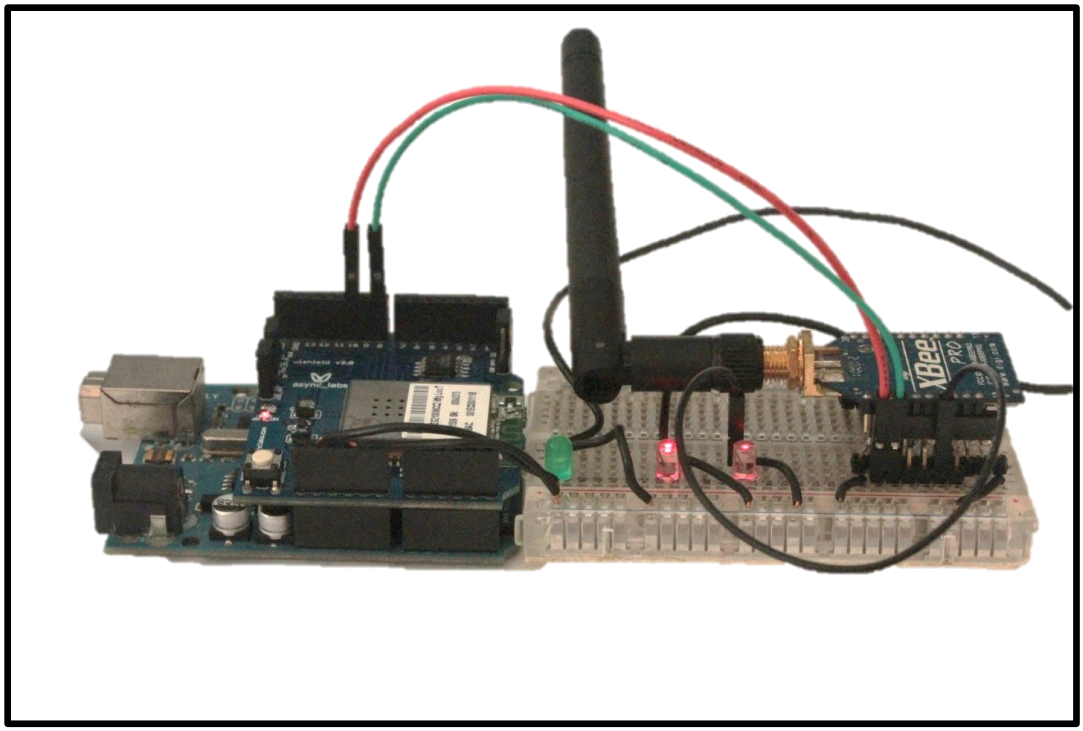


Figure 4-20 Arduino Coordinator Node

In Figure 4-20 above it's shown the basic interface between the XBee module and the Arduino microcontroller. Similar to section 4.6.2 above the Arduino requires the connection of the pins 2 (UART data Out, red) and 3 (UART data in, green) from the XBee radio module. This is essential in order to manage all the XBee system content and gather the end nodes data.

Also as a part of this coordinator node some physical identification elements were added. This helps to visually monitor coordinator node activities, without the use of any display, and take action accordingly. AS depicted on Figure 4-20 Arduino Coordinator Node, there are 3 LEDs in the Coordinator station as indicators. The green LED is placed on the association indicator and the ground, it bright steadily while the radio searches for a network. The red LED to the left is the On/Sleep indicator pin and tells if the radio is getting power and currently awake. The second red LED is the Received Signal Strength Indicator (RSSI), will light up when the radio receives information that is addressed to it.

4.6.4 Power Supply

As discussed on section 4.5.3 the XBee module uses around 50mA which is a depreciable amount when relating it to the energy consume of the computer. The PC coordinator module has an XBee shield called the XBee Explorer, a simple to use serial base. This serial base provides the XBee radio a direct access to the computer serial port and energy. This power will be available while the computer still on.

The Arduino coordinator module gives us a power advantage by not consuming nearly as much as a common computer. The maximum Arduino consumption is 50mA, important to notice that it will be the same consumption of an XBee radio. The more power consuming element is the internet shield (WiShield) with a power consumption of 230mA. This coordinator also can be powered from an USB port from a computer or from a power supply direct from a common home energy outlet.

CHAPTER 5

WLSN SOFTWARE DESIGN

5.1 INTRODUCTION

In order to establish a sensor interface, configure the mesh network, and manage the sensed data, we must have software that performs the specified task to obtain our liquid level measurement. Each network elements has a set of instructions that binds them to their specified task. The algorithm creates multiple events to analyze, graph and display the data in order to present and graphic interface easily to read and understand by the user.

Since there are two different server stations, PC and Arduino, there are specific codes that fit in each of the server stations, each one with a specific task work in order to analyze the data.

This section describes in detail our implementation of a real time liquid level monitoring system for wireless mesh networks. First we'll describe the system as a whole. Then we describe the wireless nodes behavior and internal algorithms. Following that is the coordinator node data gathering, and filtration algorithms. We will also describe the data flow in the network of nodes.

5.2 SENSOR INTERFACE SOFTWARE DESIGN

As mentioned in section 4.2, as a sensing unit we used the MPX2010DP pressure transducer. The MPX2010DP is rated to 10kPa in differential pressure; this is equivalent to just over a meter of water dept.

In order to apply a standard practice over the pressure transducer on the operational environment we must define the behavior or this sensor through a gamma of pressure. Correlating the experimental pressure from the analog value we can design an algorithm for this sensor. To relate these two values we made a sensor characterization using a small water container of 16 inches and varied liquid level in order to obtain digital data in bytes.

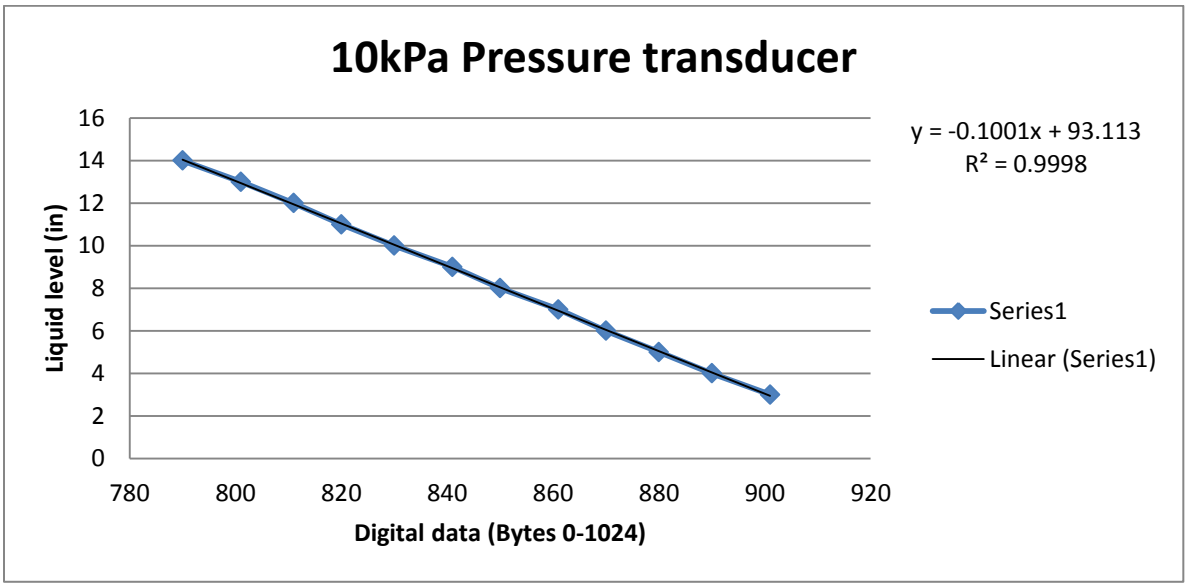


Figure 5-1 Experimental results for the 10kPa pressure transducer

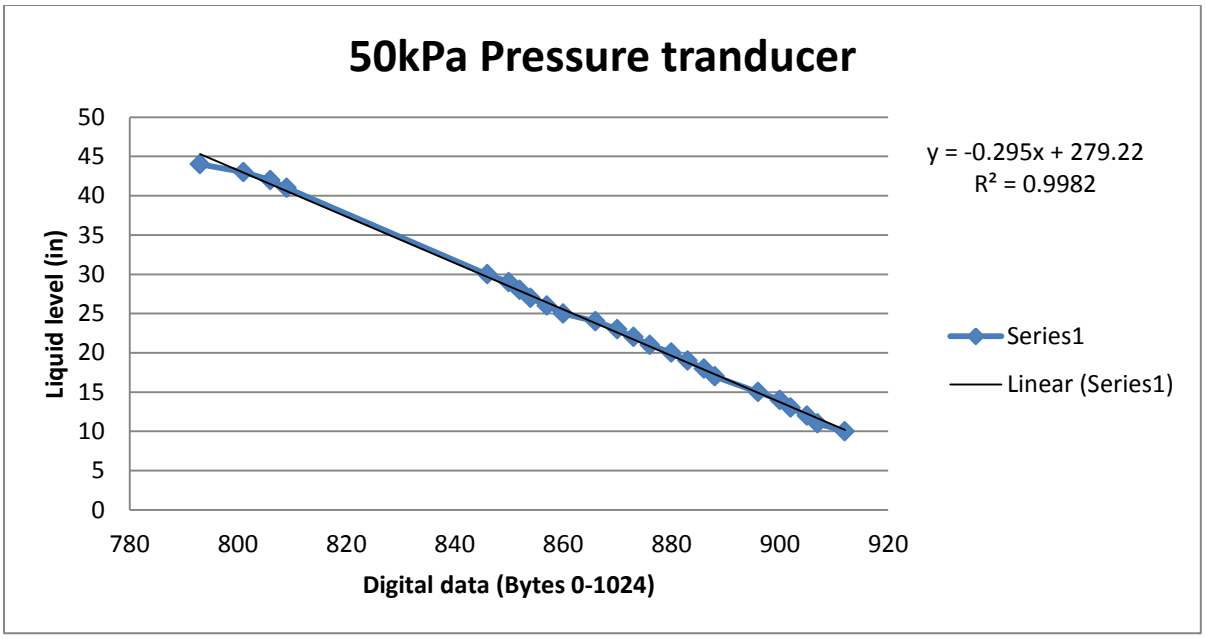


Figure 5-2 Experimental data for the 50kPa pressure transducer

Using both of this characterization equation we were able to simulate and design a digital to liquid level translation of all wireless nodes for our main algorithm.

5.2.1 XBee Module Configuration

In order to get to the hardware interface, XBee radio transceiver, properly paired and working it has to have a specific setup configuration. This configuration will enable each node to interact with the base coordinator XBee, as well as the coordinator node with the end nodes. To use the XBee to their full potential we used them in API mode, as described on chapter 1. This mode specifies the application level protocol in which modules and devices talk to each other, be it wirelessly or via serial port. The following procedure demonstrates and describes the steps of

setting up the first pair of nodes before sending data. These steps were done after the XBee radio modules were upgraded, the remote and the coordinator mode.

5.2.2 PAIRING AND OTHER PARAMETERS

We need to pair the XBee modules so they can communicate at any given speed, in this case high speed, automatic resending of lost packets. For this it is set the registers on "Modem Configuration" in X-CTU.

XBEE radios only operate at a given baud rate[4], this is the number of bits per second that the XBEE can send. A brand new XBee will default to 9600bps, which is pretty slow. We can change the baud rate, by changing the ATBD[5] register. Both of our XBee's have to be the same baud rate to talk to one another. The available baud rates (and corresponding ATBD value) are:

- 1 = 2400bps
- 2 = 4800bps
- 3 = 9600bps
- 4 = 19200bps
- 5 = 38400bps
- 6 = 57600 bps
- 7 = 115200 bps

---

[4] **Baud Rate** is the number of pulses per second, also known as modulation rate.

[5] **ATDB** is the dedicated XBee AT command for adjusting the baud rate at the module. This can be adjusted prior to use.

The next parameter of interest is the Personal Area Network ID. This is a number shared amongst each XBEE in a network. In this part of the experiment we were using 2 XBee radio transceivers, but we could have many more in a single network. XBee radio transceivers that are transmitting on different networks do not "see" each other. The default PAN[6] is 3332, so I avoid that number. The PAN ID is stored in ATID.

Once both of our XBee radio modules are on the same network, we can give each one an address number, denoted by ATMY. We can also set the destination address, which is what address number to talk to, denoted ATDL (for destination low, we really don't need to use the high bytes if we keep our address numbers < 16 bits in length). A sample setup of two XBee radio modules that will talk directly to one another, at 38.4kbps:

The first set up for this experiment was these parameters.

Table 5 Parameters for XBee radio module initial setup

XBee 1		XBee 2		XBee 3		XBee 4	
ATID	1111	ATID	1111	ATID	1111	ATID	1111
ATMY	10	ATMY	11	ATMY	11	ATMY	11
ATDL	11	ATDL	10	ATDL	10	ATDL	10
ATBD	5	ATBD	5	ATBD	5	ATBD	5

[6] **PAN ID** stands for Previous Access Network Identifier

From this point on the XBee radio modules will be available to communicate with each other. In this initial setup we used only the first 2 modules, but we configure the rest in order to use in next chapters. This setup will enable us to know the capabilities of XBee to maintain communication harmony between nodes. But in order to explode maximum use and to acquire the data from the sensors the XBee must have the hardware interface that permits the gather of data. These steps will be discussed on next chapter.

5.3 DATA MANAGEMENT SYSTEM SOFTWARE DESIGN

Data management programs where developed in Processing language, a language built on the Java programming language, but uses a simplified syntax and graphics programming model.

The Arduino and PC programs read the real time data strings from coordinator nodes, convert them to numerical values and present the data in three different ways. The Arduino program sends the information to a host website that can graph sensor value in a web page. This web page can be accessed from any part of the world.

There are two main algorithms designed for the computer based coordinator. The first one presents a graphic unit interface for three XBee radio transceivers. This includes the primary information of deployed nodes and presents a visual representation and numerical values of sensed data.

5.3.1 PC – GRAPHIC USER INTERFACE

This software interacts directly with the coordinator node in order to gather the sensor input data from the remote node. To make this happen it would be needed to create functions that can directly request or send commands to the XBee coordinator node.

Currently there are different types of libraries that provide serial and parallel communication. In the PC server algorithm we used a library that facilitates receiving multiple sample I/O packets in API mode (ATAP1) from the 802.15.4 XBee radio module, and returns an object with the data to analyze. This library is specific to Processing and for XBee radio module. It was developed by Rob Faludi and Dan Shiffman.

The Figure 5-3 is a screen print of the results of the algorithm. It is easy to understand and easy to digest the information presented to the user. It is used for 3 wireless sensor nodes and displays the needed data.

In the Figure 5-3 we can see the distribution of XBee nodes by colors. Also their respective analog values are at the left of the GUI, in the respective Value Box for each end node. Seeing the figure can notice that the numerical values are consistent with the graphic value. Taking as example the second node, this node is coded as red and a red line in the graph. By visual inspection we can notice that the final value taken was most likely 3.1, going to the second value box at the left we can see that the analog value being reported by the software is 3.16325, pretty near but more accurate than the graph. This specific graphic representation was made using a potentiometer sensor to relate rotation with time to get the analog value in order to verify data

accuracy. We used potentiometer sensors in order to have a better understanding of the graphic behavior of fast and abrupt changes.

This algorithm also reports the sensor node address and received signal strength indicator (RSSI), important parameter when configuring the nodes or identifying module signal.

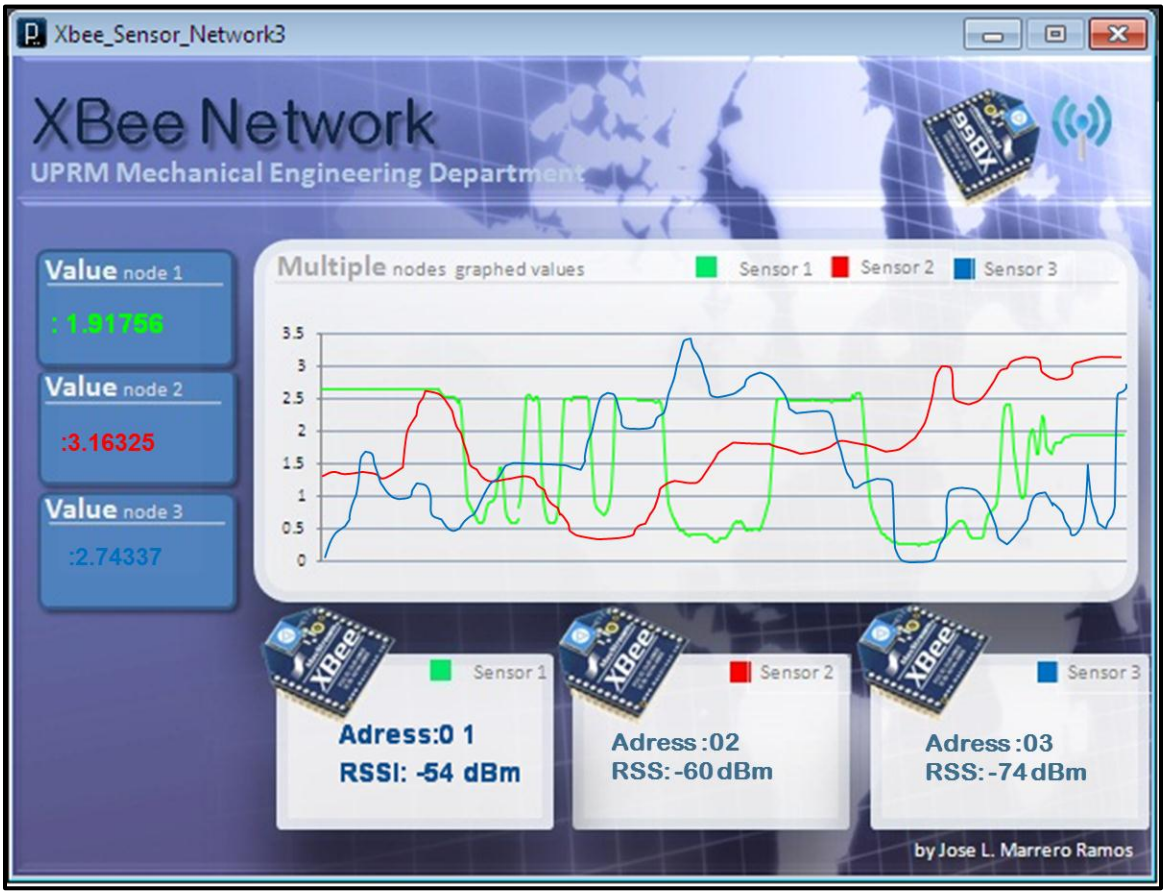


Figure 5-3 Wireless Sensor Network GUI, For 3 XBee Nodes

This algorithm has a minimum requirement of two XBee series 1, one remote (broadcasting three analog values) and one coordinator (attached to the serial port of the computer). This software will detect a node when powered on and work accordingly.

5.3.2 ARDUINO – MICROCONTROLLER SOFTWARE

The Arduino has his own programming environment, based on Processing language. The code is developed and then uploaded to the Arduino microcontroller from a PC. Similar to last algorithm, the objectives of this algorithm is to interact directly with the coordinator node in order to gather the sensor input data from the remote node.

The Arduino software is in charge of multiple elements at the same time; and because of this multiple algorithm events were made in order to obtain harmony between them.

To obtain the sensor data an Arduino-XBee library was implemented. This library enabled us to communicate with the coordinator node and obtain the end node sensed data. Once obtained this data the Arduino software interpret the data to liquid level values.

Using the Arduino microcontroller and the Wi-Fi Shield we host given sensor data to Pachube, in a similar way that we did with the PC coordinator system. First a WiShield program was implemented in order to initialize a connection between the local network and Arduino. After the sensor data is interpreted it is sent to pachube by a post command.

5.4 ONLINE DATA HOSTING

Our software design is capable of handling the data and presents all nodes data. Also an algorithm was implemented in order to host the sensor data to the web in order to monitor the data remotely from any part of the world.

There are multiple ways to share data online; in our algorithm, both for the computer server and the Arduino server, we implemented it to host the sensed data to pachube.com [15]. We use this website as the on-line database service provider allowing the modules to host sensor data to the web.

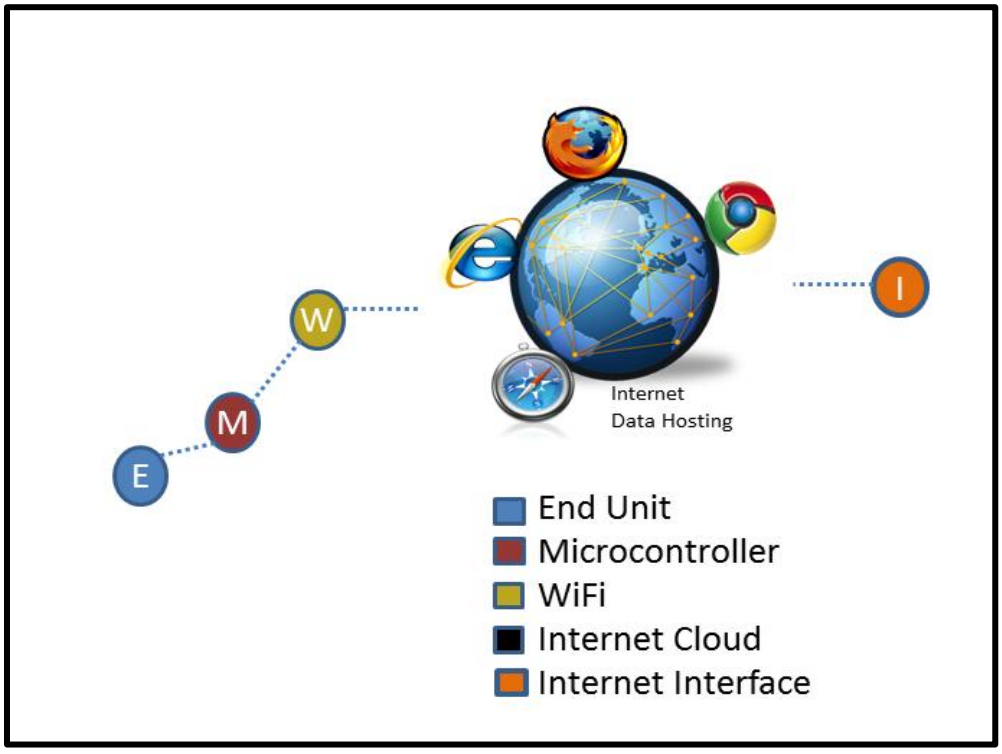


Figure 5-4 Network Topology

Seeing the high view network topology described in Figure 5-4, the microcontroller and Wi-Fi are consolidated in a PC. In the case of the Arduino based model, the microcontroller is the Arduino and the Wi-Fi unit is the WiShield module for Arduino.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 INTRODUCTION

This chapter presents results of experiments conducted on a working implementation of our proposed software middleware for real time liquid level system. The experiments help validate our solutions and ideas, as well as serve as a demonstration of the system. The idea is to set up the working network of wireless nodes and a central server using our software, and prove it reasonably functional.

Figure 6-1 shows the actual network topology for our experiments. As stated throughout this thesis our wireless sensor network composes of three end nodes and a central server. To realize the designed network topology, XBee modules were configured to behave as an end device or coordinator. The wireless end nodes have several tasks to do besides creating and maintaining the wireless mesh network. Each node receives and performs commands sent by the central server (computer or microcontroller), also return analog values to the central server. The central server (computer or microcontroller) software is in charge of managing the coordinator node, configuring the wireless nodes in the network, and calculating digital values in order to process it, graph it and host it online through a web hosting service called Pachube. Digital information is used to draw the water level in the graphic unit interface.

These two different systems (Computer and microcontroller) share the same topology without sharing the same hardware. The components of each system are depicted on Figure 6-2 and Figure 6-3, for the PC central server and the Arduino microcontroller central server. Both systems share the same end nodes sensing board and wireless module. This gives both system a great flexibility, stability, and innovation in a user friendly environment.

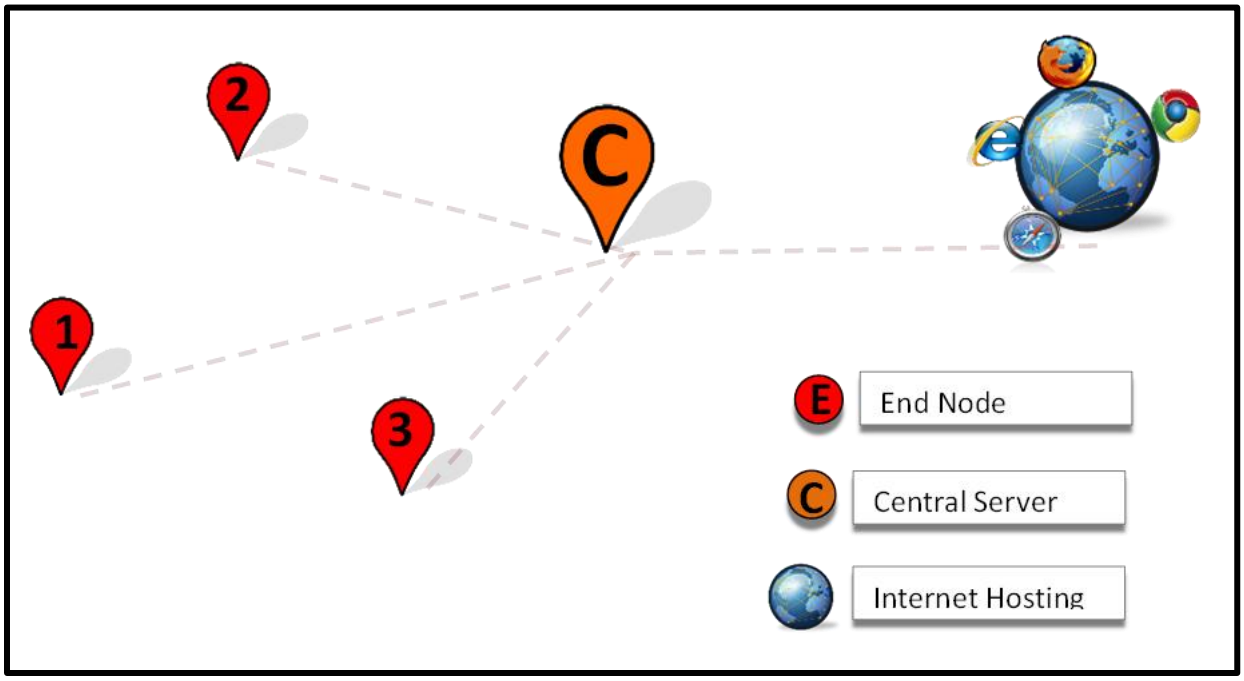


Figure 6-1 Experiment Network Topology for Both Central Servers

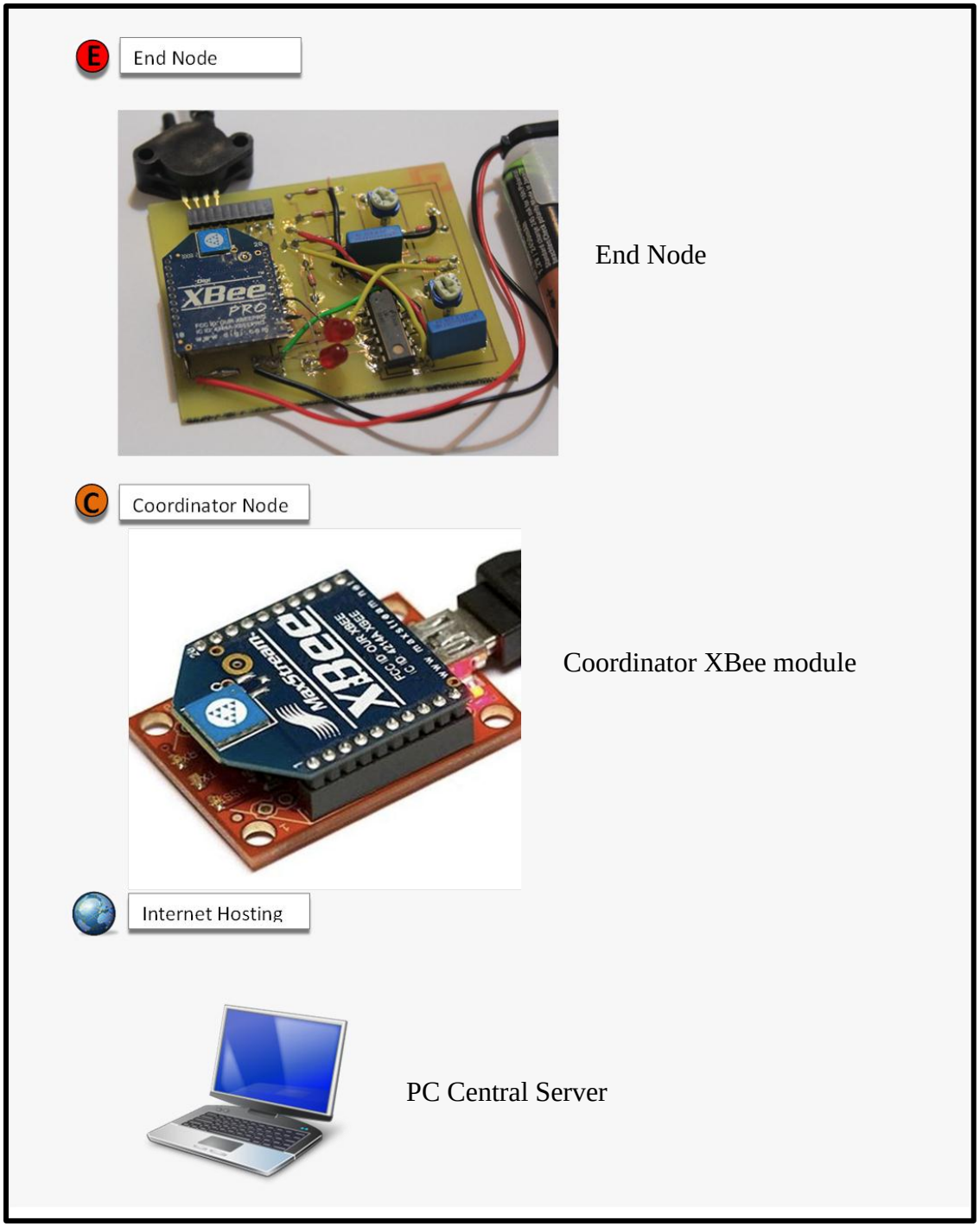


Figure 6-2 Wireless Sensor Network Components with PC-Based Censtral Server

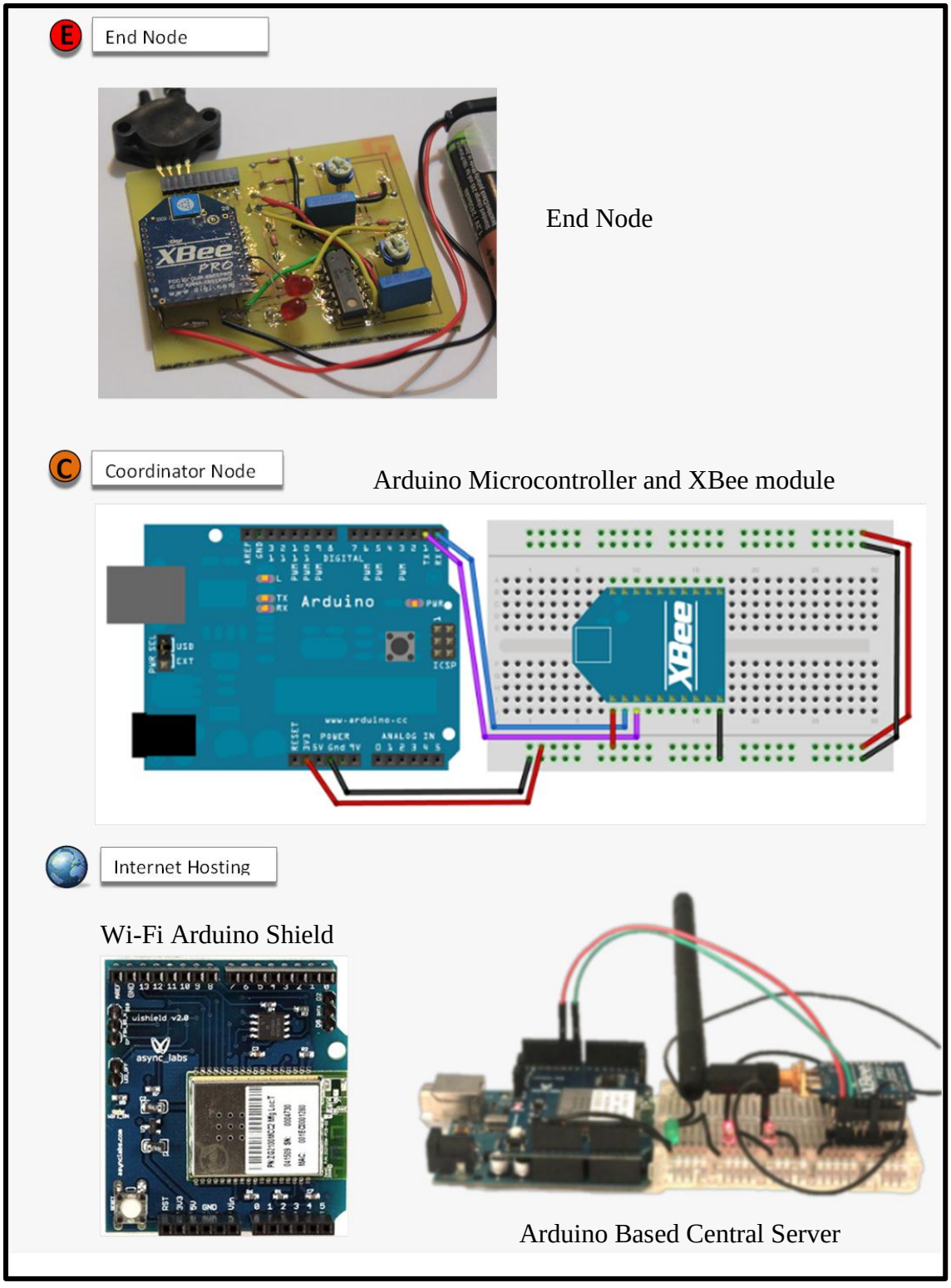


Figure 6-3 Wireless Sensor Network Components with Arduino-Microcontroller-Based Censtral Server

6.2 PRESSURE SENSOR ANALYSIS

First, in order to go further and validate the software middleware as a functional system we executed a single node prototype experiment. Using this characterization equation we were able program a new single liquid level GUI for wireless node 1. One important aspect of this experiment is to obtain response in an actual experiment environment. Most important is that the sensor behaves as intended to be successfully applied in a multiple node application.

The single node graphic unit interphase contains two different visual identifiers. The first one and the key visual display is a graph in wish we can obtain the actual liquid level value in inches. The second one is a container picture at the left with a bar graph, it is pretended to be an easy and quick way to have an idea of the liquid level relative to the container size. Also the software gives us the radio identification address and the received signal strength of deployed node.

Figure 6-4 through Figure 6-7 shows the experimental results and the sensor behavior on our graphic unit interface. This series shows how the 14 inches tank is being filled up.

6.2.1 MODULES CONFIGURATION SETUP

Attached to a single container is an end node as explained in section 4.2. The settings for the XBee radio transceivers are as follows:

Base station radio:

- Personal Area Network (PAN) ID: AAAA
- Source address: ATMY 0

- Baud rate 115200 bits per second: ATBD 7

Remote radio:

- Personal Area Network (PAN) ID: AAAA
- Source address: ATMY 1
- Destination address: ATDL 0
- Analog inputs activated:
    - ATD0 2
- Sample rate 80 milliseconds: ATIR 50
- 1 sample per transmission: ATIT 1
- Baud rate 115200 bits per second: ATBD 7

Figure 6-4 Initial liquid level (0 in).

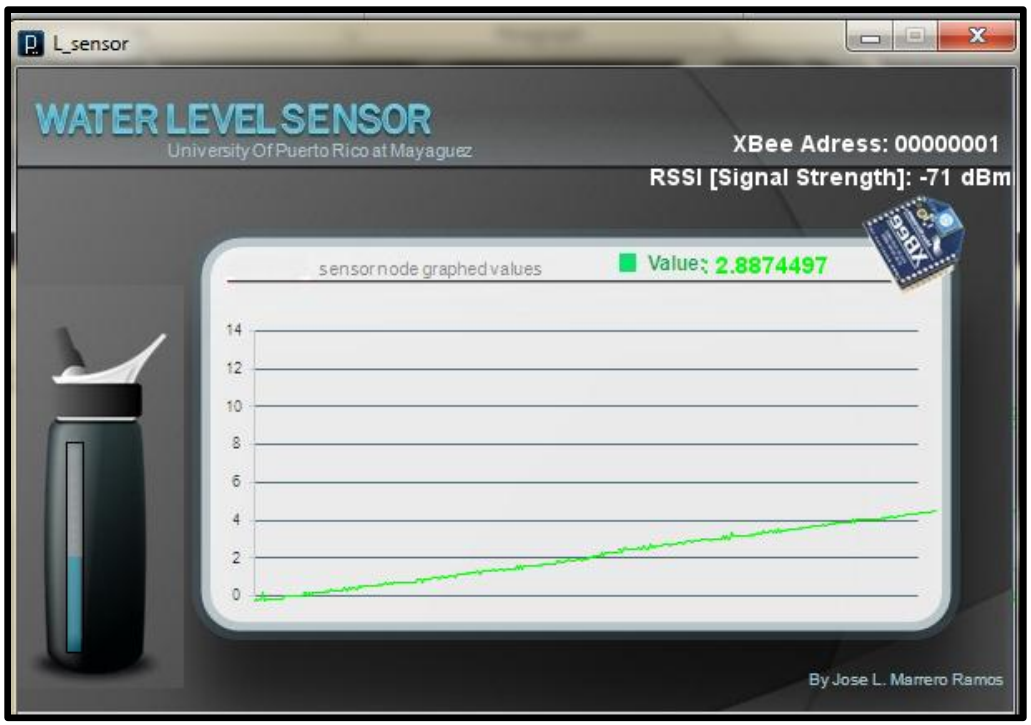


Figure 6-5 Liquid level experiment

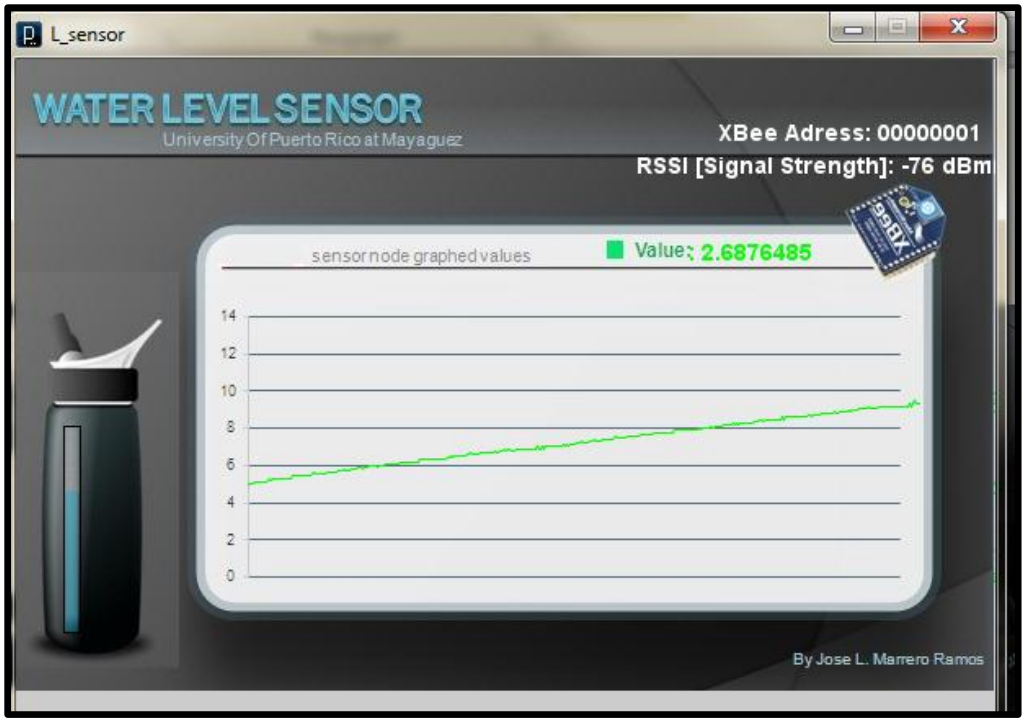


Figure 6-6 Liquid level experiment (2)

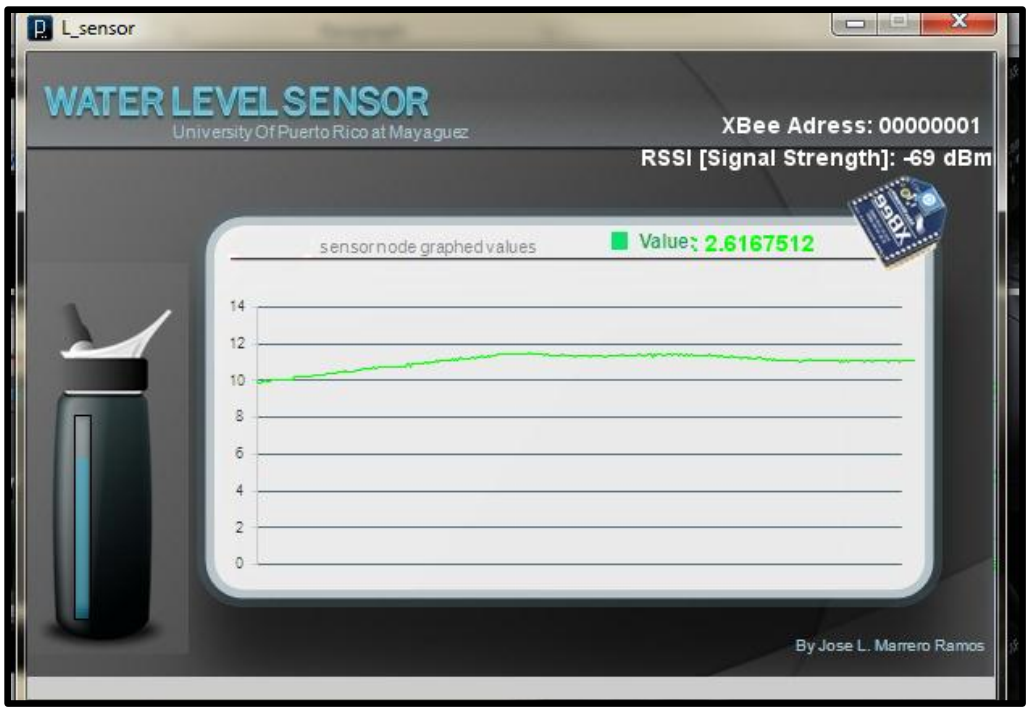


Figure 6-7 Liquid level experiment (3)

On the Figure 6-4 we can see that the system is steady and stable on the initial condition of no water. This means the pressure is the atmospheric pressure and there is no liquid on tank. It is visible some small noise, this noise is due to a variation in voltage. This variation can be caused by the water vibrations when filling the container, XBee radio setup by activating another analog port, or finally by a wrong solder. To fix this behavior a capacitor was used in other to smooth the signal.

We can see from Figure 6-5 Liquid level experiment to Figure 6-7 Liquid level experiment (3)that liquid level increment was linear until reaching full container. At the end of the graph in Figure 6-7 Liquid level experiment (3) we can see a variation of water value, this level variation is due to water waves even out.

This experiment was realized 10 times. During each experiment the liquid level in the small container was verified and matched the real value. Each value matched with the actual liquid level of small container

6.3 NETWORK PROTOTYPE IMPLEMENTATION

Now that we have confirmed a single end node system we developed a system that can put in practice 3 end nodes. With this experiment we would like to obtain the liquid level sensed value for three end nodes, test the software middleware, and prove the prototype as a functional system.

We used the same setup depicted in Figure 6-8 which consisted of three wireless nodes arranged in the base of each water tank. A PC base coordinator node was used as the subject for server connection and data gathering. Nodes are located at the bottom of the tank, in order to obtain relevant data a node height must be the same after each measurement. If the node height changes respectively to the tank bottom, since pressure depends on height, it will affect the value reading.

To notice different behaviors during the experiment different initial conditions were given to each tank. These conditions are:

- Tank 1 (Wireless Node 1)
    - Initial condition : liquid level at 35 inches
    - Changing condition: emptying tank, fast rate
- Tank 2 (Wireless Node 2)

- Initial condition : liquid level at 20 inches
- Changing condition: emptying tank, slow rate

- Tank 3 (Wireless Node 3)
  - Initial condition : liquid level at 0 inches
  - Changing condition: filling tank

6.3.1 MODULES CONFIGURATION SETUP

The settings for the radios are as follows:

Base station radio:

- Personal Area Network (PAN) ID: 58344
- Source address: ATMY 0
- Baud rate 115200 bits per second: ATBD 7

Remote radio:

- Personal Area Network (PAN) ID: 58344
- Source address: ATMY 1
- Destination address: ATDL 0
- Analog inputs activated:
  - First Node: ATD0 2
  - Second Node: ATD0 2
  - Third Node: ATD0 2
- Sample rate 80 milliseconds: ATIR 50
- 1 sample per transmission: ATIT 1
- Baud rate 115200 bits per second: ATBD 7

The experimental environment was composed by:

- One generic PC running Microsoft Windows 7, on an Intel Pentium Dual core at 2.2 GHz, and 1GB of memory.
- Three end nodes each attached to a PVC tank respectively.
- One XBee coordinator node.

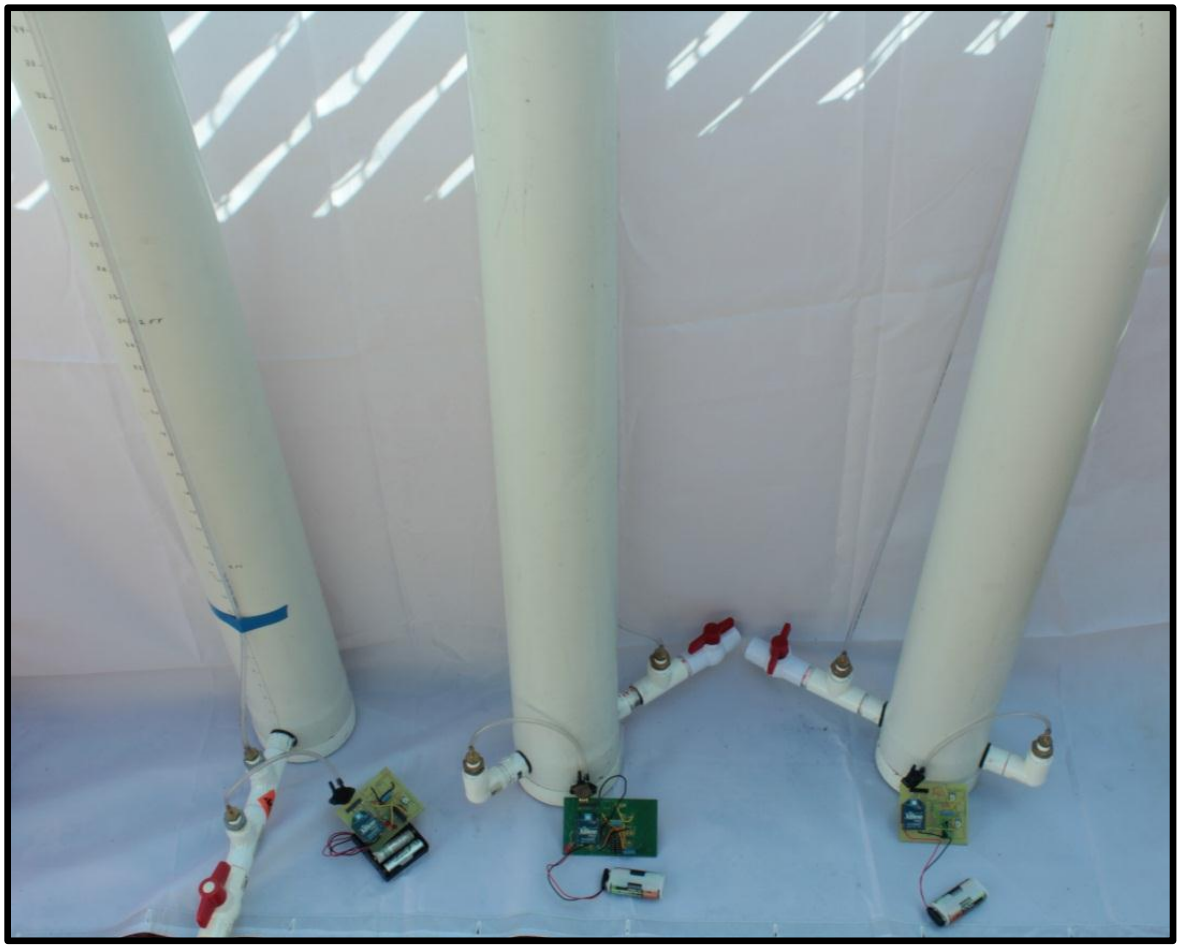


Figure 6-8 Experimental setup with 3 water tanks

Each one of the experimental tanks is made of polyvinylchloride tubes (PVC) commonly available in any hardware store. On Figure 6-9 Tank plastic tube for visual measureseach tank has a plastic gage tube in which we can measure the liquid level manually and verify this data against the data obtained on software.

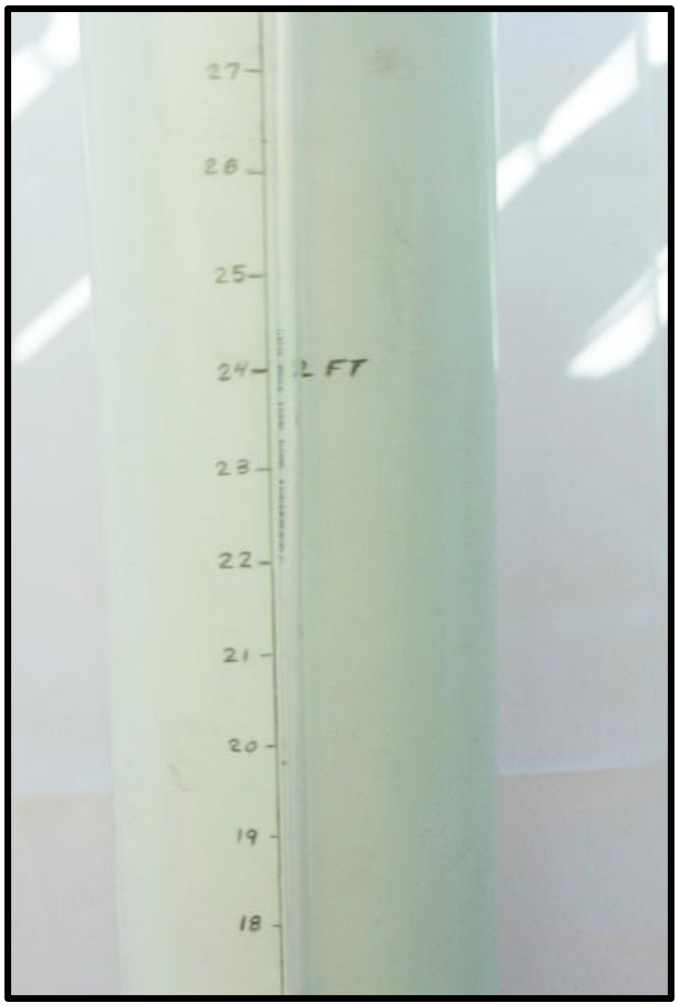


Figure 6-9 Tank plastic tube for visual measures

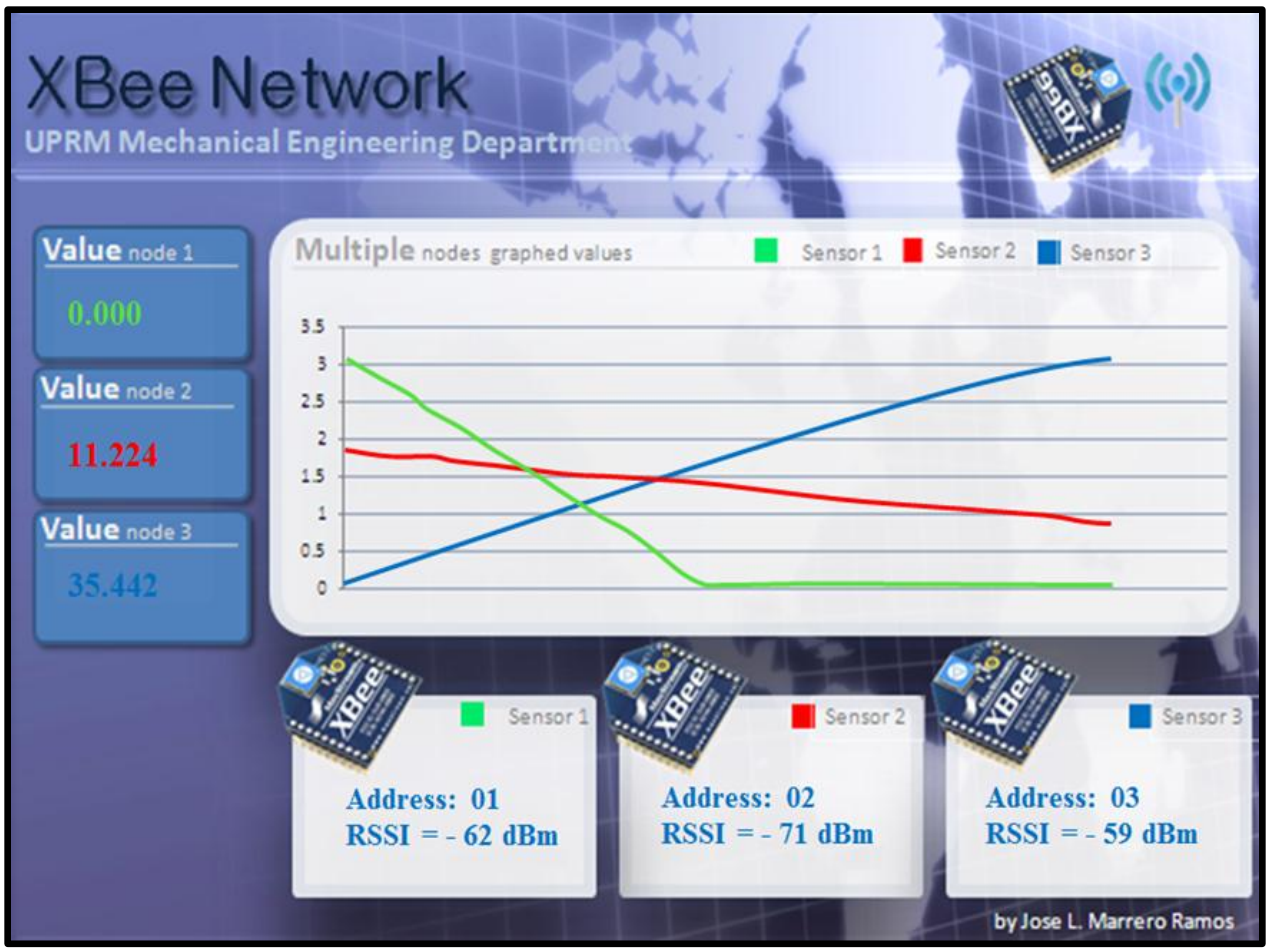


Figure 6-10 Gathered data for 3 wireless nodes

In the previous figure we can see the distribution of XBee nodes by colors. Here we can appreciate the experimental data of given run. Also their respective analog values are at the left of the GUI. We can note the liquid level change of given node respectively to the time in each graph.

Table 6 Depicted Node Setup for GUI

Green	Node 1	Tank 1	Addess:01
Red	Node 2	Tank 2	Address: 02

Blue	Node 3	Tank 3	Address: 03

The first node's (Node 1 Green) initial condition is full tank, as we can see on the graph we open the valve to a fast discharge setup. We can see that the tank 1 is empty at halftime. From half time to the end the value is zero due to lack of water.

The second node's tank (node 2 red) initial condition is 20 inches; we have the escape valve to a medium closed setup for an average water discharge rate. From the computer digital data and visual measurement the final liquid level value at tank 2 was 11.3 inches; we can find value at the second box to the left in the GUI.

For the last node (node 3 blue) the initial condition of tank's water value was 0.000 inches, in this case we added water to the tank. As we can see we have a very linear increment trough time. We can observe the final liquid level value (35.4 inches) at the third box on the left of the unit interface depicted above.

This experiment performed reasonably as expected; evidencing prototype liquid level capabilities and at the same time testing also the software middleware capabilities as a functional system. Now that we have gathered the data we would like to be able to monitor this information from any part of the web. In the next chapter we will discuss all the process in order to be able to host this gathered data online.

6.3.2 DATA HOSTING (COMPUTER NODE)

For the first set up we are using three wireless nodes and the coordinator, same setup as the experiment of section 6.3. To use Pachube an account has to be created, gladly Pachube offers a free account but only 500 data points can be received per day (24 hr.). A feed has to be registered in order to set up and let collect our data that we would like to push into pachube.

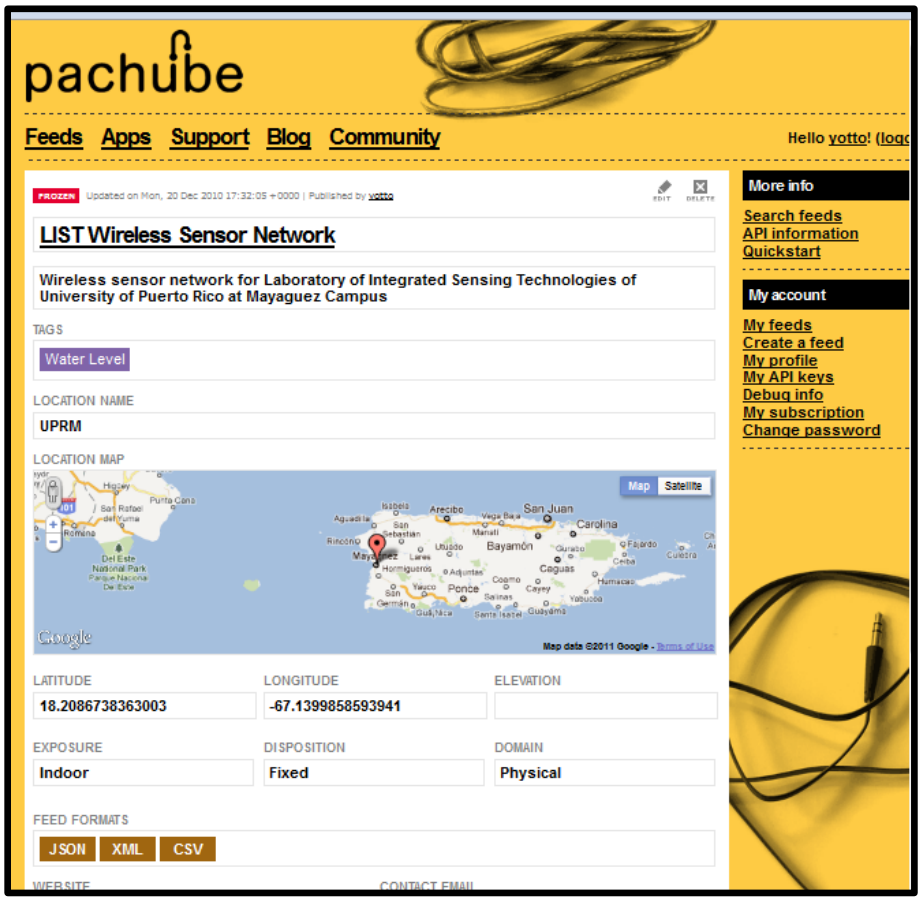


Figure 6-11 Pachube Web Interface

As shown in this figure pachube has the ability to present the localization of given node setup and export given data in different formats (i.e. “.csv”).

In order to complete this experiment we used JPachube library, this library handles the connection with Pachube. When uploading data online we have to be aware that pachube will only take 1 data every 3 minutes. In this experiment we filled up all three tanks at different speed ratios in order to see the pachube capabilities graphing our liquid level and at the same time implement the simple sensor network pachube code [20] and test as a functional system. We want to verify that our implementation performs reasonably as expected.

Similar to the last experiment we used the same setup depicted in Figure 6-8. The PC base coordinator node was used as the subject for server connection and data gathering. Nodes are located at the bottom of the tank, in order to obtain relevant data a node location must be the same after each measurement. If the node height location changes respectively to the tanks we will be posting the data for given height, since pressure depends on height.

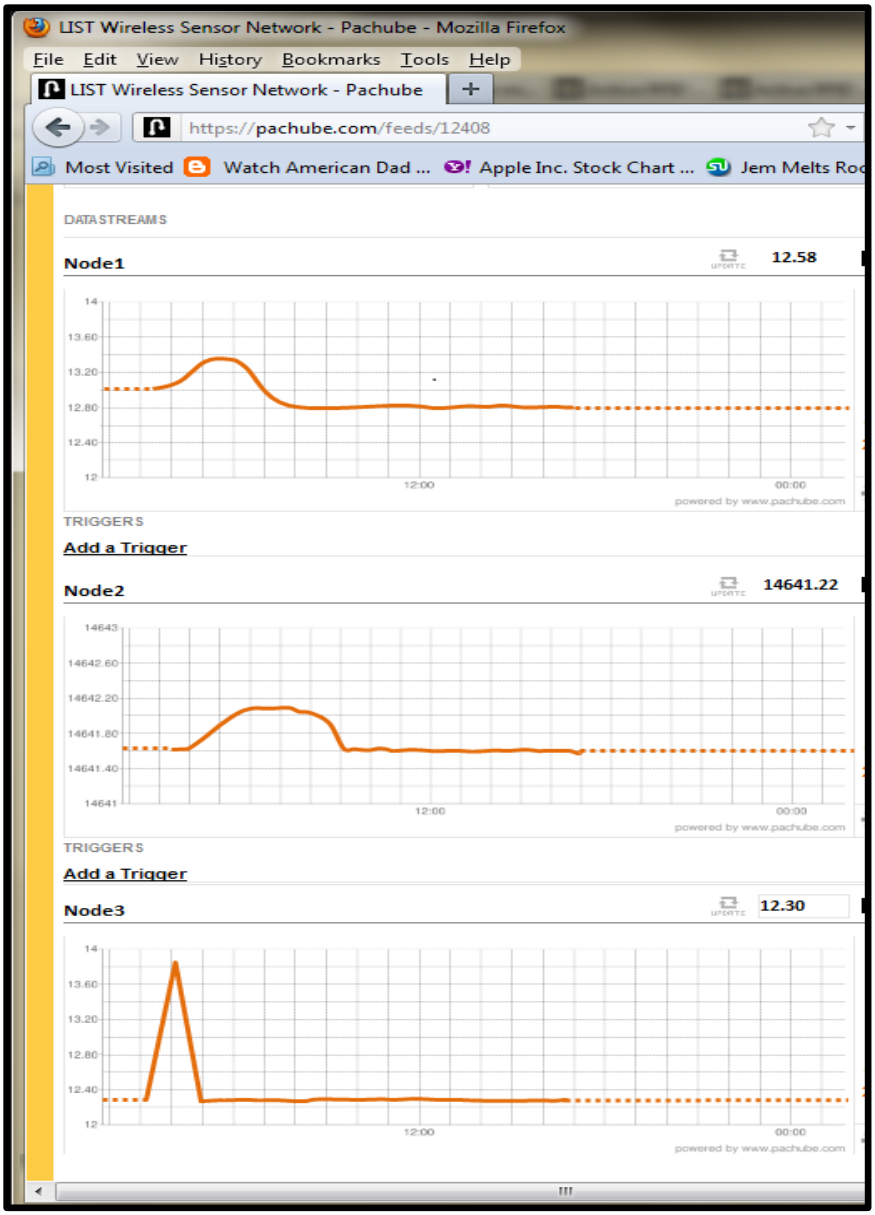


Figure 6-12 Sensor data hosted in pachube.com

This figure shows the pachube website graphing the three nodes of our sensor network. As we can see on the web all our tanks were filled up with different ratios. The first node started at half volume then filled up unto maximum capacity and then emptied at a similar flux rate. For the second node the tank was empty as an initial condition. We filled up the tank and waited for a

couple of minutes and the emptied the tank to a similar rate. For the last node we filled up the tank to full capacity in a fast pace and then empty it also at a similar rate. The data being projected on the web were reflecting the real data on the website. This experiment performed reasonably as expected, and obtaining the same behavior demonstrated the capabilities of the implemented algorithm and the prototype liquid level capability as a functional system.

6.4 ARDUINO NETWORK PROTOTYPE

Similar to last experiment we can host given sensor data to pachube using a microcontroller Arduino and XBee nodes. This experiment was carried out by using 3 end nodes with the Arduino coordinator station. The idea is to set up the Arduino coordinator node with the end nodes host the gathered data on the web and prove it reasonably functional. Then the Arduino interacts with the Wi-Fi-Shield in order to communicate through Pachube and push data online.

The same liquid level of each node was calculated several times by the central server using different sets of data. Additionally the liquid level was changed several times in order to verify sensor accuracy and linearity.

As shown in 4.2 this remote node is also the one also used for this configuration. It is a simple configuration with minimal interconnection and hardware requirements.

The WiShield algorithm [21] uses a basic protocol to connect to the internet while the Arduino pushes the data through pachube.

```
#include <WiServer.h>
#define WIRELESS_MODE_INFRA   1
#define WIRELESS_MODE_ADHOC   2

// Wireless configuration parameters ----------------------
----------------
unsigned char local_ip[] = {192,168,1,2};   // IP address of
WiShield
unsigned char gateway_ip[] = {192,168,1,1};   // router or
gateway IP address
unsigned char subnet_mask[] = {255,255,255,0};   // subnet
mask for the local network
const prog_char ssid[] PROGMEM = {"PASSWORD HERE"};      //
max 32 bytes
```

This algorithm is used to register the WiShield into the local Wi-Fi. In order to use the XBee with the Arduino we have to use another XBee RXTX Libraries called XBee-Arduino which has the API capabilities. In order to push data into pachube we must provide our own Pachube Key. The Arduino code must contain this key in order to register the website.

```
IP Address for Pachube.com
uint8 ip[] = {209,40,205,190};
char hostName[] = "www.pachube.com\nX-PachubeApiKey:
YOUR_PACHUBE_API_KEY\nConnection: close";
char url[] = "/api/2556.csv?_method=put";
```

When Pachube acknowledges the WiShield the Arduino pushes the data in order for pachube to graph the sensor values.

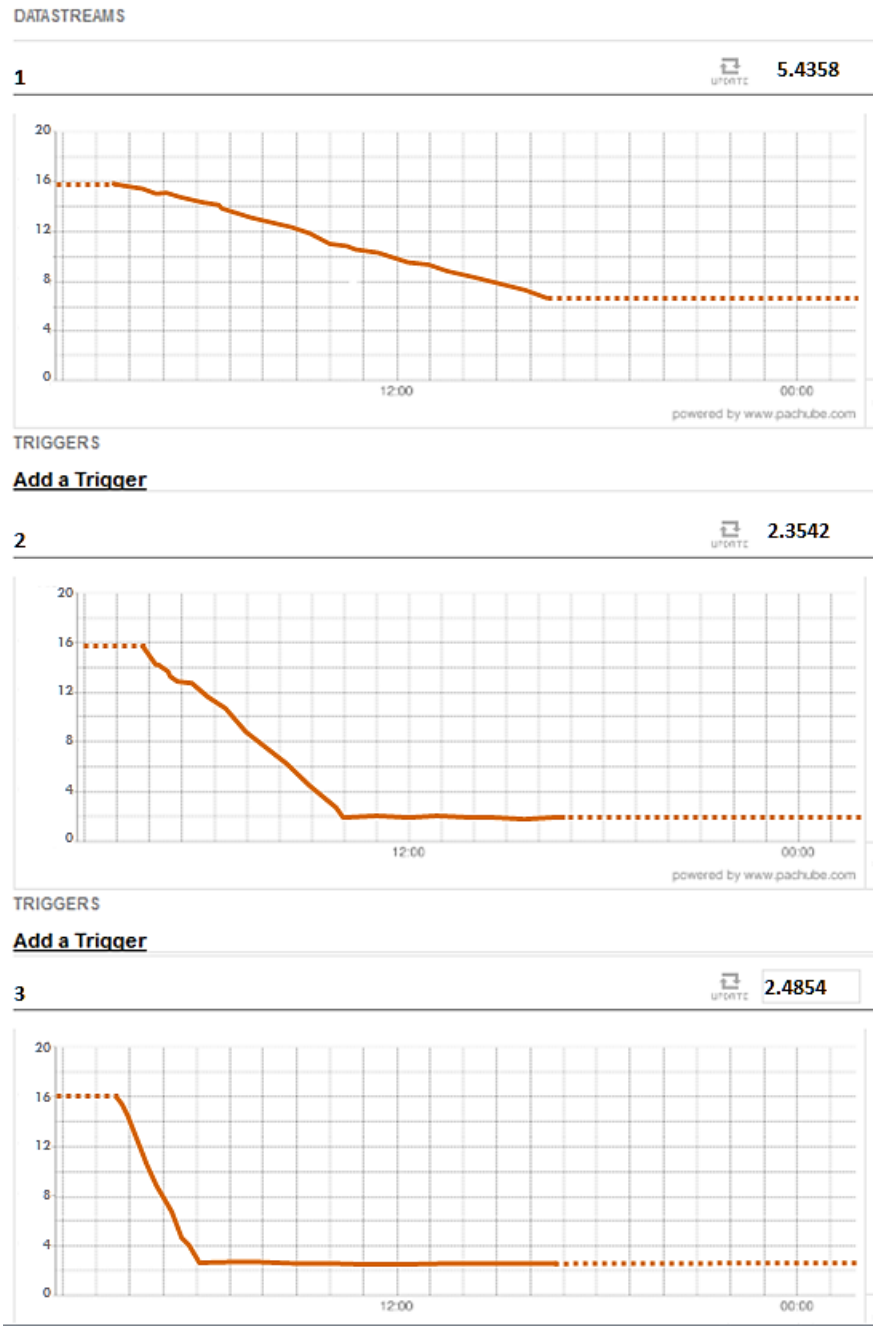


Figure 6-13 Pachube Sensor Data (Arduino/WiShield Setup)

This figure shows the pachube website graphing the nodes of our sensor network. This figure shows us the water discharge at different rates of three tanks setup. All analog values were

verified against the tank visual gage. The data being projected on the web were reflecting the real data on the voltmeter. This experiment performed reasonably as expected, and obtaining the same behavior demonstrated the capabilities of the implemented algorithm and the prototype liquid level capability as a functional system.

CHAPTER 7

CONCLUSIONS & FUTURE WORKS

7.1 CONCLUSIONS

A basic, low cost interface structure for liquid level sensing based on wireless mesh networks has been developed. The hardware infrastructure was built upon the Maxstream XBee RF module. This system is easy to deploy, and any type of analog sensor can be added to the system; this opens the opportunity to numerous kinds of applications from agricultural applications to home and industrial, monitoring different physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants.

A data management algorithm was developed and demonstrated as functional. The study demonstrated that the wireless sensor network can be an effective tool for liquid level monitoring. We presented a working prototype implementation of the systems as well as experiments that demonstrated the system's capabilities and functionality.

7.2 SUMMARY OF CONTRIBUTIONS

As a major contribution we built a prototype of real time liquid level monitoring system for 3 water tanks based on wireless mesh networks using off-the-shelf technology. Developed software to display the acquired liquid level data from three wireless end nodes. Designed and

implemented a compact printed circuit board for the system end node. Implemented a graphing algorithm to parse sensor data online. Conducted experiment with our prototype demonstrating that the system works efficiently

In chapter 2 we gave an introduction of the related technologies to liquid level sensing and wireless sensor network technologies. Most of the sensor technologies are not compact and has more power consumption. When talking about the discussed wireless modules in Figure 2-11 Timeline for wireless nodes platform [2] most of them are bulk big and more costly compared to the XBee RF module. This chapter discussed the elements that compose a wireless network are explained to obtain a view of the new systems nowadays. Discussed the physical principles of liquid level sensing for a fluid column and a closed container. A technological review on our wireless module and its protocol are presented including the wireless module board used in initial experiments. Described the technology used as central server including wireless module and its specifications. The wireless module XBee can behave as a "wireless cable", even though this module is used in API mode.

We presented all the parameters and a basic knowledge of how do they operate in our setup.

- *XBee Wireless Module* – It is a stand-alone, ready-to-use solution for low-power, low-cost, wireless sensor networks. They operate within the ZigBee mesh networking protocol at the network level or MAC level.
- *ATBD*- It is the XBee AT command to change the baud rate. Number of pulses per second, also known as modulation rate.
- *ATID*- It is the XBee AT command to change the Personal Area ID (PANID).

- *ATMY*- It is the XBee AT command to change the address number.
- *ATDL*- It is the XBee AT command to change the destination low address.

In Chapter 3 we described wireless network topologies, multi-hop and multipoint-to-point. Multi-Hop Cluster Network architecture has the advantage to analyze a cluster of information, like a container array, and Multi-Hop Multi-Point to Point Network is designed to reach larger areas, like larger water containers. And reviewed of what does wireless sensor network consists off.

- *Wireless Nodes* – The nodes have several tasks to do besides creating and maintaining the wireless mesh network. Each node receives and performs commands sent by the central server (computer or microcontroller); also return analog values to the central server.
- *Central Server* – The central server software is in charge of managing the coordinator node, configuring the wireless nodes in the network, and calculating digital values in order to graph it. Digital information is used to draw the liquid level in the graphic unit interface.

In Chapter 4 we described the hardware interface proposed for the experiments. Described the Wireless Network System and all interacting and interdependent components forming the core system. Starting with the sensing unit, a MPX2010 silicon piezoresistive pressure sensor to the XBee wireless RF module, which gives our system the necessary properties to interact with the working body. An amplifier circuit was implemented to magnify the voltage differential to a higher level in order to couple with the XBee radio transceiver. The XBee obtained sensor data

through an I/O. The analog data obtained from the sensor was then traduced inside the XBee with and analog to digital converter (ADC). This ADC is an XBee internal device that converts a continuous analog quantity of signal into a discrete time digital representation of itself. This provided insulated measurement is then broadcasted as a radio wave out of the XBee to the coordinator node. For both our prototype systems our end node maintains the same hardware.

We described the full development of a printed circuit board designed for the system's end node. A toner transfer method was used in order to etch a copper template. All material for the development was disclosed as well as the processes made for the full manufacture of this custom PCB. Issues concerning printer dpi images were discussed, and presented the PCB schematics for two different boards. The materials in order to populate our designed board were disclosed. Advantages of applying a soldering mask to our circuit were discussed; one is that prevents solder from bridging between conductors, thereby preventing future short circuits.

We discuss about our wireless communication unit which is developed based on XBee RF modules, engineered to operate within IEE 802.15.4 standards. The XBee RF wireless module was used as a stand-alone, ready-to-use commercial device which benefitted us with low-power, low-cost, wireless sensor networks.

Estimated the power consumption for the end node for a battery of 5,000 mAh. Based on these numbers we compute their power consumption as 60 mA, assuming led is not constantly on. At full power the node consumes 60 mA, using 2 AA batteries of 5,000 mAh in about 83.333 hours almost 3.5 days. But this is not the case since most of the time the nodes are at sleep mode. If we

set a 10 second time interval between each server broadcast, we get 7936.5 hours of operation for a node which translates to about 330.5 days of operation.

Chapter 5 presented the results of experiments conducted on a working implementation of our proposed software middleware for real time liquid level system. In order to apply a standard practice over the pressure transducer on the operational environment we defined the behavior or this sensor through a gamma of pressure, and obtained both characterization equations to simulate and design a digital to liquid level translation of all wireless nodes for our main algorithm. The experiments help validate our solutions and ideas, as well as serve as a demonstration of the system. Discussed the setup of the working network of wireless nodes and a central server, and prove it reasonably functional.

Presented the hardware configuration necessary for the XBee radio, pairing and operational parameters. These parameters are configured in XCTU, a XBee configuration software. Each node were assigned with its given configuration a given baud rate (BD), set, same ID and enabling API. Also presented the computer data management program; it was developed in Processing language, a language built on the Java programming language. The Arduino microcontroller has his own programming environment, based on Processing language. The Arduino and PC programs read the real time data strings from coordinator nodes, convert them to numerical values and present the data in two different ways, numerical and graphical. The computer and microcontroller program sends the information to a host website that can graph sensor value in a web page. This web page can be accessed from any part of the world in most of the commonly available web browsers. Both programs filter each end node device in order to

obtain given digital value from the pressure sensor. The computer program presents a graphic unit interface for three XBee radio transceivers. This includes the primary information of deployed nodes, such as ID, received signal strength (RSS) and liquid level. Also presents a visual representation and numerical values of sensed data in a user friendly environment. The Arduino microcontroller parse the data from the end devices filters it and host it online.

In Chapter 6, we presented a working implementation of our proposed software middleware for real time liquid level monitoring systems on wireless mesh networks for both coordinator server station, PC and Arduino.

We executed a single node prototype experiment go further and validate the software middleware as a functional system. When confirmed a single end node system we developed a system that can put in practice 3 end nodes. With this experiment we obtained the liquid level sensed value for three end nodes, test the software middleware, discuss all the process in order to be able to host this gathered data online and prove the prototype as a functional system.

The experiments helped validate our solutions and ideas, as well as serve as a demonstration of the system. We calculated sensor values several times in order to validate the system, and prove it reasonably functional. The experiments helped us verify the software middleware as a functional system, demonstrating the prototype liquid level monitoring capabilities. We concluded that based on the findings, an increase in nodes will not result on increases in data noise or differences in data analytics. We also demonstrated the capabilities of both prototypes to host data online through a web host Pachube using two different setups, central server as a

computer and as a microcontroller. Our work demonstrated the feasibility of real time liquid level monitoring with of-the-shelf components.

7.3 FUTURE WORK

This section gives directions and suggestions for future developments concerning our work.

- *Wireless Nodes* – Wireless nodes firmware could be greatly improved on several areas regarding power management, temperature sensing in harsh locations, safety systems improvements, level monitoring for other liquids, River dams improvements, chemical to water supplies, home automation, liquid transportations (chemical or hazardous), sensorial help for blind people, parallel systems sensing and return, extension on XBee module management, and the ability of remote I/O manipulation. This last one is an interesting and useful topic since it enables a remote manipulation of the microcontroller's internal modules. It adds a variety of capabilities to the system.
- *Wireless Nodes Hardware* – An elegant prototype package with minimal size, antenna protection, coin battery, weather protection, and external ports.
- *Central Server* – The central server should implement a generic communication interface to enable Multilanguage programming. Improvements should be made to data filters and calculation speeds.

REFERENCES

[1]. Digi International, Inc. (Sep 2009). : Product Manual v1.xEx -802.15.4 Protocol. *XBee/XBee Pro OEM RF Modules*. 1 (1), p1-69.

[2]. Ian F. Akyildiz and Mehemet Can Vuran. *Wireless Sensor Networks,* JohnWiley & Sons ltd 2011.(49)

[3]. Thomas P. Gootee. (2007). *Making PCBs.* Available: http://fullnet.com/~tomg/gooteepc.htm. Last accessed 1st Oct 2011.

[4]. Harrop and Raghu Das. . (Jun 2011). Active RFID and Sensor Networks 2011-2021: Rapidly growing sector. *IDTechEx Ltd, Cambridge, MA, 2011*. 1 (1),

[5]. Application of ZigBee/IEEE 802.15.4 Protocol Compatible RF Module XBee/XBee Pro; *WANG Jingxia(Shenzhen Polytechnic,Shenzhen* 518055,China, CNKI:ISSN:1006-7787.0.2007-03-008

[6]. Vongsagon Boonsawat, Jurarat Ekchamanonta, Kulwadee Bumrungkhet, and Somsak Kittipiyakul. (2010). Xbee Wireless Sensor Networks for temperature monitoring. *Thammasat University, Pathum-Thani, Thailand* . 1 (1), p1-6.

[7]. Arduino Community. (2005). *Arduino information and projects.* Available: http://www.arduino.cc/. Last accessed 10th Sept 2011.

[8]. Digi International Inc, XBee-PRO OEM RF Modules, Product Manual v1.x.4x - Digi Interna-tional Inc.11001 Bren Road East Minnetonka, MN 55343877912-3444 or 952 912-3444 http://www.digi.com

[9]. Vachirapol Mayalarp, Narisorn Limpaswadpaisarn, Thanachai Poombansao, and Somsak Kittipiyakul. (2010). Wireless Mesh Networking with Xbee. *Sirindhorn International Institute of Technology, Thammasat University, Pathumthani, Thailand*. 01 (1), p1-5.

[10]. Fry and Reas. (2001). *Processing.* Available: http://processing.org. Last accessed 1st Oct 2011.

[11]. Jonathan Oxer and Hugh Bleemings (2009). *Practical Arduino Cool Projects for Open Source Hardware*. Berkeley, CA 94705: Technology in Action, Apress . p2011.

[12]. Direct Industry, *Pulsar IMP Ultrasonic Level Sensor,* Available: http://www.directindustry.com/prod/pulsar-process-measurement/ultrasonic-level-sensors-25501-57561.html. Last accessed 23th Mar 2012

[13]. C3 Energy3 Components, *Continuous Output Level Transducer, Available:* http://www.cynergy3.com/productdetail.aspx?id=36, Last accessed 23th Mar 2012

[14]. Pyroton India Inc., *Magnetic Float Level Transmitter,* Available: http://www.indiamart.com/pyrotron-india/level-control.html#magnetic-float-level-transmitter, Last accessed 23th Mar 2012

[15]. Direct Industry and Eletrotec, *Compact optical liquid level switch,* Available: http://www.directindustry.com/prod/elettrotec/compact-optical-liquid-level-switches-5837-462220.html, Last accessed 23th Mar 2012

[16]. Tom Igoe. (2008). *XBee Library.* Available: http://www.tigoe.net/pcomp/blog/archives/projects/index.shtml. Last accessed 15th Apr 2010

[17]. IEEE Professional Association. (2005). *IEE 802.15.4 Standards.* Available: http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf. Last accessed 20th Jul 2011.

[18]. OMRON technical support (2007-20012), *Technical Guide on Pressure Sensor, Available:*h*ttp://www.ia.omron.com/data_pdf/guide/56/pressure_tg_e_2_1_1(principles). pdf* Last accessed 20th Mar 2012.

[19]. Fry and Reas. (2001). *Processing Wiki.* Available: http://wiki.processing.org/w/Main_Page. Last accessed 16th Oct 2011.

[20]. Robert Faludi (2011). *Building Wireless Sensor Networks*. Gravenstein Highway North, Sebastopool CA 95472: O'Reilly Media, Inc. p267.

[21]. Greg Eigsti. (2009). *Pachube and WiShield.* Available: http://asynclabs.com/forums/viewtopic.php?f=15&t=86&start=10. Last accessed 10th Oct 2011.

[22]. Lee, Kun-Hong**.** RFID sensor and Ubiquitous Sensor Network system Thereof. US, Sep 20, 2007. US 2007/0216531 A1.

[23]. Norsheila Fisal, Ahmad Sharmi Abdullah, Muhammad Yusof Mohd Noor (2008). *Development of WSN and Optical Applications*. Perpustakan Negara Malaysia: UTM.

[24]. Gloria F. Serrano Moya, Jose L. Zamorano Flores. (2010). Wireless Sensor Network for Monitoring Temperature. *International Congressnon Instrumentation and applied sciences*. 1 (1), p1-7.

[25]. Chen Yanshi, Jennifer Huang. (2009). WIRELESS SENSOR NETWORK IN TRANSPORT INDUSTRY. *SIM UNIVERSITY*. 1 (1).

[26]. Peter Fyon. (2010). Zzigbee Mesh Network. *Department of System and Computer Engineering*. 1 (1),

[27]. 10kPa On-Chip Temperature Compensated and Calibrated Silicon Pressure Sensors, MPX2010 Series, Freescale Semiconductor, Inc, 2005-2008,(4)

APPENDIX

1.1 MULTIPLE NODE ALGORITHM

As stated in chapter 5, we are using the difference in pressure to solve our water level monitoring problems. This algorithm can be used for one, two or three end nodes and can be fixed for more nodes as necessary. We are using the pressure formulas in order to obtain the height of water, also using a sensor characterization. Our systems gather analog data then converts this data to digital, and with the proper characterization, it translates to distance information useful to the graphing algorithm. This algorithm uses XBee RXTX Libraries; there are multiple libraries for Xbee online. The libraries we are using are not so popular but let us use a combination of Tom Ignore graphing algorithm with the API capabilities we are searching for, in order to construct this middleware. It works by relating this analog valued with the actual pressure on the tank.

In order to implement the algorithm there is a set of events we will have to do. Our first piece of algorithm here is the initial setup. But first we state the commands that will be importing the XBee and serial libraries respectively.

```
import xbee.*;
import processing.serial.*;
```

Then we state the needed objects. “Serial port” to call the exact serial port we are using on Xbee, and the XBee reader “XBeeReader” wich is a constructor that takes the parent PApplet “this” and a reference to a made serial port

```
Serial port;
```

```
XBeeReader xbee;
```

We can start delimiting a font for the text wanted to be displayed. “tipodefont” is a variable that will be used in the future on the algorithm when setting the font type to be used.

```
PFont tipodefont;
int fontSize = 14
```

We now can start the main construction of the program. With “void setup() {” we can now start writing the command line arguments.

```
void setup() {
size(600, 430);          window size
frameRate(60);          setting the frame rate
smooth();                clean the jagged edges
```

First we will need to find the serial port in use by the coordinator in the computer. To do this we can ask for a serial list with “println(Serial.list());”.

Serial ports list to find ours:

```
println("Available serial ports:");
println(Serial.list());
```

Once we know the port being used by the coordinator, we use this equation to give a value to the variable “port”.

```
port = new Serial(this, Serial.list()[2], 115200);
```

Initializing the xbee object.

```
xbee = new XBeeReader(this,port);
xbee.startXBee();
```

1.2 GRAPHING ALGORITHM

This event in the algorithm specialized in drawing a graph out of the analog values of the XBees. To initialize this event it is use public void draw, and to filter the node that is going to use this loop it is used a “if” with the intentions of graph the node with the address equal to 1.

```
public void draw() {
if (address == 1){
```

In order to draw just when we got a new reading we use another “if”, this way we assure we are graphing new data. If we want a continuous graph we would like to create a condition that only draw when you have a new reading on the node.

```
if (newData == true) {
image(bottom1,132,286);
```

```
image(valuespic1,7,91);
```

If we got a new data now we are going to iterate over the number of sensors:

```
for (int thisSensor = 0; thisSensor < numSensors; thisSensor++) {
```

If you have new data and it's valid (>0), then we will graph it:

```
if (analog[thisSensor] > -1) {
```

Now we need graph the values of the sensors values on the screen, basically plotting continuous line segments. Map the sensor values to the screen size for graphing:

```
y = int(map(analog[thisSensor], 0, 1023, 228, height-35));
oy = int(map(previousValue[thisSensor], 0, 1023, 228, height-35));
```

If by any chance we get a big change, increment the hit counter:

```
if (abs(y - oy) >= hitThreshold) {
hits++;
```

In order to note the magnitude of the hit:

```
lastHitValue = abs(y- oy);
}
}
```

Set up a different the graphing color for each axis:

```
switch (thisSensor) {
case 0:
stroke(0,0,0,0);
break;
case 1:
stroke(0,255,0);
break;
case 2:
stroke(0,0,0,0);
break;
}
```

Draw the graph line from last value to current; this gives the illusion like we would be doing a graph instead of a bunch of lines.

```
line( xpos, oldYPos/1.5, xpos+1,yPos/1.5);
newData = false;
```

Also this is part of the graphic unit interface we write some informational text at the top of the program. This text includes the Xbee address the dBm and the output from the Xbee.

```
noStroke();
fill(16,78,139);
```

```
text("Adress:0 " + address, 172, 360);
text ("RSSI: " + rssi + " dBm", 172, 380);
//VoltageA = analog[1];
//VoltageA = analog[1]*.5*3.3/951;
VoltageA = 3.3-analog[1]*.00323;
fill(0,255,0);
text(": " + VoltageA , 20, 145);  // for input 3 ad "X: " + analog[0] +
//text("Last hit value: " + lastHitValue, 10, 100);
}
```

When the graph ends at the right of the screen we would like to start all over again at the beginning. In order to do that we clear all graph and start back in the left again.

```
if (xpos >= width) {
xpos = 162;
image(template,0,0);
}
else {
xpos++;
}
}
}
```

1.3 FILTER ALGORITHM

This filter is the heart of the code. This filter function works like a “serialEvent()" and is called when data is available to be read from your XBee radios. What we do basically on this algorithm is hearing all Xbee signals in the air and assigns a name that the algorithm can understand in order to work properly.

We start opening the event with:

```
public void xBeeEvent(XBeeReader xbee) {
```

Grab a frame of data to analyze

```
XBeeDataFrame data = xbee.getXBeeReading();
```

This version of the library only works with IOPackets

```
if (data.getApiID() == XBeeDataFrame.SERIES1_IOPACKET) {
```

Get the transmitter address

```
address = data.getAddress16();
```

Because we now the Xbee address we can now filter by addresses all data and will be a lot easier to manage on the previous algorithm.

```
if (address == 1) {
```

Get the RSSI reading in dBM

```
rssi = data.getRSSI();
```

Save previous state of analog values:

```
arraycopy(analog, previousValue);
```

Get the current values, (-1 indicates channel is not configured):

```
analog = data.getAnalog();
newData = true;
}
```

The same is done for the other xbees, whose address will be equal to 2

```
else if (address == 2) {
  rssiX2 = data.getRSSI();
  arraycopy(analogX2, previousValueX2);
  analogX2 = data.getAnalog(0);
}
```

Now trip the new data flag so the draw() loop will graph:

```
newData = true;
}
else {
println("Not I/O data: " + data.getApiID());
}
}
```

1.4 USER INTERFACE ALGORITHMS

The user interface algorithm is one of the most important algorithms

We can start delimiting a font for the text wanted to be displayed. “tipodefont” is a variable that will be used in the future on the algorithm when setting the font type to be used.

```
PFont tipodefont;
int fontSize = 14
```

Then we can declare the names of the images we would like to use in order to display them.

```
PImage template;
PImage valuespic1;
PImage bottom1;
PImage valuespic2;
PImage bottom2;
PImage valuespic3;
PImage bottom3;
```

We can use this command in order to bring an array of fonts, available to the system, with a number. The number “20” indicates the font to use in the list.

```
tipodefont = createFont(PFont.list()[20], fontSize);
textFont(tipodefont);
```

After initializing the Xbee reader we can upload the pictures and reference it with the declared names above.

```
template = loadImage ("template.jpg");
valuespic1 = loadImage ("valuespic1.jpg");
bottom1 = loadImage ("bottom1.jpg");
valuespic2 = loadImage ("valuespic2.jpg");
bottom2 = loadImage ("bottom2.jpg");
valuespic3 = loadImage ("valuespic3.jpg");
bottom3 = loadImage ("bottom3.jpg");
```

Then we can call these images whenever we like trough the algorithm. Also it is very important not to use the same color for all three graphs, in order to give a better visual representation of each graph and whose Xbee does it corresponds we will use a different the graphing color for each axis:

```
switch (thisSensor) {
case 0:
stroke(0,0,0,0);
break;
case 1:
stroke(0,255,0);
break;
```

```
case 2:
stroke(0,0,0,0);
break;
}
```

Also displaying the names of each Xbee and its corresponding sensor value, address and signal strength is good to know in order to know if the Xbee is working properly.

```
fill(16,78,139);
text("Adress:0 " + address, 330, 360);
text ("RSSI: " + rssiX2 + " dBm", 330, 380);
VoltageA = 3.3-analogX2[1]*.00323;
```