TECHNOLOGICAL EDUCATION INSTITUTE OF CRETE
SCHOOL OF ENGINEERING
DEPARTMENT OF INFORMATICS ENGINEERING

POSTGRADUATE PROGRAM OF STUDIES
«INFORMATICS & MULTIMEDIA»

Title of Master Thesis

DEVELOPMENT OF PLATFORM WITH USE OF WSN (WIRELESS SENSOR NETWORKS) TECHNOLOGY FOR COLLECTING AND PROCESSING ENVIRONMENTAL DATA, SURVEILANCE OF SPACE AND DIRECT AWARENESS UNDER EXTREME CONDITIONS

PANTELIS M. FASOULAKIS

MARCH 2015

Supervisor: Associate Professor Vlisidis Andreas

Examiners: Associate Prof. Vlisidis Andreas

Assistant Prof. Panagiotakis Spiros

Assistant Prof. Stratakis Dimitris

Presentation Date: 12/03/2015

Acknowledgements

First of all I would like to thank Prof. Andreas Vlisidis, for the opportunity given to me to deal with a very interesting thesis under his supervision. Also, I want to thank the collaborator of the MTMD Lab, Michalis Fragiadakis for his assistance in relation to internet technologies used in the implementation of the diploma thesis.

Abstract

Wireless sensor networks are recognized as one of the most important technologies of this century and can represent a real and workable solution for surveillance of space i.e. precision agriculture, road safety, military applications, forest fire detection, medical applications, smart building applications, e.t.c.

In this article we will present the general theory behind the sensors and sensor networks, some word for TinyOS – the operating system of a WSN and will analyze the communication protocol of our WSN, XMesh which we used in this work , based largely on standards 802.15.4 and Zigbee. Also, we refer some basic features of the hardware which we used in our application such as Micaz module, MTS400 sensor board and Mib520 programming interface board.

Finally, in chapter 7 we present the our WSN Application which is the creation of a network which will consist of small sensor nodes, providing for the surveillance of a space (i.e. the prediction and detection of fire) in real time. The sensors will take and will offer measurements of various environmental parameters, such as temperature, humidity, etc. at regular intervals. Then the data will be forwarded by the network to a central station and will be stored in a central database. The user using a Web application, which we are called to implement, will have access to current and older data.

Table of Contents

Chapter 1
INTRODUCTION - MOTIVATION

Wireless sensor networks have already started many years to be used for a wide range of applications including usually functions such as detection of events (event - driven) or periodic measurements shots (time - driven). Some of the scientific and industrial fields where it is appropriate to use WSN are:

- Environmental control and recording biodiversity
- Home automation
- Precision agriculture
- Pharmaceutical and Healthcare
- Telematics
- Industrial automation
- Robotics

In the category of environmental control falls the application to design and based on this to program the hardware in the following chapters.

The application is intended for registration of environmental conditions in woodland for two purposes. The first is the collection data for the scientific study of the microclimate of the area. Microclimate is the climate condition of a relatively limited area (a few square meters to thousands of acres) which different from those that surround it. The microclimate conditions depend on many factors in the most basic humidity and temperature. The microclimate in woodlands plays leading role in the development of flora and fauna which in turn interact with it.

The second objective is the risk of fire detection and early warning measures. Experience has shown that the critical time for the spread of a fire are the first 15 minutes.

It follows that the design of WSN based on two main axes. Periodically taking measurements and storing in the database and identify events that need immediate treatment. In the first case the delay sending data is not critical, so the network may use practices to increase the life time in weight of immediacy. The second is obvious that the objective is send the data at all costs.

Structure of Work

The second chapter is a general introduction to WSN, where describes the features, requirements and the various mechanisms used for their satisfaction. It becomes

reference to the architecture of the units and in several protocols and routing algorithms that determine the network architecture.

The fourth chapter summarizes the standards 802.15.4 and Zigbee, defining a specific protocol stack for WSN and their use is widespread. It becomes reference to many items of standards that used by the hardware and software of the application.

In the third chapter we deal with the software, and more specifically with the operating system for embedded TinyOS systems, and in the fifth chapter with the XMesh software that implements the protocol stack of network.

The hardware of the Crossbow-Memsic company (wireless nodes, sensors) that we used to implement the present application, is presented in Chapter 6.

The seventh chapter presents the design and the planning of our application using all previous tools and simulates the operation of network of 5 nodes in order to draw conclusions.

In the last chapter we refer some interesting works which provide smart solutions to support wireless sensor network environmental monitoring applications.

Chapter 2
Wireless Sensor Networks

Introduction

In recent years, the desire for connectivity has caused exponential growth in the use of wireless communications, which combined with the progress in the mechanisms of transmission (RF systems), evolution in low-power circuits (VLSI) and embedded systems led to the implementation of its small size, energy-autonomous, self organized nodes (smart sensors), low power and cost, able to observe and measure changes in natural environmental phenomena. At the same time, these nodes are able of wireless communication and transfer of data to a central node via the available integrated radio transceiver. Thus, we have the Wireless Sensor Networks or WSN consisting of a large number of such nodes, fitted to the observation area. These nodes have the ability to sense (sensing), trigger (actuation), and processing of data collected by the microprocessor which have within them, and they can spend most of their life time in idle state and are automatically activated when an event occurs.

Wireless sensor networks are relatively a new technology that seems to be used more and more in the future because of potential that have to collect and process data. The fact is that the standards and security protocols used in the wired and other networks do not match in wireless sensor networks, mainly due to the strictly limited resources that are available.

During the collection of data from the nodes of the network plays a key role in the energy consumption of each node, which determines the lifetime of the network. To increase the lifetime of a wireless sensor network, various techniques have been developed which are designed to minimize energy consumption at the nodes. Such techniques is the packet data aggregation in order to reduce the overall load on the network, and the use of routing protocols taking into consideration the reduction of total energy consumption in a network. [2]

The basic properties of WSN are summarized in the following points:

- Provide reliable control and take precedence over the corresponding conventional sensor systems
- Have ability to self-organization
- Provide limited-range communication and multi-hop routing (multiple jumps)
- Have a high sampling frequency and high resolution due to the potential of measuring by many nodes
- Dense placement of nodes and collaborative effort
- Enable remote control, as it does not require the physical presence of the user for their operation

•Frequently changing topology due to attenuation of the signal and failure of nodes

•Provide greater fault tolerance because of their dense layout

•Limitations in energy, transmit power, memory and computational capability

Differentiation in relation to traditional networks

Wireless sensor networks differ from traditional ad-hoc networks and can not be applied to these methods and protocols that have been developed for conventional networks. Although all WSN are networks consisting essentially of tiny computers, differ from common networks for the following reasons:

- They present significant limitations in computing power, energy, storage and bandwidth. In traditional wireless networks, the functions of routing and management of mobility of nodes are carried out to optimize the quality of service (QoS) and efficiency of bandwidth. The energy consumption goes into the background, as the energy source can be replaced or recharged at any time. Instead, the nodes forming the WSN networks are designed to operate without the presence of the user, so functions such as routing and energy consumption play a vital role in network design.

- The WSN are closer at the distributed systems despite the typical networks, where users are connected to a node (or a set of nodes) and require a service from another node. The nodes of the WSN work together to produce results, exchanging information with each other, while the user is rarely interested in the results of individual nodes. Consequently, the network provides no connection between different sections but information services to users.

- Usually, the nodes of a WSN are static after placing, with the exception of a small number of potential moving nodes.

- The nodes are prone to failures, mainly on hardware level and this causes the change of topology most often in relation to common networks.

- In the WSN, the nodes are many more and send data with much more low rates, which for the purposes of such networks is sufficient (typical 250 Kbps)

Applications

Nowadays there is a lot of movement around the applications of wireless sensor networks, because of the benefits they provide and the characteristics that make them suitable for use in all and most areas where conventional networks cannot meet. Some of these applications are: [2]

- **Environmental applications**

There are various applications associated with the environment and the type of used sensor is differentiated in accordance on the application. There are sensors for applications meteorological research, for study of pollution, rainfall sensors, water level sensors and for measuring physical parameters such as temperature, atmospheric pressure, humidity and others. Still, there are types of sensors to observe and record the animal kingdom and the movement of birds during periods of migration. Another application relates to the recording of critical parameters and environmental conditions that affect the earth's climate. Finally, an important environmental application is the fire detection, particularly for countries that have a significant problem with the fires as the Greece. Due to the fact that the WSN nodes can span to a wide range and cover large areas that are inaccessible for humans but also for the means it uses to fire fighting, make ideal for prevention and immediate notification of the competent authorities.

- **Smart buildings and facilities management**

The aim of using WSN in buildings and facilities is to reduce waste energy by controlling the conditions inside buildings as regards humidity, ventilation and air conditioning. Measurements are performed by covering room, controlling the flow of air, temperature and other physical parameters. This is achieved not only saving energy but also improves the standard of living. It is also possible to control the mechanical pressure levels in seismically active zones, verifying if the building is safe or is on the verge of collapse.
Another important application of WSN is the installation of sensors on bridges for the measurement of vibrations caused by the movement of vehicles either by an earthquake, in order to promote the safety of the drivers. Also, the installation in large buildings and facilities for monitoring any unwanted instances in some space, i.e. non-authorized intrusion to a place but even incorporation of sensors deep inside on machines where the wired sensors would not be installed. This enables the easy machine monitoring and preventative maintenance.

- **Medical applications**

The use of wireless sensor enables remote, home monitoring in cases of chronic diseases or elderly. Devices can be moved or even worn could record important functions of the patient in case necessary to notify the relatives, the doctor or the hospital. The idea of integrating wireless biomedical sensors in the human body is quite promising, although there are many additional difficulties such as system security, reliability, minimize maintenance, reducing consumption and the exploitation of human energy (heat) to provide power to the sensors.

- **Industrial applications**

In industries, the control of systems and applications throughout their function plays an important role for supervising but also for the safety of personnel working in them. The environment in which the processes are carried out can be dangerous for the health of the personnel, due to the high temperatures or due to toxicity from the existence of harmful gases or may not even be accessible to humans. In such cases it is necessary to use wireless sensors and the main services they provide is the automation of processes, such as maintenance of machines through making information on their situation and take immediate decisions in cases of errors. Remote control service provided by WSN coupled with the existence of electronic programs, providing full management and control of demanding applications. One such example of industrial application are oil refineries, where wireless sensors measure the temperature at all stages of the processing of oil and when the allowed limit is exceeded, especially alarms inform the experts.

WSN Architecture

Wireless sensor networks can consist of a large number of individual nodes, from a few hundred to thousands, placed in a relatively closely spaced in the region of interest for measuring a particular phenomenon. These nodes, which are small in size and low consumption, can communicate with each other at relatively small distances (10 to 70 meters). They consist of individual sections of sensors and data processing and their main goal is to work together to exchange information about the data collected and the parameters for the network status.

Another feature of WSN is that once placed nodes, forming their own network via algorithms and protocols that run. Also, these protocols are taking advantage of the dense arrangement of nodes in order to convey a message, through the procedure of jumps (hops). This communication through multiple jumps is called multihop communication and thus lower energy consumption is achieved. Also, communication through multihop contributes to better coverage of the region and to address the problem of signal attenuation due to long distance.

In addition, each node incorporates a processor which enables it instead sends the data directly to a designated node that is responsible for processing, using the same

first processor for performing simple defined calculations and then sends only the necessary and partially processed data. [2]

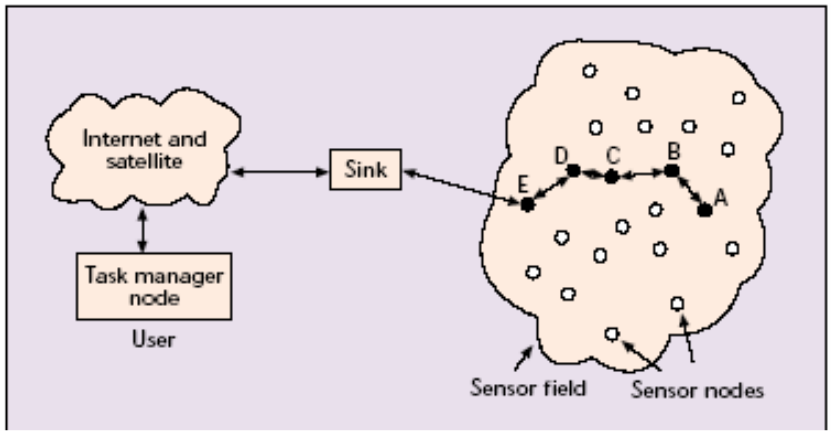


Figure 2-1: Typical form of sensor networks

A structure of a WSN is consisted of 2 types of nodes:

- **The data Source**, which perceives, processes and communicates wirelessly, while interacting with the physical environment in which it is located, collecting measurements of physical parameters. So each node of the network is a source.
- **The data Router**, which transfers the data from a nearby node to another or to the base station node which is become the processing and analysis of collected data from different nodes.

The base station is usually referred to as node-collector (sink node), in the sense that in this land the results of other nodes. There are three options for the location of the node-collector which differentiates partially his function. It may be a node like the others in the network, but can also be a unit offline. In this case, the collector node can be a laptop, which interacts with the network to exploit the data. In the third case, the collector node is an intermediate node between the sensor network and a larger network such as the Internet and called getaway. Usually the base station is more remote from the rest of the network and has greater energy reserves than other nodes, usually connected to a laptop.

The stack of network levels of a wireless sensor network consists of the same levels that are defined in the OSI model but without the session layer and adding three more at the entire stack. These three levels, arrange the proper cooperation of nodes between them and the sharing of resources. These are as follow:

•**Power management** level which manages how a node uses its energy reserves

•**Mobility** level monitors the existence of neighbors in the network at any time, to verify if there are connections available to forward the messages to the base station.

• **Task management** level, manages the workload and determines which nodes are active. For example, the nodes that have more energy take a more active role in the network. The tasks are simple computational processes, characteristic of the operating system of a WSN.

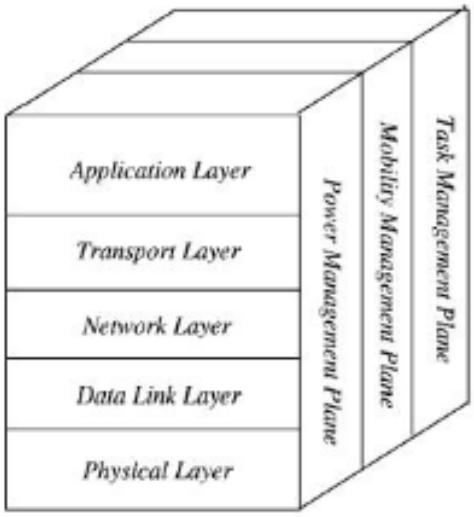


Figure 2-2 : The stack of protocols in to a WSN

Nodes Architecture

The basic building blocks of each wireless node are the processing unit, the sensing unit and the power unit. [2]

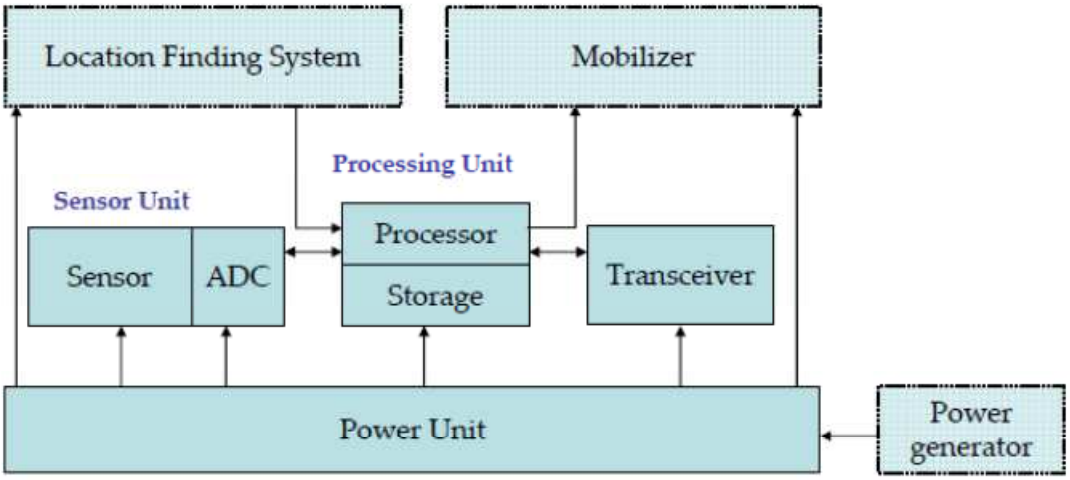


Figure 2-3 : The Architecture of a wireless node

❖ **Processing unit**

The processing unit is the central element of a wireless node and contains the processor and a relatively small storage space. For applications of sensor networks, the role of the processor selected to play a microcontroller, which meets the requirements of low energy consumption, connectivity with large number of peripherals, the requirements in operating voltage and low cost. In addition, a part of the microcontroller can be put in sleep mode and another to operate, resulting in reduced consumption, while characterized by fast wake time. The microcontroller manages the collaboration with other sensors, does the communication with the transmission system and performs defined simple calculations (the tasks). Also, performs basic calculations, signal processing and correlations processes of data taken from different sensors for the creation and delivery of a single packet, which implies a reduction of the transmitted information and therefore reduce energy consumption. Other possible processes are data compression and processes on network security. The role of memory undertakes to play mainly flash memories with typical value 128Kbytes.

❖ **Sensing unit**

The sensing unit includes sensors and converters analog signals to digital - ADCs. Its main function is to sense or measure the physical parameters of the phenomena. The physical parameters could be the pressure, temperature, humidity and others. The analog voltage generated on the sensor and a physical quantity is converted into a digital signal by the ADC and is transferred to the processing unit for further analysis. The characteristics and requirements for sensors comply with the requirements for the rest of the system, such as low power consumption, small size and the potential of autonomous operation and adaptability according to the environment. The meaning of sense refers to terms such as the exposure time which is the time that the sensor observes a phenomenon, the calibration that are the values to use as reference and the sensing coverage.

❖ **Transceiver unit**

This unit connects nodes with the rest of the network and consumes more power than other units. The transmission medium is varied and includes radio frequency signals (RF), optical signals by laser and infrared. The communication using laser consumes less energy but requires line of sight between transmitter and receiver, communication via infrared signals has limited capacity to transmit, thus, the prevalent use for communication in sensor networks is radio signals, which require the use of antenna. The aim of the transceiver is to convert a sequence of bytes or frames sent by the microcontroller to radio waves and vice versa. The transceiver characterized by four operating states:

- *Transmit State*

In this situation, the transmitting section of the transceiver is active and the antenna emits radio wave

- *Receive State*

In this state, it is activated the receiver and receives data packets

- *Idle State*

In this situation, while the transceiver is ready to receive data, in practice it does not take anything. In the idle phase, many parts of the receiving circuit is active while others may be inactive

- *Sleep State*

In sleep state, the most important parts of the transceiver are inactive. There are transceivers that have multiple sleep states, which differ in the percentage of the circuit remains inactive, for recovery times and for startup energy

❖ **Power unit**

Energy consumption is one of the most important parameters in the design of wireless nodes. The power supply unit usually includes a couple of common AA batteries, rechargeable or not. The highest consumption is caused from the wireless communication system and occurs during the transmission of data, while the collection and the local processing of data causes lower consumption in relation to the transmission. For example, the transmission of a 1Kbyte within 100m spends much energy as spent for the execution 3 to 100 million instructions per second. Also, there is research to produce energy from the sun, heat and vibration.

Finally, from *figure 2-3* we see that there are other parts that we have not mentioned, because they are not important components of a wireless node but are used depending on the application. These are the Location Finding System, which can be a unit GPS, the Power Generator, for example, solar cells and the Mobilizer.

Design Specifications

The design of a WSN is affected by many factors, such as:

❖ **Lifetime**

A wireless sensor node, called as mote, is equipped only with a limited battery energy capacity. In some applications, however, the battery replacement is practically impossible (when a area is inaccessible), while the network should be operated at least for a predetermined time or as much as possible. The lifetime of a node therefore, seems to be closely related to the battery life and the life of the network is depended on the strength of the mote. As supplement of energy, can be used limited power sources (such as solar cells).Although these are not strong enough to ensure continuous operation, however, can provide an important help. For these reasons, conservation and energy management are among the most important factors in network design. The definition of the lifetime of the network is depended on the application. An option is defined as the elapsed time until left without any action, the first node. Other cases are up to the network is divided into two or more parts, or when is viewed for the first time area not covered by a node at least.

- **Quality of Service**

The concept of quality of service (QoS) in WSN networks differs significantly from conventional networks. In WSN networks do not play a dominant role in the high transmission rate and speed, but the reliable transfer of data. This transfer should be done without having to retransmit the message so consumes less energy. The traditional requirements for quality of service, such as delay or limit minimum bandwidth do not play an important role when applications are designed to be tolerant to the delay or when the bandwidth is small, as in the case of WSNs.

- **Fault Tolerance**

An important indicator of performance of a sensors system is the tolerance to errors. Some nodes may be closed or disabled due to lack of energy, due to interference from an external cause or even to suffer some physical harm, which will likely lead to a permanent breakdown in communication. This failure of individual sensors should not affect the functionality of the network as a whole. Here lies the reliability of a WSN with the tolerance to errors, which is an important indicator of the performance of a sensors system. Tolerances on errors in a WSN network depend on the respective application. If for example, the environment in which they are placed has low levels of interference, then automatically the tolerances are small. Therefore, a network of WSN is cost-effective and reliable, when the nodes that are developed in such a way as to achieve multiple coverage in the region from more nodes than those who normally would be required.

- **Scalability**

Depending on the application, vary the number of nodes involved in the formation of the network. There should be appropriate mechanisms to provide for the addition of new nodes and the expansion of the network, without impairing its

operation, while the already applied protocols should be adapted to the new size of the network.

❖ **Cost**

Basic unit of a WSN network, as already mentioned, is the sensor node. So is an essential condition for the prevalence of such networks, the complexity and cost of mote should be kept at low levels. The costs include network management. In addition to the ability to edit data, it must have the ability to self-organize and self-preservation, without human intervention. The network will be presented flexible and should change or select his various functional parameters. At the same time, it need to detect errors either to nodes or in communication and replaces the dysfunctional nodes. It should also be able to interact with the external maintenance mechanisms to ensure his extended operation at a necessary level of quality.

❖ **Hardware Constraints**

The limitations are posed by the hardware of the wireless nodes are related firstly with the size of these. At the same time, it should all the building blocks that make up it to work efficiently and in accordance with the requirements, so the node satisfies the requirements of the application that is running and the network, i.e. the low consumption, low cost, autonomy and easy adaptation to the environment.

Operating Mechanisms

To meet the mentioned specifications, it was necessary to find new mechanisms of network communication, such as new protocols and architectures that support them. These mechanisms are:

❖ **Multihop communication**

Direct communication between two nodes in WSNs can be unprofitable for some reason, such as for example when the two nodes are in a long distance so and require enough energy to communicate. This problem is treated with a mechanism that is called communication through multiple jumps, when conveying a data packet from one node to another, intermediate nodes are used to promote the package so that each node to spend minimal energy for transmission. The jumps that makes the packet from a node to another are called hops, hence the term multihop.

❖ **Self-Configuration**

A network of sensors should have the ability to form most of his functional parameters autonomously and independently of any external intervention. For

example, sensor nodes must be able to determine their location based on the location of other sensors in the network. At the same time, the network will be replacing those nodes are offline due to a lack of energy or other damaged.

❖ **Data Aggregation**

In some cases, data are obtained from neighbor sensors, express the same information, so it makes no sense to be transmitted to the base station for processing them. These data can be aggregated and processed locally in some node, through the algorithm that is applied and then are transmitted to the base. This process has the effect of reducing the amount of redundant data to transmit, and to improve the energy efficiency of the network.

❖ **Data Centric**

Conventional networks focus on data transmission between devices, each one characterized by a unique network address. This mode is called address centric. In WSN instead can apply a data - centric mode, since due to the large number of nodes, sensors for measurements of the same phenomenon can be grouped on the basis of this feature to have a common network address.

Communication Models in WSNs

The simplest forwarding model is the unicast model which includes switching a packet from connection to connection along the network, from the sender to the unique recipient (point-to-point network). However, there are other forwarding models such as multicast and broadcast model. In the broadcast transmission the packet is copied and the copies are sent to all nodes on the network or to a designated number of nodes, when the network consists of thousands of nodes. The transmission of the packet which is copied is done with a special code in the address field. In the multicast transmission, a copy of the package is sent to all the members of a specified set of nodes, not necessarily adjacent. The multicast is usually harder to achieve, while both this and the broadcast routing algorithms are more complicated in comparison with the unicast, which is the majority of Internet communications. [2]

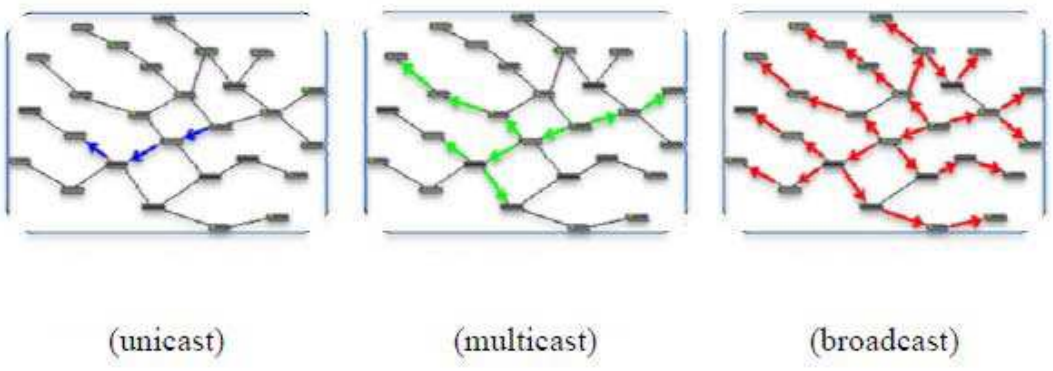


Figure 2-4 : Communication models in WSNs

Chapter 3
TinyOS: An Operating System for WSN

Introduction

To TinyOS developed and evolved from the University of Berkley and is open source software. This is an operating system, small in size and economical energy in the management of sensors. Provides a set of software building blocks from which the developer can choose the appropriate components. The size of these files is the order of 200 bytes so the size of the overall program remains minimal. The operating system that manages both the hardware and the wireless network is performing the measurements of the sensors getting routing and controlling energy consumption decisions.

Due to the limitations imposed by the nature of the sensors, a new programming language developed, the NesC, which implements the structural design requirements and the reuse of TinyOS for tiny sensors. For the implementation of the reuse, the TinyOS implements a architecture distribution into individual components (component-based). In addition to optimize the power management, it uses a performance model based on events (event-based) where events are leading programs and related resources are released at the end of their use. Furthermore, TinyOS is optimized for memory usage and offers high efficiency in terms of energy consumption. In addition, it provides interfaces between the individual sections that make it up and belonging to adjacent layers of his architecture. Such diagram layers of this architecture is shown in the next image. [2]

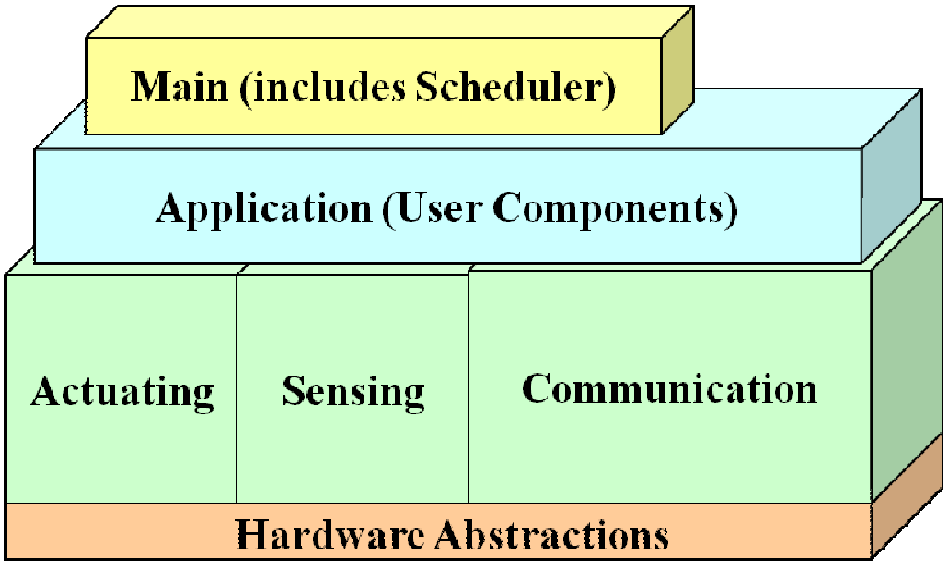


Figure 3-1: Layers Diagram of TinyOS Architecture

The adaptation of the TinyOS on different types of platforms is easy because of the easy and the removable hardware-level stratum in core. This stratification shall also facilitate the development of wireless networks. Due to its efficient design, great

community support and open source, the TinyOS became the most widespread operating system for WSNs.

The programming language NesC

Introduction

The first wireless embedded sensor systems were running on personal computers and used mainly Linux programs. When the development of these networks passed by microprocessors in microcontrollers, Linux had ceased to be the appropriate choice. The applications of systems of that era were deployed mainly in standard C language or directly in assembly language. However, this programming language is difficult to analyze and also can easily come out of control when the complexity of the application grows. In complicated systems, the problem may deal with object-oriented programming, which makes easier the separation of complex programs to independent easy composed parts. But the object oriented programming requires dynamic memory allocation and tends to require more programming resources, something which it considers unsuitable for embedded systems. The language NesC (network embedded systems C), which was developed by researchers at the UC Berkeley University, represents a promising field for application designers. It is properly designed for embedded systems networks and supports a programming model that integrates with environmental reactivity, concurrency, and communication capability.

Applications of sensor units (motes) are deeply linked to the material and each unit runs an application every time. So there are a number of unique challenges that the holistic system's design of NesC must address:

- **Driving from the interaction with the environment**: In contrast with the traditional systems, the motes used in collecting data and monitoring the local environment, rather than for general calculations. This particularity leads to two observations: the first is that the motes is basically driven by events (event driven), in response to environmental changes (arrival of a message, retrieve data from sensors) rather than driven by interactive or batch processing. The second observation is that the 'arrival ' of an event or process of data is coincidental activities, thus requiring an approach for concurrency management that deals with any errors (bugs) as conditions of competition.

- **Limited Resources**: These units (motes) have very limited resources, because of the particular needs for small size, low cost and low energy consumption. These restrictions are not expected to vanish, as the benefits from the expectation of Moore's law will continually leads to decrease in size and cost despite an increase in capacity in the same size.

- **Reliability**: Although it is expected that these units can suffer from damages due to hardware errors, there is a strong need for applications which can run

for a long time. For example, the environmental monitoring applications must be able to collect data without human intervention, for months at a time. An important goal is to reduce errors during execution (run-time errors), as there is no effective mechanism of recovery errors except the automatic restart of system.

- **Small requirements for real-time functions:** Although there are some operations that are time critical, such as management of wireless communication or monitoring of power of sensors, in general there are no great requirements for real time functions. Indeed, experience has shown that time constraints can be satisfied having absolute control of the application and the operating system and at the same time reducing the usage (utilization). One of the few critical in terms of time functions in sensor networks is the wireless communication. However, given the basic unreliability of radio link will not necessarily need to meet tough requirements in this area.

The NesC language is an extension of C and expected has similar syntax, however provides three important elements that it is differentiated significantly:

- The NesC language defines a component model that supports systems driven by events. This model provides bidirectional interfaces to simplify the flow of events and allows efficient and lightweight implementation without virtual functions and dynamic elements.
- At the same time provides a simple but specific concurrency model in combination with extended analysis during compilation: the compiler of NesC identifies most cases of competition data (data race) during compile time. This combination allows the creation of modern applications that require limited resources.
- Finally, the language NesC provides a unique balance between program analysis, to improve reliability and reduce code, and the ability to create complete applications.

Because the language NesC has proven to be effective in case of application development for wireless sensor networks used as the programming language for the operating system TinyOS. Investigations are underway to develop the standard of TinyOS and for other programming languages but so far NesC is the only language that can be used to develop programs in TinyOS.

Components and Interfaces

The NesC is a language based on components. The NesC components use a purely local namespace. This means that in addition to declaring the function that implements a component must also declare functions calling from another component. The names that a component uses to call these functions have strictly local scope: the

name to which it refers is not necessarily the same as that in which the operation is implemented. When a component (A) indicates that calls a function (B), introduces substantially the AB name as a global namespace. A different component (C), which calls the function B, inserts CB as a global namespace. Even though the A and C are referred in function B, can refer to completely different implementations.

Each component has a specification, a block of code that declares the functions that provides (implements) and functions that use (calls). In practice, the components very rarely declare their individual functions in their specification. Instead, the NesC has interfaces, which are collections of related functions. The reference standards are almost always in proportion to the interfaces. These interfaces are unique access points to the component and are interactive. An interface, generally, represents a service (such as sending a message), and is identified by a type of interface (interface type).

The linking of services and users together is called wiring, because the code is divided into components, so that special operating units are wiring, for running an application. A component can only refer to variables from its own local namespace and cannot be named variables in another component. However, it can be stated that uses a function that is defined by another component or that provides a function that another component can call.

The interfaces in NesC are, as we said, bidirectional. An interface represents a set of functions, called commands, that the provider of an interface must implement, as well as a set of functions, called events, which the user of an interface must implement. In order to call a component, the commands in an interface, must implement events for this interface. A unique component can use and provide more than one interface and multiple instances from the same interface. For example, the Timer interface provides start and stop commands and the fired event.

In the following figure, the provided interfaces are those that are shown over from the TimerM component, while the ones that are used, are below from it. The ' arrows ' pointing downwards, illustrate the commands, while those pointing upwards depict the events. Although the same interaction between the timer and the client could be accomplished via two separate interfaces (one for the start and stop commands and one for the event fired, grouping these commands and events to the same interface makes defining clearer and helps to avoid errors during interconnection of the components together. The functions are performed in two phases, so that can be easily modeled by placing requested commands and response events in the same interface. [2]

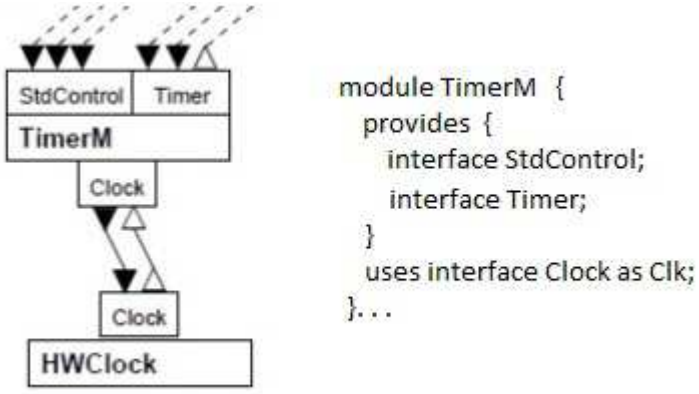


Figure 3-2: Graphical representation of the component TimerM

This separation in the definitions of the types of interfaces by using them in the components promotes the definition of standard interfaces, making the components more flexible and reusable. It is worth to note that a component may provide and use the same type of interface or provide the same interface more than once. In these cases, the component must give each instance of the interface (the interface instance) a separate name by using the characteristic word 'as', as shown in the figure 2.2 for the Clk. Finally, an important but critical issue is that two-way interfaces can support very easily the hardware interrupts. On the contrary, the interfaces based on unilateral calls' procedures dictate the hardware polling or using two separate interfaces for hardware functions and their respective interrupts.

The structure of TinyOS

In order to meet the required levels of parallelism (concurrency control levels), the TinyOS uses a structural model that is based on situations instead of threads. Converting the individual sections of processes (components) to 'state machines', creates the potentiality of efficient use of CPU and memory. So instead are devoted multiple stacks for each current process, can to share a single execution context between state machines.

Each component now uses events and commands to pass quickly from situation to situation. Typically, these transitions are instantaneous, requiring very little computing power. Thus, each component temporarily binds the processor for as long as these changes of statements last. If, of course, that requires more computing power, then a used process, which is scheduled by the scheduler and run without being interrupted by other processes (only the largest priority events) until its completion. The process that is followed in the scheduling of tasks is FIFO (First in First Out). The structure of TinyOS is shown in the following figure: [2]

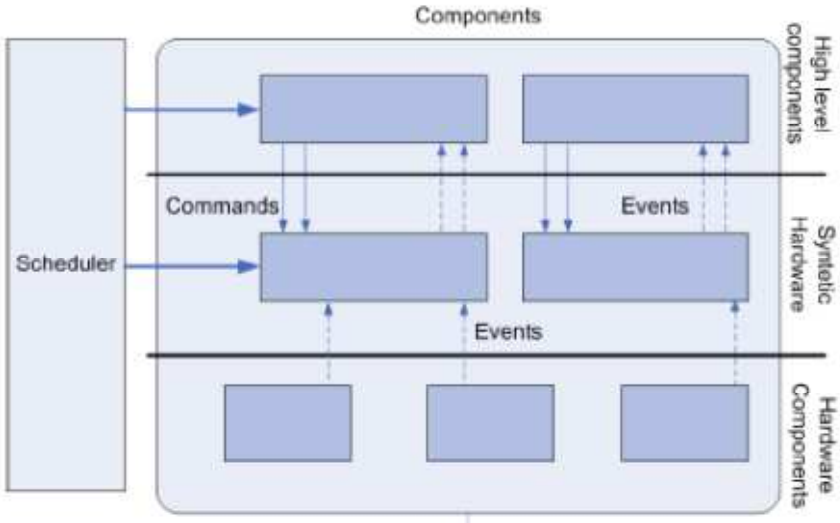


Figure 3-3: the structure of TinyOS

These components satisfy the requirement for sufficiently structured software architecture. Each component, as shown in the picture, consists of four parts:

- a Command Handler
- an Event Handler
- an amount of memory (Frames)
- a set of possible processes (Tasks)

The memory is used for saving the internal state of the component, while the processes as well as commands and event handlers are executed based on the contents of this memory box. In addition, each component indicates the commands that it uses and the events it generates. In this way, are created levels of layers of components, so that these higher levels each time using commands, call functions provided by the lower levels components and the latter to generate events to the first. Thus, it is introduced the concept of bi-directional interface which defines the commands and events that is used by the interaction of the components each other.

The commands are requests to lower level components. Each one of these returns to the component that is called information around his state. This information is taken by the command handler and is placed in the component's memory and then promotes a process in the stack of processes to run. The confirmation that the involved operation in each specific command completed with the creation an event from the lower-level component.

The event handlers react to events that are generated by lower-level components or when these are directly connected with the hardware, on interrupts. Also, as in commands, the contents of frame memory are modified and are created processes. Thus, the events can call commands, create new events and processes and terminate processes, and cannot be created by commands and interrupted by processes. The lower level events, of course, are created by hardware interrupts.

Processes perform the basic operations and intensive calculations. Run until their completion and can be stopped only by events. Also, are scheduled by the appropriate

scheduler with FIFO manner (serial), so theses must be short. However, it can be used a scheduler based on priority instead of FIFO, in order to reduce the overall delay execution of processes.

The Contiki Operating System

At this point it's worth to refer a few words about another lightweight and flexible operating system for tiny networked sensors, the Contiki. It is an open source operating system for networked, memory-constrained systems with a particular focus on low-power wireless Internet of Things devices. Examples of where Contiki is used include street lighting systems; sound monitoring for smart cities, radiation monitoring systems, and alarm systems. Contiki was created by Adam Dunkels in 2002 and has been further developed by a world-wide team of developers from Atmel, Cisco, Oxford University and many others. The name *Contiki* comes from Thor Heyerdahl's famous Kon-Tiki raft. Despite providing multitasking and a built-in TCP/IP stack, Contiki only needs about 10 kilobytes of RAM and 30 kilobytes of ROM. A full system, complete with a graphical user interface, needs about 30 kilobytes of RAM [36] .

Hardware - Low-power operation

Contiki is designed to run on classes of hardware devices that are severely constrained in terms of memory, power, processing power, and communication bandwidth. A typical Contiki system has memory on the order of kilobytes, a power budget on the order of milliwatts, processing speed measured in megahertz, and communication bandwidth on the order of hundreds of kilobits/second. This class of systems includes both various types of embedded systems as well as a number of old 8-bit computers.

Many Contiki systems are severely power-constrained. Battery operated wireless sensors may need to provide years of unattended operation and with little means to recharge or replace its batteries. Contiki provides a set of mechanisms for reducing the power consumption of the system on which it runs. The default mechanism for attaining low-power operation of the radio is called ContikiMAC. With ContikiMAC, nodes can be running in low-power mode and still be able to receive and relay radio messages [36] .

Networking

Contiki provides three network mechanisms: the uIP TCP/IP stack, which provides IPv4 networking, the uIPv6 stack, which provides IPv6 networking, and the Rime stack, which is a set of custom lightweight networking protocols designed specifically for low-power wireless networks. The IPv6 stack was contributed by Cisco and was, at the time of release, the smallest IPv6 stack to receive the IPv6 Ready certification. The IPv6 stack also contains the RPL routing protocol for low-power lossy IPv6 networks and the 6LoWPAN header compression and adaptation layer for IEEE 802.15.4 links.

The Rime stack is an alternative network stack that is intended to be used when the overhead of the IPv4 or IPv6 stacks is prohibitive. The Rime stack provides a set of communication primitives for low-power wireless systems. The default primitives are single-hop unicast, single-hop broadcast, multi-hop unicast, network flooding, and address-free data collection. The primitives can be used on their own or combined to form more complex protocols and mechanisms [36] .

Simulation

The Contiki system includes a network simulator called Cooja which simulates networks of Contiki nodes. The nodes may belong to either of three classes: emulated nodes, where the entire hardware of each node is emulated, Cooja nodes, where the Contiki code for the node is compiled for and executed on the simulation host, or Java nodes, where the behavior of the node must be reimplemented as a Java class. A single Cooja simulation may contain a mixture of nodes from any of the three classes. Emulated nodes can also be used to include non-Contiki nodes in a simulated network [36] .

Programming model

To run efficiently on memory-constrained systems, the Contiki programming model is based on protothreads.[11][12] A protothread is a memory-efficient programming abstraction that shares features of both multi-threading and event-driven programming to attain a low memory overhead of each protothread. The kernel invokes the protothread of a process in response to an internal or external event. Examples of internal events are timers that fire or messages being posted from other processes. Examples of external events are sensors that trigger or incoming packets from a radio neighbor.

Protothreads are cooperatively scheduled. This means that a Contiki process must always explicitly yield control back to the kernel at regular intervals. Contiki processes may use a special protothread construct to block waiting for events while yielding control to the kernel between each event invocation [36] .

Features

Contiki supports per-process optional preemptive multi-threading, inter-process communication using message passing through events, as well as an optional GUI subsystem with either direct graphic support for locally connected terminals or networked virtual display with VNC or over Telnet [36] .

A full installation of Contiki includes the following features:

- Multitasking kernel
- Optional per-application pre-emptive multithreading
- Protothreads
- TCP/IP networking, including IPv6
- Windowing system and GUI
- Networked remote display using Virtual Network Computing
- A web browser (claimed to be the world's smallest)
- Personal web server
- Simple telnet client
- Screensaver

Chapter 4
Standard 802.15.4 and Zigbee

Introduction

The hardware and the communication protocol which we used in this work, based largely on standards 802.15.4 and Zigbee. For this reason and because it constitutes an important development in the field of WSN, it is worth mentioning in detail to them.

The 802.15.4 standard defines specifications for the physical layer and MAC layer, low rate transmission for wireless networks (LR-WPAN) formed from stationary or moving devices supplied by batteries, or some other limited energy source. The design of the template focuses primarily on minimization of energy consumption of devices, in reliable data transmission and easy network installation, based on simple and flexible protocols.

On the other hand, the ZigBee Alliance provides the highest levels of the Protocol (from the network layer to the application level) for data used within the network, for security services and solutions in wireless home networks and control of buildings.

Compatible with standard, devices can operate in three frequency bands specified by this. In the area of 868 to 868.6 MHz, with only one channel and data rates up to 20 kbps, 905 - 928 MHz area, separated into 10 channels and data rates up to 40 kbps and finally separated into 16 channels of 5 MHz in 2.4 - 2.4855 MHz band, where transmission rates are achieved up to 250 kbps. Although that in the last two bands are available more than one channel, the MAC protocol of the standard uses one each time as we will see later. Although initially the LR-WPANs (Low Rate Wireless Personal Area Network) formed from nearby links up to 75 m, there is the possibility of increasing the scope of communication at the expense of data rate. In many applications, such a compromise is not only desirable but necessary. [38]

Structure of Network and types of Nodes

In standard are defined two types of nodes concerning the MAC layer:

- **Full Function Device (FFD)**
- **Reduced Function Device (RFD)**

A FFD node can operate in three ways: as a central coordinator of a PAN network, as a local coordinator in any area of the network and as a simple device. The network formed by FFD nodes that have potential to communicate with any node within their range, from RFD nodes that communicate only with the nearest FFD node and a central coordinator FFD node usually connected to a computer or another network. The FFD nodes are the backbone of the network while RFD is intended only simple operations. The devices can be used to formation three types of topology as it shown below (star, mesh, cluster - tree, Figure 4.1) [2].

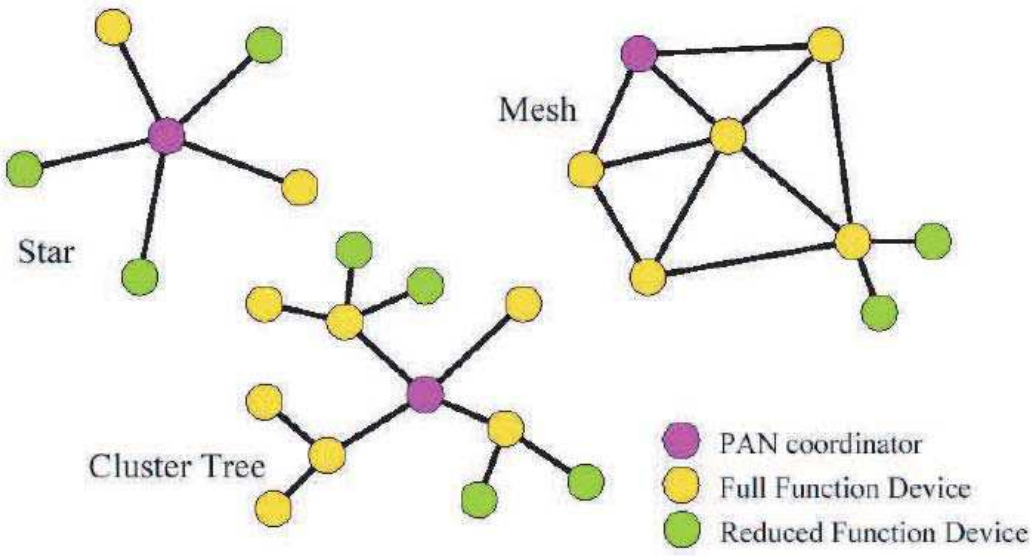


Figure 4-1: Topologies of 802.15.4 standard

Architecture of 802.15.4 standard

The architecture of 802.15.4 is defined through a set of separate layers, as in OSI model. Each level is responsible for a part of the template functions and provides services at the highest levels. So the architecture of a device is consisted of the physical layer, which includes the RF transceiver along with some low-level control mechanisms and the MAC level, which provides access to the physical channel for all types of transmission .

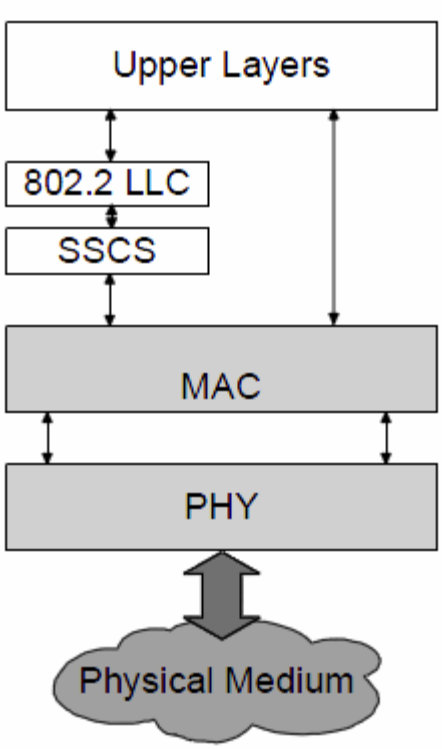


Figure 4-2: The architecture of a LR-WPAN device

The physical layer

The standard defines the allowable transmission technique for physical level DSSS technique, with BPSK modulation or O -QPSK. The physical level is carrying out the following tasks:

- **Receiver Energy Detection, ED**

It is an estimate of the received signal strength in the range of the channel, without recognition or decoding. The measurement result is stored as 8-bit integer and used by other network levels.

- **Link Quality Indicator, LQI**

After receiving a package the physical level calculates a quality assessment. The estimation can be done with the help of ED price, is also stored as an 8-bit integer and is available to upper levels.

- **Clear Channel Assessment , CCA**

It can be achieved in three ways. The first is by checking the ED price. If a certain threshold has been exceeded, the channel is considered occupied. The second is to detect carrier (Carrier Sense). The channel is occupied only if detected signal with modulation that is defined by the standard. The third way is a combination of two previous.

Finally, the physical layer is responsible for the final definition of the channel frequency and obviously for the sending and receiving of packets to / from physical media.

The structure of the physical layer packet

The header SHR consists of the preamble signal that serves to synchronize and from the SFD section, that specifies the end of SHR and the beginning of the rest of the package. PHR header is always 8 bits and contains information about the length of the frame. Last follows the data section of the physical layer packet (PHY Payload), which includes the MAC layer frame and is variable length, as it is shown below.

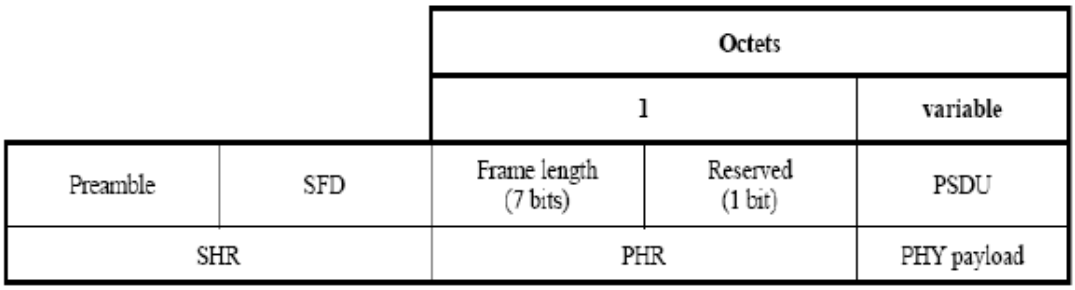


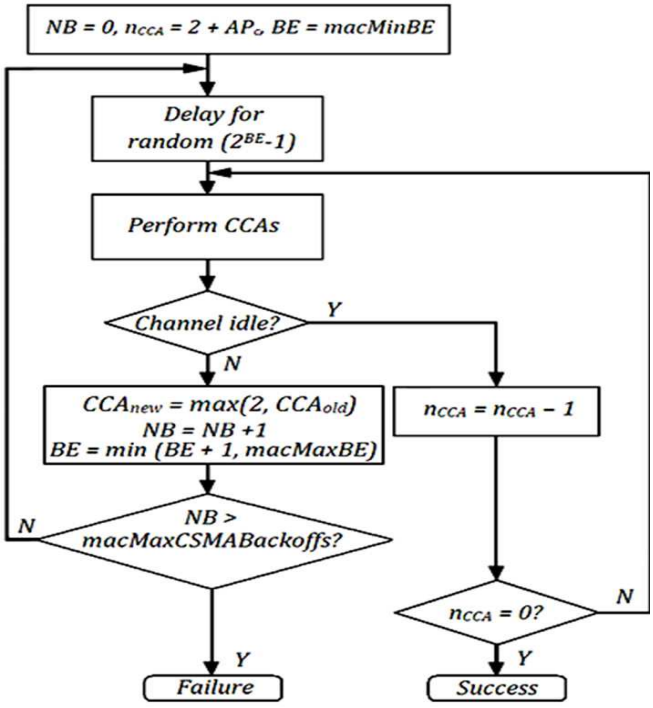
Figure 4-3: Structure of the Physical layer packet

The MAC Layer

The MAC layer uses a Multi Access Protocol with Carrier Detection and Collision Avoidance or CSMA -CA (Carrier Sense Multiple Access with Collision Avoidance), through which is selected the time that will emit a device or to put in waiting to receive a packet. The protocol provides synchronization using beacon frames and time slots (slotted CSMA), so that a unit coordinator can communicate with many simple devices to avoid large percentage of the collisions, and the unnecessary consumption of energy as the devices are turned off at selected sockets. The slotted CSMA - CA is ideal for star topologies where communication is a master with many slaves. For peer networks where communication occurs between router devices, more appropriate is the unslotted version of the protocol, such as mesh networks. It follows a short description of this version.

Unslotted CSMA – CA (figure 4-4)

The algorithm of the Protocol provides for the detection of the channel before transmitting. If it is idle it starts to broadcast. As broadcast does not detect the channel, but sends the entire frame, which can be damaging to the recipient due to interference there. If the channel is busy, the sender shall defer the transmission until the channel to become idle and then begins to transmit. If a collision occurs, the clashed devices expect a random time (**backoff**), using the regression algorithm of Ethernet and retries later[2].



NB: number backoff periods that the transmission has postponed,
BE: exponent of regression algorithm, **macMaxBackoffs**: maximum allowable value of NB
macMinBE: initial value of BE, **macMaxBE**: maximum value of BE

Figure4-4: The flowchart of unslotted CSMA-CA algorithm

MAC Frame Structure

The MAC frame of 802.15.4 standard consists of MHR header and the MAC data segment. The information contained in MHR relate to addressing, frame type (i.e. the type of data contained in the payload) and the count of the package.

Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/ 14	variable	2
Frame Control	Sequence Number	Destination PAN Identifier	Destination Address	Source PAN Identifier	Source Address	Auxiliary Security Header	Frame Payload	FCS
		Addressing fields						
MHR							MAC Payload	MFR

-General MAC frame format

Figure 4-5: MAC frame structure

Zigbee

Although many times it is created the impression that the standard 802.15.4 and Zigbee are identical, this perception is incorrect. The standard Zigbee has emerged from the cooperation of many companies and essentially is an extension of the 802.15.4 protocol stack, as it implements the network and application levels, based on the services they provide the physical and MAC level of 802.15.4. The key features of Zigbee is a low data transmission rate, support up to 254 devices in a star topology (and theoretically infinite in mesh topology) and fast rewind devices from SLEEP mode.

The Zigbee protocol stack is based on the OSI model. A detailed description of this is shown in figure 4-6. The first two layers are defined of 802.15.4 standard while the other layers of the Zigbee protocol.

Network Layer

The network layer bridges the two standards, ensures correct operation of the MAC layer and simultaneously provides the appropriate services to the application layer through NLDE - SAP and NLME – SAP units. The NLME unit undertakes to create appropriate packets based on the received data from the upper level and decides on their proper routing. Also, the NLME unit provides a variety of services including installation of the new network, the recognition of neighboring devices, the search and record of routes, addressing newcomer devices and the selection of packet routing mechanism.[2]

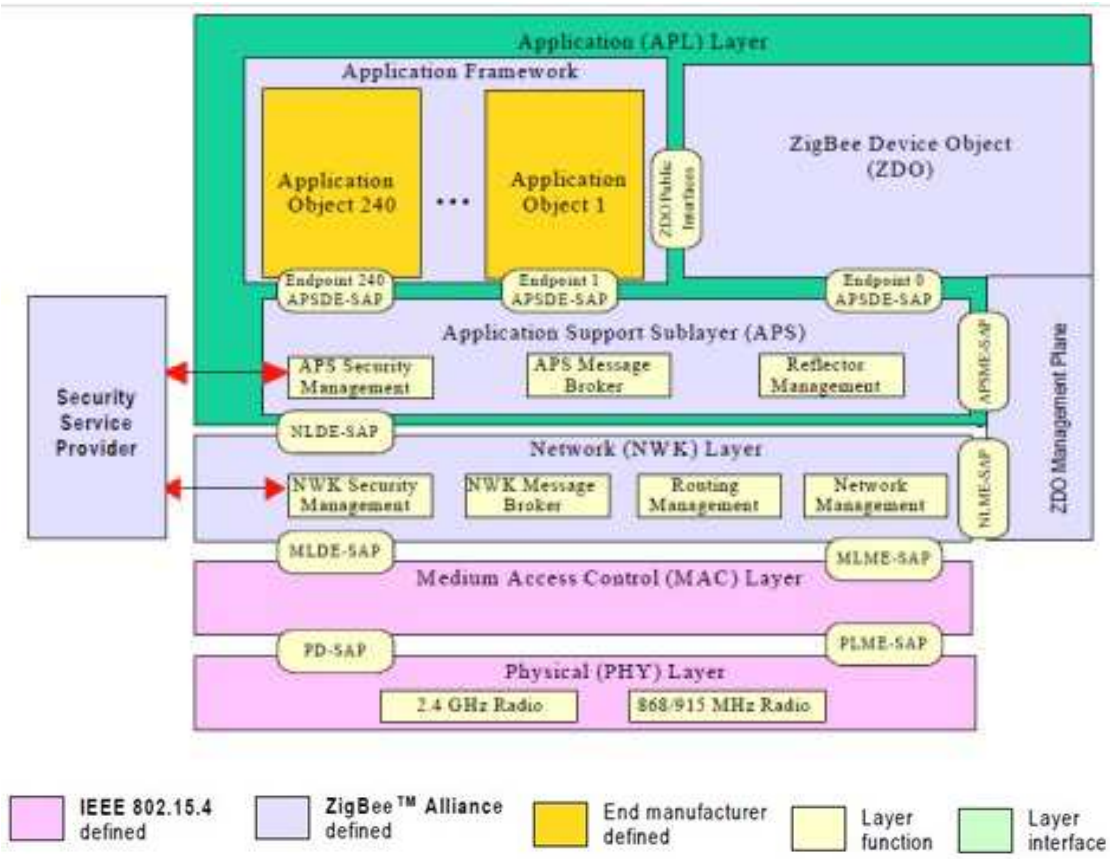


Figure 4-6: Zigbee protocol stack

Application Layer

This is the higher and more complex level that is defined of the standard. It consists of the Application Objects, the Zigbee Device Object and the Application Support Sub layer.

- **The Application Objects** are essentially applications running in a Zigbee device and are subject to one of the many profiles that are defined by the model, as are the profiles of Bluetooth.

- **The Zigbee Device Object** provides the interface to Application Objects that are used to identify other devices and services that they provide. The ZDO determines the role of a device on the network. The ZDO is also an Application Object that is implemented first in each device.

- **The Application Support Sub layer provides** the service for the exchange of data between two or more Application Objects.

Packet Routing in Zigbee Mesh Networks

The routing algorithm in Zigbee mesh networks based on the central idea of the algorithm AODV (Ad-Hoc on Demand Distance Vector), where each router participates in the frame relay from a specific source to a destination, creates a new record for the route in a routing table that is stored in memory. The minimum elements that can contain this entry are the distance to the destination (usually measured in hops or jumps – transmission costs) and the address of the next router on the way to its final destination. The paths are formed upon request, using a process to search a path in which the device – source transmits a Request packet (route request) (figure 4-7.a) and the device – destination sends back a Response packet (route reply) (picture 4-7. b).

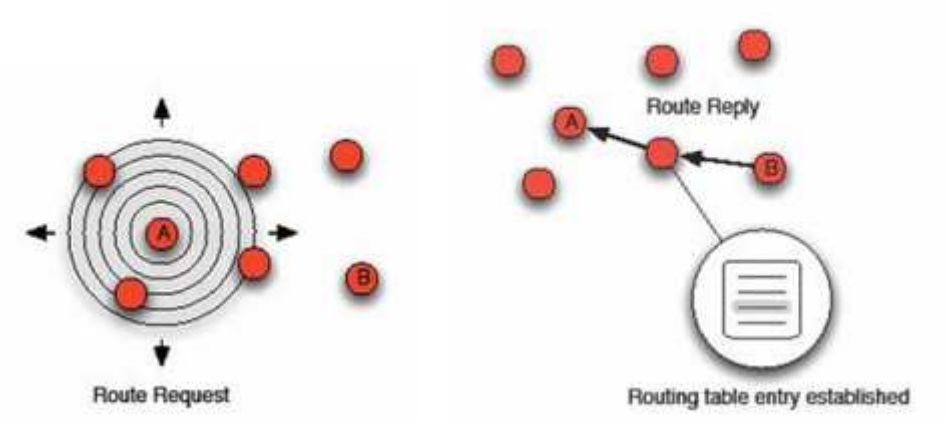


Figure 4-7: (a) the device A sends a routing request to destination B to all the neighbors. (b) the request finally arrives in B which sends the response. Along the path are created new records in routing tables that define the new path.

Just the data of all the routing tables of intermediate nodes are renewed, the route is ready for use (figure 4-8). Depending on the existing hardware and the operating conditions, there are many variations of the basic AODV algorithm that aimed at reduction of RAM which the routing tables occupy, but also to reduce network traffic caused by the process of formation of paths.

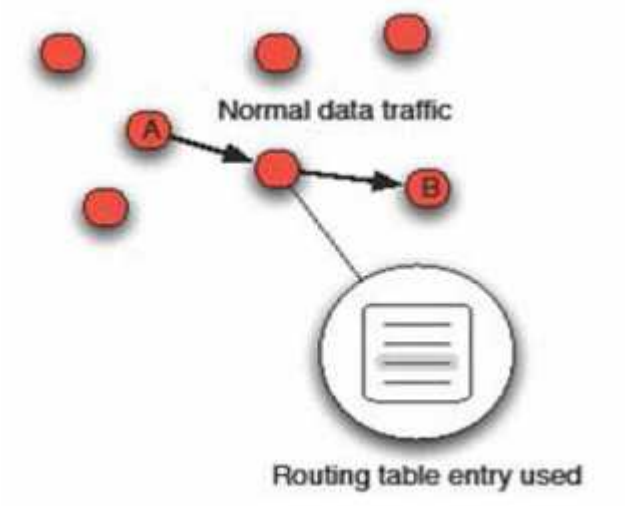


Figure 4-8: Using the path that is listed in the routing tables.

Chapter 5
XMesh Communication protocol

Introduction

The XMesh is a network protocol that was designed by the Crossbow Company for multi-hop, ad-hoc and mesh WSNs. It is basically a software library which uses the TinyOS operating system that runs on MICA2, MICAz, MICA2DOT and IRIS nodes of the same company. The aim is the communication between the nodes, even if one is not within the communication range of each other, which is achieved with the technique of multi-hopping, namely message forwarding to intermediate nodes – stations, until they reach their final destination. A large part of the Protocol is based on ZigBee and 802.15.4 standards. [11]

The units that make up a multi-hop mesh network are four types:

- **Endpoints:** These are nodes that are located in the edges of the network. They send only their own data and not forward packets that may arrive from neighboring nodes.
- **Routers:** These create their own messages and simultaneously operate as intermediaries, promoting packages that take from neighboring nodes.
- **Gateways:** These collect data from the network and provide the interface with the central station of the network. They are essentially the gateway through which network parameters are configured and supervised their operation.
- **System Software:** It provides the network protocol that allows the self-organization and self-regulation of the network.

XMesh features and advantages

The XMesh has several options and features, some of which are:

TrueMesh

TrueMesh technology refers to the ability of nodes to make dynamic search of new routes for the delivery of packages, especially when parts of the network going down either due to interference, either because the case where one node has exhausted his energy or just lie about the specific time in sleep mode. The nodes are scattered

across the network, recognize each other, and create a tree of paths based on the strength of the signals they receive.

Multiple transmission types

The XMesh provides three ways of communication between nodes:

- **Upstream:** Deliver packets from one node to the central station.
- **Downstream:** Delivery packets from the main station to one or more nodes.
- **Single-Hop:** Deliver packets only to neighboring nodes.

Quality of Service (QoS)

It provides two options regarding the reliability of communication:

- **Best Effort**: In this case it becomes the best effort packet delivery locally. The node that is waiting for acknowledgment from the neighbor and if not received it again tries (Link Level Acknowledgement).
- **Guaranteed delivery**: For each a message sent upstream or downstream, an ACK signal is always sent back to the node that sent the original message (End to End Acknowledgement).

Energy States

A node can be set to three different energy modes:

  - High Power (HP)
  - Low Power (LP)
  - Extended Low Power (ELP)

For energy management strategy of XMesh nodes will refer in more detail below.

Health Diagnostics

There is the possibility of the nodes to send periodic information about their status. The information concerns data such as network traffic, the battery voltage, the parent node and the node's neighbors and the power of the signal that they send (RSSI indicator).

Time Synchronization

In LP mode network synchronization is supported on the basis of a general time constant (± 1ms). This way syncs not only messages, but also measurements of the sensors.

Over the Air Programming

Possibility of reprogramming nodes remotely through messages sent downstream from the main station. It works only for nodes in HP status.

Watch Dog

The XMesh has the WDTM.nc component which enables a node to make a watchdog reset in case it loses 5 RUM. The watchdog timer activation increases the current in SLEEP mode in 15mA.

Power Management

As mentioned in the chapter on the TinyOS the power management of processor is done directly by the operating system. The programmer does not have direct access to commands that alter energy mode of the microcontroller. Instead the TinyOS itself, through a series of checks that are performed by the scheduler and the module HPLPowerManagement decides when to put the processor into SLEEP mode or IDLE, and what peripherals will stay active, depending on the options that have the model of the processor. Since the issue of CPU consumption is solved, the onus falls on the energy management of the RF transceiver, via the communication protocol. The XMesh as designed based on TinyOS, follows a similar energy management strategy. The developer is asked to choose among three types of operation that have already been implemented by the manufacturers.

XMesh HP

In this mode, the transceiver is permanently active. The consumption ranges from 15 to 30 mA, depending on the model. The node is capable of receiving and sending packets at any time and acts as router. RouteUpdate and Health packets sent at high rates per 36 ", in result to reduction of the network-forming time and the entry of a new node at this.

XMesh LP

The communication of nodes can be done in two ways, with synchronization or asynchronously. In the second way the process of node's communication with regard to receiving messages based on the preamble sampling technique, while regarding the sending, used the mechanism of CSMA– CA of 802.15.4 Protocol. The default for the wake-up sequence is 125 ms, which means that the node awakes to hear, 8 times per second. The waiting time T after it sends the wake-up sequence is 15 ms. The total duration of the preamble signal is:

Preamble = wake-up sequence + T = 140 ms

The RouteUpdate packets are sent much more infrequently than in the HP status and network bandwidth is lower. This has resulted in that the network has been formed later, but this is not great for networks that are going to work for very long without a lot of changes in their composition and layout. The basic consumption is 80 μA, while for a network 50 nodes with sending of packets per 3 minutes, the average is estimated at 400 μA.

XMesh ELP

Appropriate only for nodes located at the edges of the network (endpoints). The operating procedure that is followed is as follows: After the opening of a node, it remains ON until it normally enters at the network. After it finds the neighboring nodes, stores their addresses, select one as a parent based on signal strength and then immediately puts the transceiver in a state of SLEEP. When the node wakes up to transmit measurements will first try to send to the parent who had initially chosen. If it fails it will try with another nearby node that had been saved. In this situation the nodes spends minimal amounts of energy and theoretically can operate for years. However, one major drawback is that the designers have not predicted the existence of a preamble signal, resulting in a node in ELP can communicate with just nodes in HP mode and not with those who have set to LP. [11]

Parameter	XMesh-HP	XMesh-LP	XMesh-ELP
Route Update Interval	36 sec.	360 sec.	36 sec if built using HP 360 sec if built using LP
Data Message Rate	10 sec, typ.	180 sec., typ.	N/A
Mesh formation time	2-3 times Route Update Interval for mesh with average of 2.5 hops		
Average current usage	20-30 mA	<250 µA [1] < 400 µA [2]	50 µA

Figure 5-1: XMesh Performance Summary Table

Formation of a multi-hop network

For the formation of the network, two parallel processes run, Link Estimation and Parent Selection.

Link Estimation

A node which listen the channel picks up information about network traffic in his neighborhood. This information is used to create a table that lists the addresses of neighbors (neighborhood table). At the same time controls how well hears a neighbor watching the values of a variable in the header of the packets. This variable contains every time a different serial number (sequence number). Based on these values, assessment takes place through an algorithm EWMA (Exponentially Weighted Moving Average). In particular the sequence numbers that taken is known how many packets were normal received and how not (missed). Then the calculation of assessment of receipt (RE) is done by applying the EWMA algorithm in percentage of received packets, so and shows that:

Received_percentage = 255*received/(received+missed)
RE = (1-a)*RE + a*Received_percentage

Where **a** is the EWMA factor with values from 0 to 1, while 255 is used for normalization so that the **RE** to ranges in [0, 255]. The size of the neighborhood table is 16 for nodes in the network and 40 for the node of the central station. If anyone finds more than 16 neighboring nodes, then will delete them with the lowest quality of emission from the table.

Parent Selection

After the node detects the neighbors and forms the neighbor table, selects through this, one parent based on the minimal cost of communication. A node may be a candidate parent if fills in the following conditions:

- It has already entered into the network
- It there was not child (child node) of the node for the three last RUI (Route Update Intervals), so as to avoid cycles in the network formation
- It is not in a ELP state

Route Update Messages (RUM)

The final stage of the formation of the network is achieved by periodically broadcast by all nodes, messages that contain information about available routes on the network. These messages are called Route Update Messages (RUM) and the time that elapses between two successive emissions Route Update Interval (RUI). The value of RUI and the structure of RUMs are common to all nodes in the network. (For characteristic values RUI see figure 4-1)

The information carried by the RUM is:

- **Parent ID:** If the node has not yet joined the mesh, this field is 0XFFFF.
- **Cost:** It tells other motes in the neighborhood how much it will cost to send a message upstream to the base station.
- **Hop count:** The number of hops to send a message to the base station.

This information is determined by variables such as:

- **RE:** see above.
- **SE (send estimation)**: When a node transmits a RUM, incorporates in it and the price, RE. The nodes receiving the message based on the RE, calculate the SE.
- **LC (Link Cost)**: Link Cost to a specific neighbor. Once SE and RE are known, the LC is computed as:

$$LC = (1<<18) / (SE*RE)$$

- **NC (Neighbor Cost):** Cost of sending a package to a neighbor.
- **OC:** Cost of sending a packet from the node to the central station. It is the main criterion for the selection of the parent and is computed as:

$$OC = LC + NC$$

Sending and receiving Packets

Messages Structure

XMesh messages are an extension of the messages found in a TinyOS application. The difference is due to an extra header, which adds the XMesh immediately after the basic header of TinyOS. In figures 5-2 and 5-3 show the structure of a message in general and in detail, respectively.

Component	Description
Header	Composed of standard TOS header and XMesh multihop header
Payload	User data
CRC	CRC check (Refer to Appendix E)

Figure 5-2: XMesh Message Structure

Type	Name	Description	Component
uint16_t	addr	Destination address (next hop)	TinyOS Header (MICA2). The MICAz header has 5 additional bytes.
uint8_t	type	AM type; defines type of message	
uint8_t	group	AM group	
uint8_t	length	Remaining bytes in message, N/I CRC	
uint16_t	sourceaddr	Address of mote that sent the message	XMesh multihop header
uint16_t	originaddr	Address of mote that originated the message	
uint16_t	seqno	Message sequence number.	
uint8_t	socket	Application ID	
uint8_t	data	Payload	Payload size is set with the TOSH_DATA_LENGTH define. Default size is 29 bytes
uint16_t	crc	CRC check (Refer to Appendix E)	CRC

Figure 5-3: XMesh Packet Structure

In the header of the TinyOS, includes very basic information about the identity of the message, which is the destination (addr), the message type and the group to which the message belongs. Message type gets specific values and from them the recipient understands what type of message will be received. Some of the types of messages are the AM_HEALTH, for packets with information about the status of the node, the AM_DATA2BASE, which means that the message is directed towards the base station without waiting for ACK or AM_UPSTREAM_ACK signal, i.e. ACK message from node towards the base. The group is a value that identifies the network. If a node receives a message with *group* different from the known value (0x7D), will ignore it, believing that sent from other network. End, the value *length* indicates how many bytes are still remaining excluding two of the CRC.

In the header of the XMesh are contained the address of the node that first created the message (originaddr) and the address of the node who sent it last (sourceaddr). So the recipient knows who's promoted not only the message, but also who created it originally. *Seqno* value enables to the recipient to know if messages come with the correct order while the socket is a value that characterizes the network.

Next follows the section that contains the data whose size can be up to 34 bytes and at the end of the message are the two bytes with the value of the CRC. The maximum total size of packet is 55 bytes. Larger packets cause a great increase in the use of SRAM.

XMesh Messaging API

As mentioned in the beginning of the chapter in the XMesh Messages can be transmitted UPSTREAM (from node to basis) or DOWNSTREAM (from the base station to the node). The DOWNSTREAM path is always the same as the previous UPSTREAM and to achieve DOWNSTREAM communication with a node, it should have sent at least one message to the base. Messages can be sent with two options QoS (Quality of Service).

- **Link Level Acknowledgement:** It concerns the communicating between two adjacent nodes. The sender sends the message again if he do not receive ACK signal from the neighboring receiver. But the acknowledgement is only local. So for a multi-hop message is not guaranteed its arrival at the final destination. Using the LLA option it saves energy and is useful for applications that do not require 100% delivery of data.

- **End to End Acknowledgement:** In this case apart from the local ACK, is sent and another of the final recipient of multi-hop message. Unlike before, if the End to End Ack is not reached, the choice about whether the message will send is not automatic but is left to the user. Apparently the guaranteed transmission has paid an increase in energy consumption and bandwidth.

The XMesh provides a set of interfaces that the user can connect to the components of TinyOS, through the process of wiring, so that calling their commands in the application. We only refer them: the MhopSend Interface, Receive and

ReceiveAck Interface, Intercept and Snoop Interface, PromisciousSniff Interface and RouteControl Interface.

Health Packets

There are two types of packets: the statistics health and neighbor health. The first include statistical data about the packets which the node sends (number sent packets, number forwarded packets, number resent packets etc) as well as the battery voltage that feeds it. The neighbor health contains information for the "neighborhood" of the node. Each neighbor health packet can contain up to five id adjacent nodes together with data concerning the connection's quality with these (LQI) and the distance from the central station (hop count). The send health packets becomes periodically with period defined by the user. The send statistics health packets alternates with the sending neighbor health packets. If the node has more than five neighbors sent two consecutive neighbor packets, as shown in the picture below.

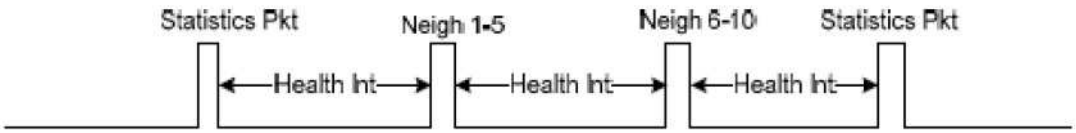


Figure 5-4: Send packet sequence Health

Chapter 6
The Hardware

Introduction

The hardware that is used for our application as we see in the next chapter, was designed and manufactured by Crossbow company and composed of four wireless nodes type Micaz, board-type sensors MTS400 and a programming board type MIB520. In this section is given a brief description of the structure, function and capabilities of these components.

Micaz Module

The micaz is one of the latest generations of wireless sensors that were first developed at Berkeley University in California and is produced by the Crossbow company (now Crossbow have bought by Memsic Company). Micaz consists of microcontroller Atmega128L of Atmel, is compatible with IEEE 802.15.4 protocol, transmitting in the frequency band of 2.4GHz and is suitable for low-power wireless networks. The Micaz has some new features that take full advantage of the functionality of the family of wireless networking products MICA of the Crossbow. [12] These include:

- RF transceiver that is compatible with the IEEE 802.15.4/ZigBee protocol.
- Global emission zone 2.4-2.483GHz.
- Technology DSSS - Direct Sequence Spread radio Spectrum which is resistant to RF interference and provides data security.
- 250 kbps data rate.
- TinyOS 1.1.7 operating system and later versions, including the software of Crossbow for creating mesh network.
- Plug and play functionality for all the sensor boards of Crossbow, data acquisition boards, gateways and software.

Figure 6-1: MICAz module top view

In the figure below, is graphically presented inside the micaz, where it is shown the microcontroller, the RF transceiver, the flash logger and 51-pin socket that supports analog inputs, digital inputs/outputs, interfaces I2C, UART and SPI which make it easy to connect with other peripherals. The sensors that can accommodate are barometric pressure, temperature and humidity, light, dual-axis accelerometer and other boards of Crossbow.

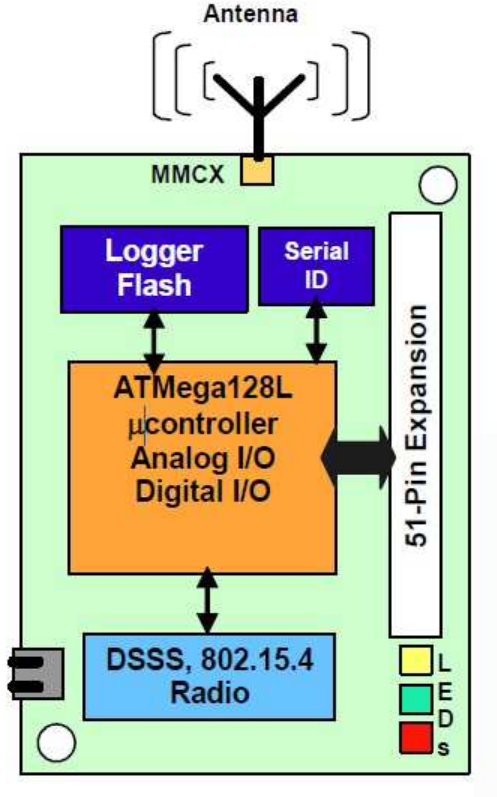


Figure 6-2: The Micaz Hardware Architecture

Some of the applications in which the micaz is suitable for use are:

- Environmental applications (measuring temperature, humidity, pressure).
- Monitoring facilities inside buildings.
- Security applications in buildings and bridges by measuring vibrations.
- Applications in vehicular traffic (telemetry).

The following table summarizes the main features of the central units of the motes of the Crossbow (mica2, micaz and iris).

Platform	MICAz	MICA2	M9100	M2100	M2110
Microprocessor	ATmega128L	ATmega128L	ATmega1281	ATmega1281	ATmega1281
Radio	CC2420 (2.4GHz)	CC1000 (433MHz,916MHz)	CC1000 (433MHz,916MHz)	CC2420 (2.4GHz)	RF230 (2.4GHz)
External Serial Flash	AT45DB041 512 Kbytes The serial flash can be used for over-the-air-programming (OTAP) and/or data logging				
Unique ID (integrated circuit)	DS2401P 64-bit This chip contains a unique 64 bit identifier.				
51-Pin Connector	Yes, except for OEM modules This connector brings out most of the ATmega128L signal				

Figure 6-3: The main features of micaz, mica2 and iris.

MTS400/MTS420 Sensor Board

The MTS400 offers five basic environmental sensors with an additional GPS module option (MTS420). The features offered on these boards allow for a wide variety of applications ranging from a simple wireless weather station to a full network of environmental monitoring nodes. Applicable industries include agriculture, industrial, forestry, HVAC and more. These environmental sensor boards utilize the latest generation of low-power digital integrated sensors which provide extended shelf life.

The GPS module offered on the MTS420 (figure 6-4) may be used for positional identification of Motes deployed in inaccessible environments and for location tracking of cargo, vehicles, vessels, and wildlife. [13]

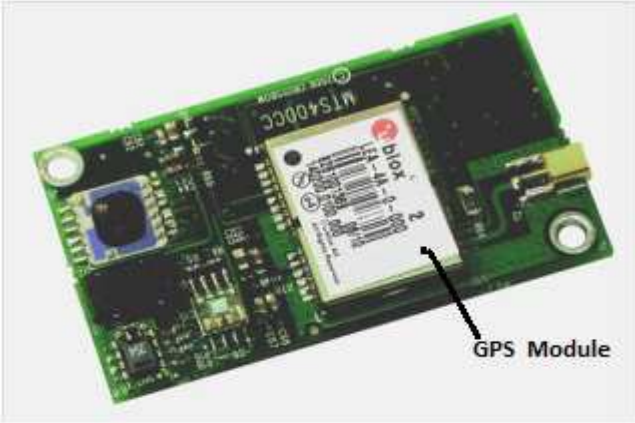


Figure 6-4: Photo of MTS420CC. The MTS400 does not have the GPS module

Humidity and Temperature Sensor

The humidity and temperature sensors are located both on the same integrated circuit SHT11 of Sensirion company , that is supplied with voltage from 2.4 to 3.6 V, has internal A/D Converter 14-bit, for the conversion of the measured analog signals to digital and TWI (2-Wire Interface), for communication with the microcontroller (it is shown in figure below).

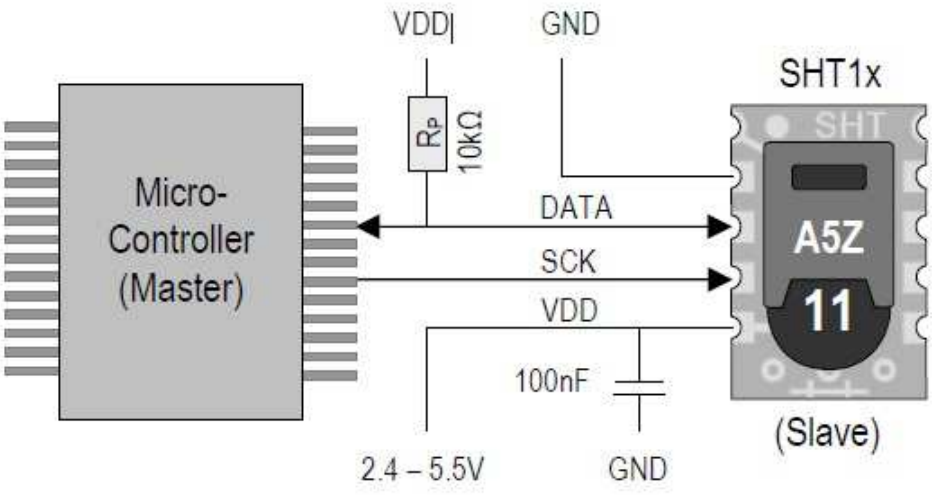


Figure 6-5 : Typical application circuit, including pull up resistor R_P and decoupling of VDD and GND by a capacitor.

Sensor Type	Sensirion SHT11	
Channels	Humidity	Temperature
Range	0 to 100%	-40°C to 80°C
Accuracy	± 3.5% RH (typical)	± 2°C
Operating Range	3.6 to 2.4 volts	
Interface	Digital interface	

Figure 6-6: Summary of the Sensirion SHT11's Specifications

With the connection of the supply voltage the sensors are ready for use in 11ms. The control of their operation is done with a command set (figure 6-5) that the controller sends via TWI.

Command	Code
Reserved	0000x
Measure Temperature	**00011**
Measure Relative Humidity	**00101**
Read Status Register	00111
Write Status Register	00110
Reserved	0101x-1110x
Soft reset, resets the interface, clears the status register to default values. Wait minimum 11 ms before next command	**11110**

Figure 6-7: SHT1x list of commands

After a temperature or humidity measurement command, the controller must wait until its completion. For 14-bit measurement, the maximum time that is needed is 320 ms. The completion of the measurement is signaled by the transition of the pin DATA to low state. The measurement is not necessary to read it immediately as it is stored in a register of the circuit. In the following figure is graphically presented the reception and emission procedures of measurements on the controller.

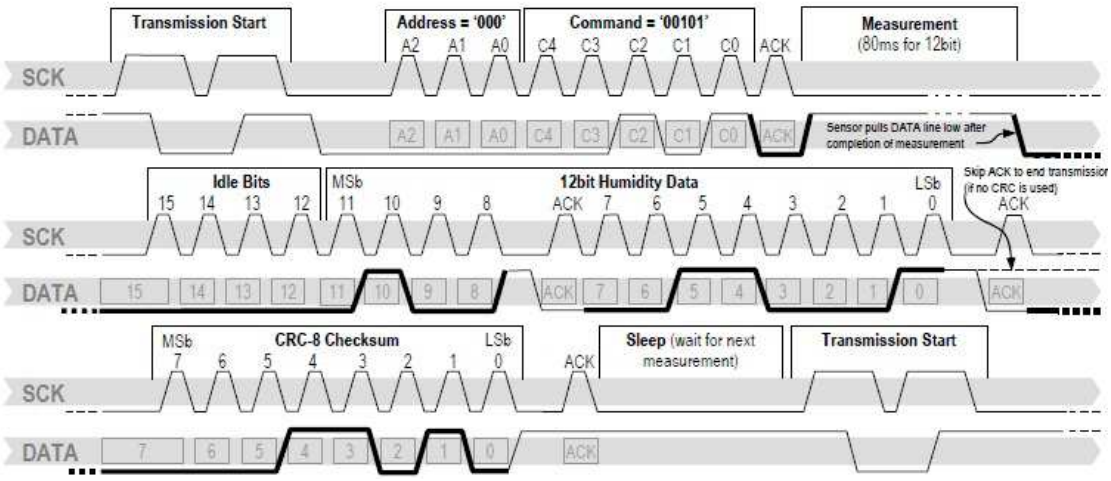


Figure 6-8: reception and emission procedures of measurements on the controller.

For the humidity sensor the relationship between output of ADC and actual value of measurement shows a non-linearity as illustrated below.

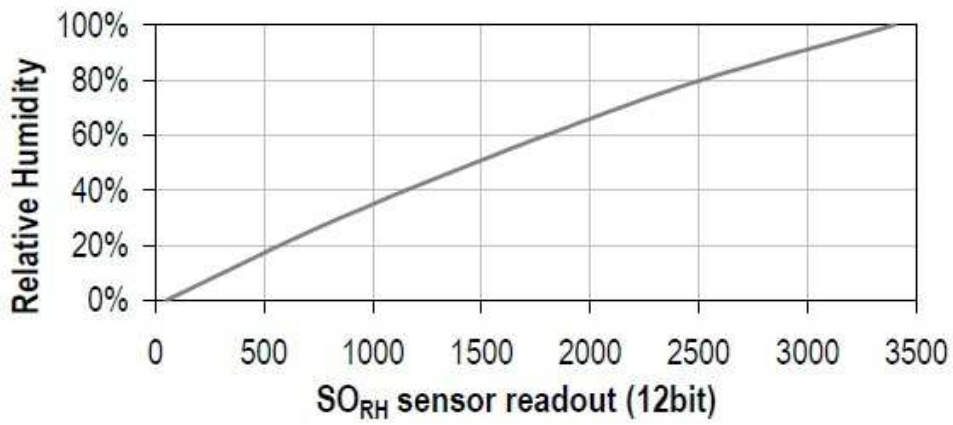


Figure 6-9: Conversion from SO_{RH} to relative humidity

Using the formula:

$$RH_{linear} = -4.0 + 0.405 * SO_{RH} - 2.8 * 10^{-6} * SO_{RH}^2 \ ,$$

Where SO_{RH} the output value of the ADC, we get a more accurate approximation of actual humidity.

Unlike the humidity sensor, the temperature sensor is linear and the only conversion that the value of output of the ADC takes, is given by the formula:

$$T = -38.4 + 0.0098 * SO_T ,$$

Where SO_T the output of the ADC.

Finally as regards energy consumption values are given in table below.

Parameter	Condition	min	typ	max	Units
Source Voltage		2.4	3.3	5.5	V
Power Consumption [5]	sleep		2	5	µW
	measuring		3		mW
	average		150		µW

Figure 6-10: Energy consumption SHT11

MIB520 Programming Board

The MIB520 provides the possibility to ISP (In System Programming) programming Micaz motes, via the ATmega16L processor that has, as well as the possibility of serial communication with PC (host PC) for data transfer on the network leading to the central hub station.

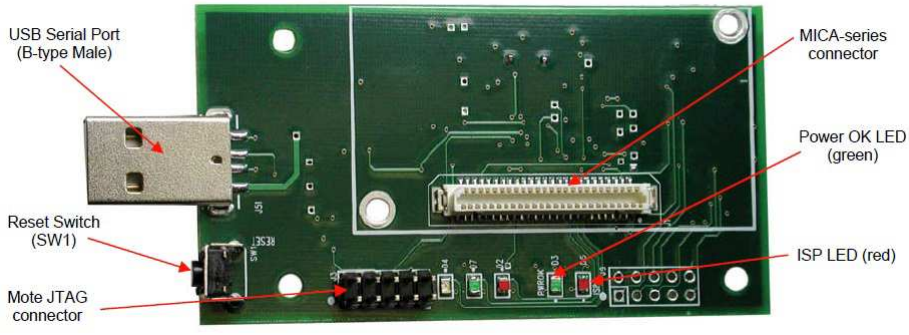


Figure 6-11: Photo of top view of an MIB520CB.

Linking MIB520 Board with computer where is installed TinyOS and MoteWorks software of Crossbow, done via USB. With installing special drivers, the USB port is working like two virtual serial COM ports. By first, it performs the programming, and through the second, performs data exchange between host PC and the rest of the network.

Chapter 7
The Application

Introduction

The aim of the application is to implement a mesh network for surveillance of space i.e. for a forest area, consisting of Micaz nodes and a central computer - host PC, to periodically collect humidity-temperature measurements and their reliable transmission and processing, firstly at local node level and finally to the central station, where it can become the entry in the database for the study or assessment fire risk. Furthermore it is given importance on as much as possible more energy saving of nodes and bandwidth of network. The application is purely experimental as the available hardware is limited, so we are going to implement a small-scale network. Nevertheless, there may arise some useful conclusions, as the application can be extended to a larger scale to cover an area as a forest with an increase in the number of nodes. We consider the simple case where each node collects a couple of measurements (humidity-temperature) at time T (3 minutes), and sends them to the central station.

The available hardware is 4 Micaz nodes, 4 MTS400 Sensorboard, the MIB520 programming board and of course a central computer that has installed the necessary software. Each one of the three nodes bears the MTS400 and collects actual measurements (temperature-humidity), while one node is connected through MIB520 with host-PC and forms together the system of the central station. All nodes have the potential to promote messages and depending on the distance between them can be communicate directly with each other. So each node can send a message either directly to the central station or to promote it at the base through another node. The formed network is mesh type, since all nodes act as routers that transmit multi - hop messages.

Figure 7-1: Our WSN Application

The nodes regularly collect the information such as temperature, humidity and forward them to the server (host-PC) through base station. Server is responsible to evaluate, process, and visualize the data coming from the base station. For this purpose PostgreSQL database, Apache server and PHP language are used. First server record data coming from base station to the PostgreSQL database, then it process the data in the database depending on user requests, and last it displays the processed data with the help of Apache Web server and PHP. User can examine the processed data by means of a PC connected to the internet.

Programming of Sensors - Wireless Sensor Network

Initially as regards the programming of sensors and their function as individual sections of the wireless sensor network used the following tools:

- **MoteWorks**

This tool was created by the Crossbow company (the production company of the sensors we chose) and allows the development of applications with sensor nodes and is especially optimized for networks that operate with low power consumption using batteries. It is based on the open-source TinyOS operating system and offers reliable

ad-hoc mesh-type network, over-the-air programming capabilities and development tools for software applications.

- **MoteView**

This tool also created by Crossbow and is designed to offer the user simplicity as regards implementation and monitoring wireless sensors network. We used this specifically for sensors programming, which is done by using MoteConfig which we describe below [16].

- **MoteConfig**

Is a utility which is installed concurrently with the installation of MoteView. This is a Windows-based GUI that is used for programming the sensors. It provides a means for setting up and installing implemented XMesh / TinyOS applications up to sensors. Also, with its installation, XMesh applications are available for each sensor board and platform that is constructed from the Crossbow [15].

- **XServe**

The XServe serves as a gateway between wireless mesh networks and applications that interact with it. At its base, the XServe provides services for routing data to and from the grid network with higher level services for data processing. The XServe can be "joined" with wireless sensor networks using XMesh protocol via a base station which runs the XMeshBase offered by the Crossbow [14].

- ***Cygwi*n**

The Cygwin is a collection of tools of POSIX (Portable Operating System Interface for UNIX) made so as to work in the Microsoft Windows operating system. The software is distributed free of charge and operates as a simulator for the execution and operation of programs that is made to work with UNIX commands such as Xserve. The commands run on the command line. The use of these tools was necessary for the installation and use of the Xserve as the communication of sensors and a computer system is done through this [17].

Starting the implementation of the system, the first thing that had to deal with was the programming of sensors. As we explained in Chapter 1, the sensors must take at regular short intervals measurements concerning natural environmental phenomena. Also, it should be arranged in a network, addressing all of the requirements that we put, as for example in case of failure to maintain contact on the remaining nodes. Also from the design phase, we chose the XMesh is our Web protocol (Chapter 5).

Sensors scheduling separated into two parts based on mesh networking protocol – Xmesh, where the sensors are separated into two categories, sensor nodes and the base sensor node. So, by making use of MoteView and MoteConfig in particular, programmed the sensors that will constitute the sensors network with one of the pre-compiled Micaz XMesh Applications of Crossbow and specifically the

XMTS400_xxx_ .exe, which corresponds to the sensor board we use i.e. MTS400. Then, we programmed also the base node with a pre-compiled Micaz XMesh Applications namely XMeshBase_xxx_.exe. It should be noted that there is the option to install either the low power or high power application in sensors. In figure 5-1 (Chapter 5) we see basic parameters on the performance of Xmesh.

Since we were able to install and to program the sensors with these applications, the sensors can now be self-organized to mesh- type network, take measurements, forward them to the base node and maintain the state of the network and hence their successful interface. But to start the networking of sensors and general the operation of the sensors must start the XServe, which is described above.

As a first step, therefore, run applications using the MoteView, observing that the sensors self-organized actually to a network, receiving measurements and react to changes of network topology. What we then wanted to achieve is to commence the XServe from our own system and to achieve the same successful operation that succeed by making use of MoteView. Now we have placed four nodes in four different room and access the data through a GUI interface provide by crossbow MoteView. During this process the database will also be generated. Figure 7.2 shows the screen shot of the output in MoteView.

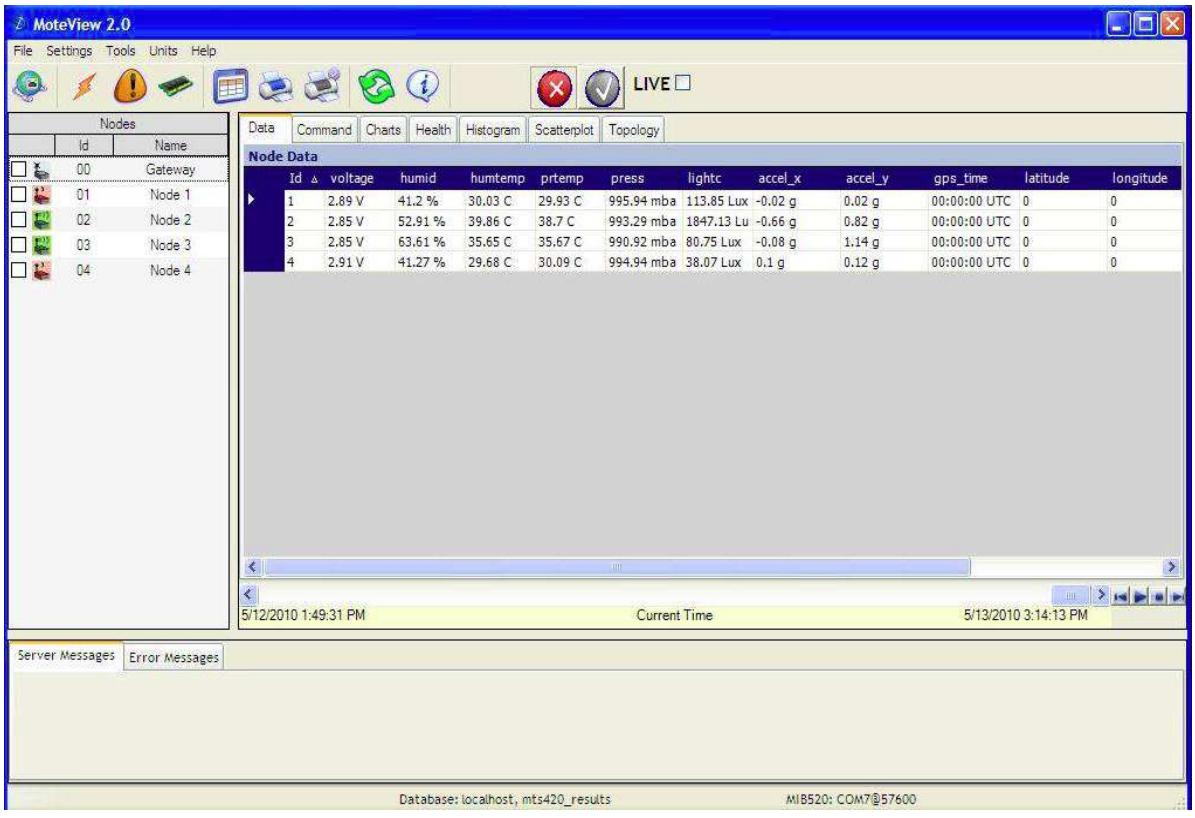


Figure 7-2: Results showing through MoteView

Database

Using the MoteView we can only view the parameters in our base station. But the data can't be accessed from other computers located in different location. To do this we have designed a web application through which the data can be accessed by other computers over LAN or Internet. The data accessed by the nodes are stored in the base station. The database used here is PostgreSQL. The database is accessed by our application and displayed as per the request. We can process the data in remote location as per our need and can monitor the WSN.

- **PostgreSQL**

PostgreSQL, often simply "Postgres", is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards-compliance. As a database server, its primary function is to store data, securely and supporting best practices, and retrieve it later, as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). It can handle workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users. Recent versions also provide replication of the database itself for availability and scalability.

PostgreSQL implements the majority of the SQL:2011 standard, is ACID-compliant and transactional (including most DDL statements) avoiding locking issues using multiversion concurrency control (MVCC), provides immunity to dirty reads and full serializability; handles complex SQL queries using many indexing methods that are not available in other databases; has updateable views and materialized views, triggers, foreign keys; supports functions and stored procedures, and other expandability, and has a large number of extensions written by third parties.

In addition to the possibility of working with the major proprietary and open source databases, PostgreSQL supports migration from them, by its extensive standard SQL support and available migration tools. And if proprietary extensions had been used, by its extensibility that can emulate many through some built-in and third-party open source compatibility extensions, such as for Oracle.

PostgreSQL is cross-platform and runs on many operating systems including Linux, FreeBSD, Solaris, and Microsoft Windows. Mac OS X, starting with OS X 10.7 Lion, has the server as its standard default database in the server edition, and PostgreSQL client tools in the desktop edition. The vast majority of Linux distributions have it available in supplied packages [18].

As mentioned above for data management and storing them in the database used the following tool:

- **PgAdmin III**

It is the most popular deployment and management tool for PostgreSQL, can be used in many operating such as Linux, FreeBSD, OpenSUSE, Solaris, Mac OSX and Windows platforms to manage PostgreSQL 7.3 and above running on any platform. The PgAdmin III is designed to meet the needs of all users, by performing simple SQL queries to developing complex databases. The graphical interface of the tool supports all the features of PostgreSQL, supports all functions and makes it easy to manage the database [19].

The database PostgreSQL is having a table called mts400_result which was the table generated by the program. The table has fields like Result_time, nodeid, parent, voltage, humidity, humidity temperature, pressure temperature, pressure, accel_x, accel_y, hours, minutes, seconds, latitude degree, latitude minute longitude degree, longitude minute. But in our application we have accessed the field result_time, humidity, humidity temperature where the result_time have data type as date and humidity, humidity_temp are int.

Web Application

The management of the system and the access to the information collected by the network of users should be done via a web application. In order to have access in our WSN, a user should have access to the Internet. The original idea is to be able to monitor the wireless sensor network means a Web Site so that at any time we can intervene either in the settings of the sensors or to see data from everywhere and whenever we want. To achieve this we should combine the "Internet" programming languages i.e. html, php, css, javascript – jQuery, xml. With the use of languages above what we accomplished is to create a website through which we can see the graphs from the sensors, the data the last about 24 hours, to download whenever we want, the data which collect the nodes in excel format so that we can work out.

For the development of web application used the following tools:

- **PHP**

PHP is a server-side scripting language designed for web development but also is used as a general-purpose programming language. As of January 2013, PHP was installed on more than 240 million websites (39% of those sampled) and 2.1 million web servers. Originally created by Rasmus Lerdorf in 1994, the reference implementation of PHP (powered by the Zend Engine) is now produced by The PHP Group. While PHP originally stood for *Personal Home Page*, it now stands for *PHP: Hypertext Preprocessor*, which is a recursive backronym.

PHP code can be simply mixed with HTML code, or it can be used in combination with various templating engines and web frameworks. PHP code is usually processed by a PHP interpreter, which is usually implemented as a web server's native module or a Common Gateway Interface (CGI) executable. After the PHP code is interpreted and executed, the web server sends resulting output to its client, usually in form of a part of the generated web page – for example, PHP code can generate a web page's HTML code, an image, or some other data. PHP has also evolved to include a command-line interface (CLI) capability and can be used in standalone graphical applications.
The canonical PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge [20].

- **HTML**

HTML or HyperText Markup Language is the standard markup language used to create Web pages. HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>). HTML tags most commonly come in pairs like <h1> and </h1>, although some tags represent empty elements and so are unpaired, for example <img>. The first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags).
A Web browser can read HTML files and compose them into visible or audible Web pages. The browser does not display the HTML tags, but uses them to interpret the content of the page. HTML describes the structure of a Website semantically along with cues for presentation, making it a markup language, rather than a programming language.
HTML elements form the building blocks of all Websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML Web pages [21].

- **XML**

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. It is defined by the W3C's XML 1.0 Specification and by several other related specifications, all of which are free open standards.
The design goals of XML emphasize simplicity, generality and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures such as those used in web services.

Several schema systems exist to aid in the definition of XML-based languages, while many application programming interfaces (APIs) have been developed to aid the processing of XML data [22].

- **CSS**

The CSS (Cascading Style Sheets-Cascading style sheets) is a computer language that belongs to the category of style sheet language used to control the appearance of a document written in a markup language. Used i.e. for controlling the appearance of a document written in HTML and XHTML languages, i.e. to control the appearance of a Web page and a site in General. The CSS is a computer language designed to develop stylistically a website i.e. to form more features, colors, align, and gives more features compared to html. For a beautiful and well-designed website using CSS is a must [23].

- **AJAX**

Ajax (short for asynchronous JavaScript + XML) is a group of interrelated Web development techniques used on the client-side to create asynchronous Web applications. With Ajax, Web applications can send data to and retrieve from a server asynchronously (in the background) without interfering with the display and behavior of the existing page. Data can be retrieved using the XMLHttpRequest object. Despite the name, the use of XML is not required, and the requests do not need to be asynchronous.
Ajax is not a single technology, but a group of technologies. HTML and CSS can be used in combination to mark up and style information. The DOM is accessed with JavaScript to dynamically display – and allow the user to interact with – the information presented. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads [24].

- **JavaScript-JQuery**

JavaScript (JS) is a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It is also used in server-side network programming with frameworks such as Node.js, game development and the creation of desktop and mobile applications.
JavaScript is classified as a prototype-based scripting language with dynamic typing and first-class functions. This mix of features makes it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

Despite some naming, syntactic, and standard library similarities, JavaScript and Java are otherwise unrelated and have very different semantics. The syntax of JavaScript is actually derived from C, while the semantics and design are influenced by Self and Scheme programming languages.
JavaScript is also used in environments that aren't web-based, such as PDF documents, site-specific browsers, and desktop widgets. Newer and faster JavaScript virtual machines (VMs) and platforms built upon them have also increased the popularity of JavaScript for server-side web applications. On the client side, JavaScript has been traditionally implemented as an interpreted language, but more recent browsers perform just-in-time compilation.
JavaScript has been standardized in the ECMAScript language specification [25].
JQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. Used by over 60% of the 10,000 most visited websites, JQuery is the most popular JavaScript library in use today. JQuery is free, open source software, licensed under the MIT License.
JQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. JQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, theme-able widgets. The modular approach to the JQuery library allows the creation of powerful dynamic web pages and web applications.
The set of JQuery core features—DOM element selections, traversal and manipulation—enabled by its selector engine (named "Sizzle" from v1.3), created a new "programming style", fusing algorithms and DOM data structures [26].

Web Services

The Web application is dynamically built, i.e. operations that we want and we need change without needing some intervention in the code. To make contact with the database essentially the application reads an XML, in which are stored all the information we need to connect (Host, Database name, Username, Password).

On the home page shows one of the most important information in our system. The current status of nodes values has as result to see directly if a Node is going to run out of battery or if a room soon has trouble with the humidity or temperature. These values are changed from the default values in the table of the administrator of the web application.

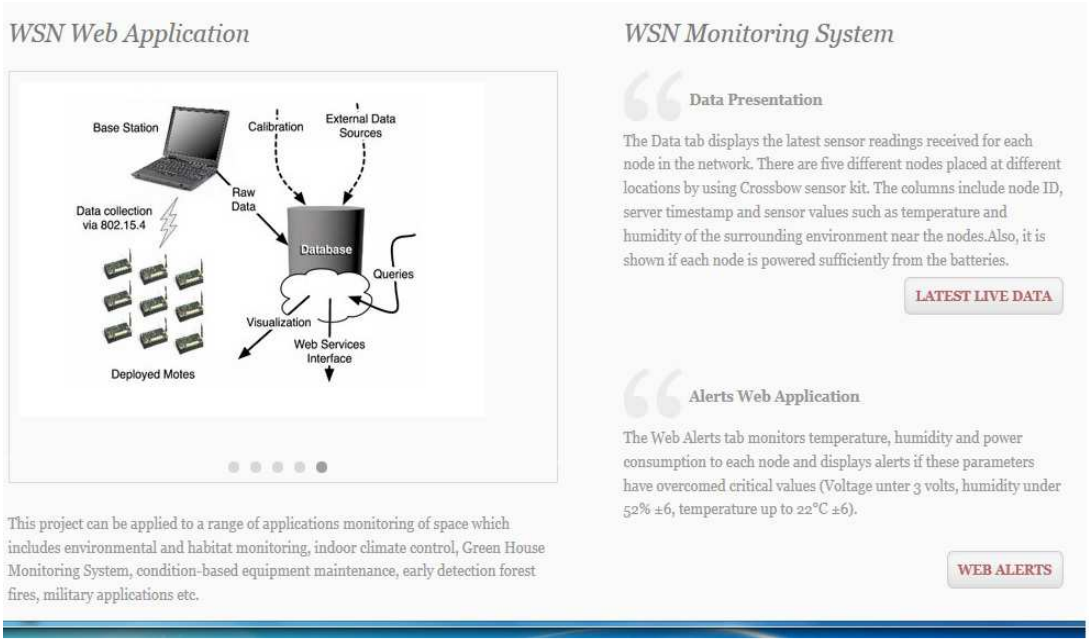


Figure 7-3: Homepage of our website

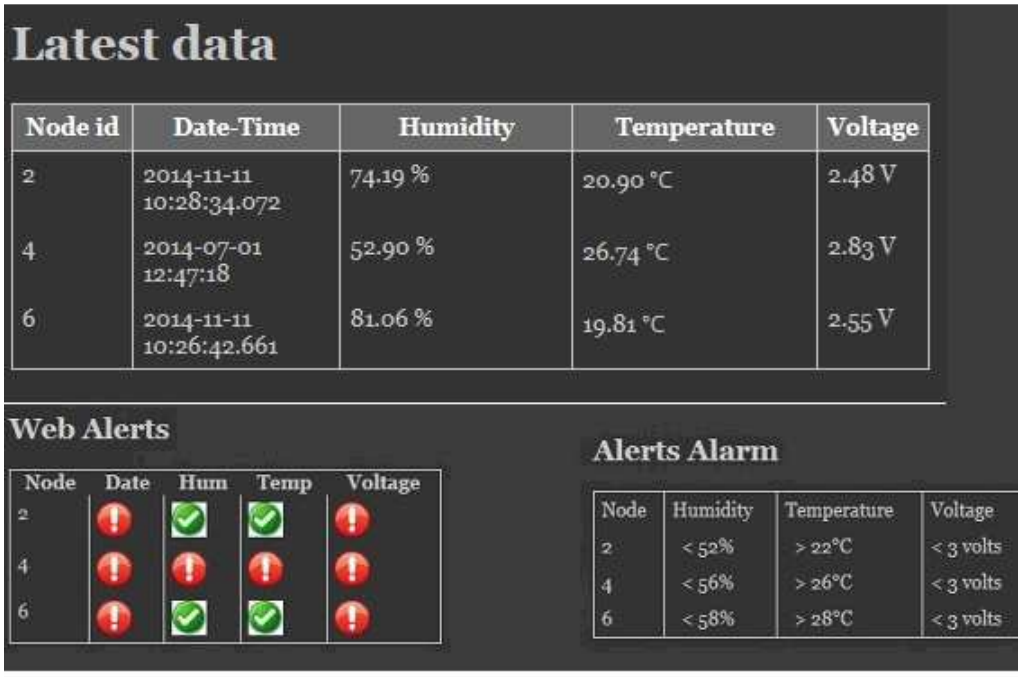


Figure 7-4: Real Time Status - Web Alerts

In ‘‘*WEB SERVICES/LIVE DATA’’* tab provides information and measurements for each node separately from the measurements which have recorded the last 24 hours. These values are humidity, temperature, battery status and of course the date and time of recording data.

Data from node 2

Date Time	Humidity	Temperature	Voltage
2014-11-11 10:28:34.072	66.61 %	20.90 °C	2.48 V
2014-07-01 12:49:15.187	66.61 %	29.55 °C	2.95 V
2014-07-01 12:45:58.656	66.61 %	29.44 °C	2.95 V
2014-07-01 12:42:58.406	66.61 %	29.41 °C	2.95 V
2014-07-01 12:39:58.687	66.61 %	29.42 °C	2.95 V
2014-07-01 12:33:57.375	66.61 %	29.42 °C	2.95 V
2014-07-01 12:30:57.703	66.61 %	29.38 °C	2.95 V
2014-07-01 12:27:56.578	66.61 %	29.36 °C	2.95 V
2014-07-01 12:24:56.343	66.61 %	29.36 °C	2.95 V
2014-07-01 12:21:56.937	66.61 %	29.37 °C	2.95 V
2014-07-01 12:18:56.203	66.61 %	29.36 °C	2.95 V
2014-07-01 12:15:56.281	66.61 %	29.36 °C	2.95 V
2014-07-01 12:12:56.359	66.61 %	29.32 °C	2.95 V
2014-07-01 12:09:56	66.61 %	29.31 °C	2.95 V
2014-07-01 12:06:54.859	66.61 %	29.29 °C	2.95 V
2014-07-01 12:03:55.453	66.61 %	29.28 °C	2.95 V
2014-07-01 12:00:54.531	66.61 %	29.26 °C	2.95 V
2014-07-01 11:57:55.578	66.61 %	29.28 °C	2.95 V
2014-07-01 11:54:55.015	66.61 %	29.28 °C	2.95 V
2014-07-01 11:51:55.046	66.61 %	29.25 °C	2.95 V

Figure 7-5: Recorded Data in the last 24 hours for Node 2

In ‘‘*WEB SERVICES/GRAPHICS’’* tab provides graphical representation of data in recent 24 hours humidity, temperature and battery status of each node in common graph with values- isolation feature. We have also the ability to save or print the graph.

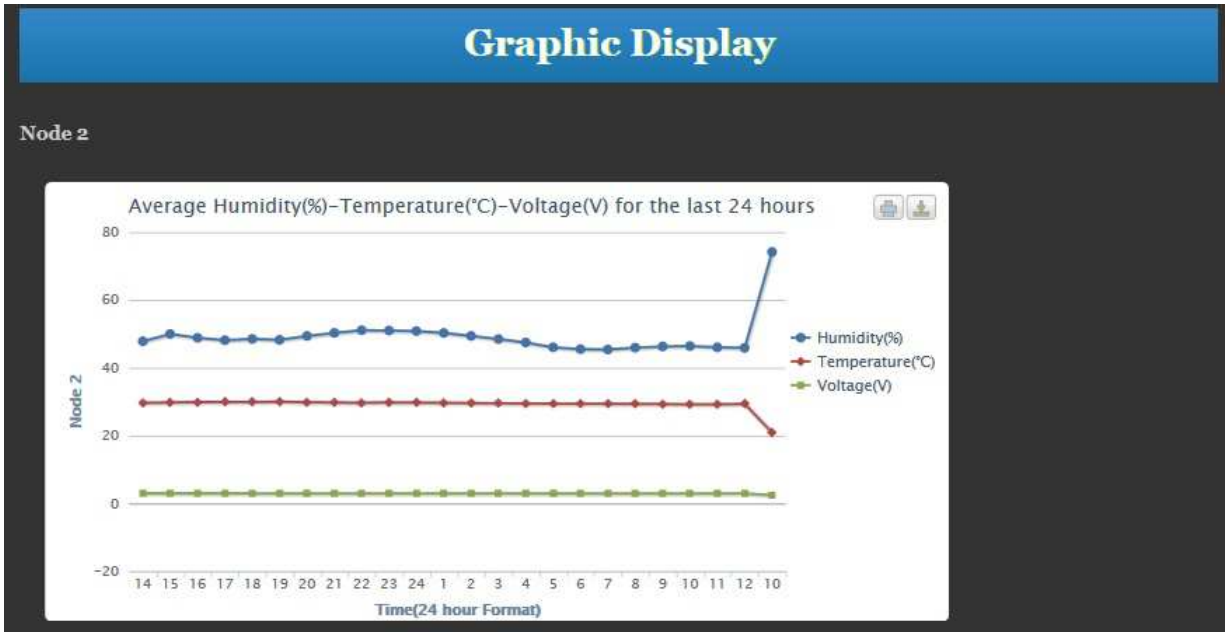


Figure 7-6: Graphical Representation of Data in the last 24 hours for Node 2

Finally, there is the possibility to save the data we want by selecting the format initially either in xls or csv and then regulate for which period we want to store our data. Once we make the appropriate settings press the DOWNLOAD button and it starts downloading.

Figure 7-7: Reports

Conclusions

Initially, We construct a full Web application and software package which operate independently and autonomously in relation to the programs provided by the Crossbow (Memsic now) company. This package is structured in such a way that it is easy to use and provides us with all the information that interest us immediately and in particular information which the original program of Crossbow did not have the ability to take or to process.

During the development of applications we concluded that there are several technical problems on the connection of two applications between them because they are based on different programming languages and environments. Also, we did the programming of network nodes with preset programs of Crossbow (MoteView 2.0) on Windows XP operating system because it was impossible to do on Windows 7. However, we noticed that after the programming of nodes, using the MoteView 2.1 program on Windows 7 the sensors nodes self-organized actually to a network and received normally measurements.

Finally, possible improvements of the platform would be the development of software to monitor and interact the user via a mobile device or tablet that use android operating system as well as the creating a new software for the nodes which will consume less energy.

Chapter 8
Related Works

Environment monitoring is a natural candidate for applying wireless sensor networks, since the physical variables that must to be monitored, e.g., temperature. They are usually distributed over large regions. Environmental monitoring applications can be broadly categorized into indoor and outdoor monitoring [28].

Indoor monitoring applications typically include buildings and offices monitoring. These applications involve sensing temperature, light, humidity, and air quality. Other important indoor applications may include fire and civil structures deformations detection. Outdoor monitoring applications include chemical hazardous detection, habitat monitoring, traffic monitoring, earthquake detection, volcano eruption, flooding detection and weather forecasting. Sensor nodes also have found their applicability in agriculture. Soil moisture and temperature monitoring is one of the most important application of WSNs in agriculture. When monitoring the environment, it is not sufficient to have only technological knowledge about WSN and their protocols. It is also necessary the knowledge about the ecosystem.

Several projects, with real implementations, had focused on environmental sensor networks; some of them are presented bellow.

GreatDuckIsland [29] was the first WSN implemented for habitat monitoring purposes. College of Atlantic and Berkeley University conducts field research on several remote islands. One of them, Great Duck Island (GDI) is located 15Km south of Mount Desert Island, Main. Studying the usage pattern of the nesting burrows when one or both parents alternate between incubation and feeding is the major objective of this project. A single hop hierarchical network comprises 32 nodes in the first phase and 120 in the last were set up at GDI. Berkeley Mica sensor nodes with TinyOS installed were used to measure temperature, humidity and atmosphere pressure and to detect the presence of the birds. Readings from sensor nodes are periodically sampled and relayed from the local sink node to base station on the island. The base station sends the data using a satellite link to a server connected to the Internet.

Sonoma Dust [30] is a WSN, constituted by 120 Mica2dot nodes that were installed on Sonoma County,California to monitor the redwood trees habitat conditions. Nodes with TinyOS were programmed to measure the environmental conditions (temperature, humidity and photo-synthetically active radiation) every 5 minutes and forwarded them through a multi-hop mesh network to a local base station. The data is sent from the base station to a computer located 70 Km away, through radio links. The nodes were programmed to run at a very low duty cycle to save energy.

A wireless sensor network was deployed to monitor eruptions at Tungurahua volcano, located in central Ecuador [31]. This single hop network is constituted by five sensor nodes where three of them are equipped with a specially constructed microphone to monitor infrasonic signals originated by volcanic eruptions. The data collected by the sensors are sent to a local sink and then relayed over radio links to a computer located 9 Km away. Mica2 nodes with TinyOS were used.

Measurement the microclimate in potato crops is the main goal of Lofar agro project [32]. The collected information will be used to improve the advice on how to combat phytophtora within a crop, based on the circumstances within each individual field. Phytophthora is a fungal disease in potatoes, their development and associated attack of the crop depends strongly on the climatologically conditions within the field. A total of 150 sensor nodes, similar to the Mica2 motes, were installed in a parcel for crop monitoring. Nodes are manually localized and their location registered on a map.

Sensor nodes are equipped with sensors for registering the temperature and relative humidity. In addition to the sensor nodes, the field is equipped with a weather station to register the luminosity, air pressure, precipitation, wind strength, and direction. The sensor nodes use TinyOS operating system. The data collected by the sensor nodes is sent over a multi-hop routing protocol to the local sink node (field gateway) and further transferred via Wi-Fi to Lofar gateway. The Lofar gateway is connected via wire to the Internet and data is uploaded to a Lofar server and further distributed to a couple of other servers.

In SECOAS project [33] a sensor network was deployed at Scroby sands off the coast of Great Yarmouth and its purpose will be to monitor the impact of a newly developed wind farm on coastal processes in the area. New sensor hardware, based on MCU PIC 18F452 was developed in this project and a new operating system, designated by kOS (kind-of operating system) was proposed to run on it. The sensor nodes are equipped with sensors for registering the pressure, turbidity, temperature and salinity. Sensor nodes, base stations on the sea and land stations, form the hierarchical and single hop network. Nodes transmit their data to the sea base stations, which will then transmit the data to the land station. Base stations are sensor nodes equipped with additional functionalities, more power supplies and larger communication range. The data accessed from the land station via Internet.

Foxhouse [34] get real time information about the habitat of foxes in a fox house. A wireless sensor network in the Foxhouse case has 14 nodes organized in two clusters. The network uses FFD nodes to relay data and RFD nodes for sensing. The sink node is connected to a personal computer where data is stored. CiNet boards compliant with IEEE 802.15.4 and based on ATmega 128L MCU are used on sensing nodes. The sensing nodes are equipped with temperature, humidity and light sensors.

In Sensorscope project [35], two networks were deployed. The first network was installed in Wannengrat to study environmental processes involving snow. The second network was installed on a glacier in the canton Valais, Switzerland, to measure air temperature, air humidity, surface temperature, wind direction and speed, precipitation and solar radiation. Seven nodes were used in the first deployment and sixteen nodes in the second. The similar solutions were used on both deployments. A Shockfish TinyNode platform was chosen and it is composed by a Texas Instruments MSP430 MCU and a Semtech XE1205 radio transceiver, operating in the 868 MHz band. The sensing nodes and the sink node uses TinyOS operating system. A multi-hop network is used to support communications between the sink node and the sensing nodes. Sensing stations regularly transmit collected data (e.g., wind speed and direction) to a sink, which, in turn, uses a gateway to relay the data to a server. GPRS, Wi-Fi or Ethernet technologies can be used to connect the sink node to the data base server, which can be installed remotely. Data is published on a real-time Google Maps-based web interface and on Microsoft's SensorMap website.

Bibliography

[1] Thesis : '' Supervision and management of crop area by using Wireless Sensor Networks (WSN)'' , Michalis Fragiadakis – George Vasilakis, Department of Informatics Engineering – Technological Educational Institute of Crete, 2012

[2] Special Scientific Work - Wireless Sensor Networks : "Development GUI with Matlab for taking measurements using the MTS400 / 420 board of the Crossbow" Xartoumbekis Giorgos, University of Patra, Electronics Laboratoy, 2010

[3] Thesis: '' Control Connectivity of Wireless Sensors Network'', Helen A. Papageorgakopoulou, University of Patra, School of Engineering, Department of Electrical and Computer Engineering, Systems and Control, 2009

[4] WSN: http://en.wikipedia.org/wiki/Wireless_sensor_network

[5] TinyOS - http://en.wikipedia.org/wiki/TinyOS

[6] IEEE standard 802.15.4-2003 - http://en.wikipedia.org/wiki/IEEE_802.15.4

[7] Zigbee specification – Zigbee Standards Organization – www.zigbee.org

[8] Δίκτυα Υπολογιστών – A.S Tanenbaum

[9] Protocols and Architectures for Wireless Sensor Networks – H. Karl, A. Willig

[10] http://en.wikipedia.org/wiki/Moving_average

[11] XMesh: XMesh_Users_Manual_7430-0108-01_C, Crossbow Technology

[12] Micaz – Mib520:
MPR-MIB_Series_Users_Manual_7430-0021-08_A, Crossbow Technology

[13]MTS400/420:
MTS-MDA_Series_Users_Manual_7430-0020-05_A, Crossbow Technology

[14] XServe: XServe_Users_Manual_7430-0111-01_D, Crossbow Technology

[15] MoteConfig:
MoteConfig_Users_Manual_7430-0112-01_A, Crossbow Technology

[16] MoteView:
MoteView_Users_Manual_7430-0008-05_A, Crossbow Technology

[17] Cygwin: http://www.cygwin.com/

[18] PostgreSQL: http://en.wikipedia.org/wiki/PostgreSQL

[19] PgAdmin III:
http://en.wikipedia.org/wiki/PostgreSQL#Database_administration

[20] PHP: http://en.wikipedia.org/wiki/PHP

[21] HTML: http://en.wikipedia.org/wiki/HTML

[22] XML: http://en.wikipedia.org/wiki/XML

[23] CSS: http://en.wikipedia.org/wiki/Cascading_Style_Sheets

[24] AJAX: http://en.wikipedia.org/wiki/Ajax_%28programming%29

[25] JavaScript: http://en.wikipedia.org/wiki/JavaScript

[26] JQuery: http://en.wikipedia.org/wiki/JQuery

[27] MEMSIC:
http://www.memsic.com/products/wireless-sensor-networks/development kits.html

[28] T. Arampatzis, J. Lygeros, and S. Manesis, "A survey of applications of wireless sensors and wireless sensor networks," in Proc. 13th Mediterranean Conf. Control Automation, Cyprus, Turkey, Jun. 2005, pp. 719-724.

[29] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. "An analysis of a large scale habitat monitoring application," In Proceedings of the Second ACM (SenSys), Baltimore, November 2004.

[30] G. Tolle. J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. W. Hong, "A Macroscope in the Redwoods," In: Proceedings of the 3rd ACM International Conference on Embedded Networked Sensor Systems (SENSYS), ACM Press, San Diego, CA, USA, pp. 51-63, November 2005.

[31] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a Wireless Sensor Network on an Active Volcano," IEEE Internet Computing, pp. 18-25, March/April, 2006.

[32] Lofar project, http://www.lofar.org/p/Agriculture.htm [July 2010].

[33] M. Britlon and L. Sacks, "The SECOAS project: Development of a self organizing. Wireless sensor network for environmental monitoring". 2nd International Work5hop on Sensor and Actor Network Protocols and Applications, August 2004.

[34] I. Hakala, M. Tikkakoski, and I. Kivela , "Wireless sensor network in environmental monitoring – case foxhouse," in Proceedings of the Second International Conference on Sensor Technologies and Applications (SENSORCOMM 2008), Cap Esterel, France, August 25-31 2008.

[35] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "SensorScope: Out-of-the Box Environmental Monitoring," Proceedings of the 7th international conference on Information processing in sensor networks, pp. 332-343, April 22-24, 2008.

[36] Contiki: http://en.wikipedia.org/wiki/Contiki

[37] Master Thesis: ''Application of Wireless Sensor Networks for Environmental Monitoring and Development of an Energy Efficient Cluster Based Routing '', Rohit Vaish, Department of Electrical Engineering - National Institute of Technology Rourkela 2008-2009

[38] ZigBee: http://en.wikipedia.org/wiki/ZigBee

Appendix:

WEB INTERFACE

Contact Database with Web Application:

To make the communication with the Database essentially the application reads an XML, which is stored all the information we need to connect (Host, Database name, Username, Password).

XML file:

```
<configuration>
<config>
<host> “host server” </host>
<dbname> “Database name” </dbname>
<user> “database user” </user>
<password> “database password” </password>
</config>
</configuration>
```

The PHP code to connect to the base is:

```
<?php
$dbh = pg_connect("host=$host dbname=$dbname user=$user
password=$password");
if (!$dbh) {
die("Error in connection: " . pg_last_error());
} ?>
```

In order to be able to read the above XML, we should perform the following php code:

```
<?php
$doc = new DOMDocument();
$doc->load( '../xml/dbconfig.xml' );
$configuration = $doc->getElementsByTagName( "config" );
foreach( $configuration as $config )
{
$hosts = $config->getElementsByTagName( "host" );
$host = $hosts->item(0)->nodeValue;
$dbnames = $config->getElementsByTagName( "dbname" );
$dbname = $dbnames->item(0)->nodeValue;
$users = $config->getElementsByTagName( "user" );
```

```
$user = $users->item(0)->nodeValue;
$passwords = $config->getElementsByTagName( "password" );
$password = $passwords->item(0)->nodeValue;
}
?>
```

Web Alerts Values:

Default alerts values in our web site are presented below:

```
<!--For all nodes: Voltage Alarm: 2,6 volts -->
        <!--Node 2: Humidity Alarm 52% - Temperature Alarm 22°C -->
        <tr>
          <td>2</td>
          <td>  < 52%</td>
          <td>  > 22&degC</td>
          <td> < 2.6 volts </td>
        </tr>

        <!--Node 4: Humidity Alarm 56% - Temperature Alarm 26°C -->
        <tr>
          <td>4</td>
          <td>  < 56%</td>
          <td>  > 26&degC</td>
          <td> < 2.6 volts </td>
        </tr>

        <!--Node 6: Humidity Alarm 58% - Temperature Alarm 28°C -->
        <tr>
          <td>6</td>
          <td>  < 58%</td>
          <td>  > 28&degC</td>
          <td> < 2.6 volts </td>
        </tr>

        <!--Node 3: Humidity Alarm 54% - Temperature Alarm 24°C -->
        <tr>
          <td>3</td>
          <td>  < 54%</td>
          <td>  > 24&degC</td>
          <td> < 2.6 volts </td>
        </tr>
```

If we would like to change these values for all nodes, it is able to do into AlertsValues.xml file, if we change the reference values of temperature, humidity and voltage:

```
<configuration>
  <config>
    <temp>22</temp> <!-- reference temperature -->
    <hum>52</hum>    <!-- reference humidity -->
    <volt>2.6</volt>  <!-- reference voltage -->
  </config>
</configuration>
```

In addition, we are able to change the values of each node (2, 3, 4, 6) separately into WebAlerts.php file:

```
switch ($nod) {

                //temp1=22°C, hum1=52%
                case 2:
                   $tempx= $temp1;
                   $humx= $hum1;
                   break;

                //temp=24°C,hum=54
                case 3:
                   $tempx= $temp1+2;
                   $humx= $hum1+2;
                   break;

                //temp=26°C,hum=56
                case 4:
                   $tempx= $temp1+4;
                   $humx= $hum1+4;
                   break;

                //temp=28°C,hum=58
                case 6:
                   $tempx= $temp1+6;
                   $humx= $hum1+6;
                   break;
```

Graphic Display:

The graphic display of data per 24 hours is become from safemode.php file:

```
<?php
$select="to_char(result_time, 'HH24'),humid,humtemp,voltage";
$from="mts400_results";
$nodes=$nodexml;
$order="result_time";
```

```
$sql = "SELECT $select FROM $from WHERE $nodes order by $order DESC  LIMIT 1";
$result = pg_query($dbh, $sql);
if (!$result) {
 die("Error in SQL query: " . pg_last_error());
}
while ($row = pg_fetch_array($result)){
$t=$row[0];
}

$sql = "SELECT $select FROM $from WHERE $nodes order by $order DESC";
$result = pg_query($dbh, $sql);
if (!$result) {
 die("Error in SQL query: " . pg_last_error());
}
$count=0;
$sum=0;
$cnt=0;
$j=0;

$sumvl=0;////////////
$sumtemp=0;////////////
while ($row = pg_fetch_array($result)){

   if ($t==$row[0]){
      $sum=$sum + (-2.0468+(0.0367*$row[1])+(-1.2955*0.000001*$row[1]*$row[1]));
      $sumtemp=$sumtemp +-39.7+0.01*$row[2];
      $sumvl=$sumvl + 1252.352/$row[3];//////////////
      $count=$count +1;////////////////////////////
   }
   else if($t!=$row[0]){
      $j=$j+1;
      $echtime[$j]=$t;
      $echumtemp[$j]=number_format($sum/$count,2);///

      error_reporting(E_ERROR | E_PARSE);

      $echtemp[$j]=number_format($sumtemp/$count,2);///
      $echvl[$j]=number_format($sumvl/$count,2);////
      $cnt=$cnt+1;
      $t=$row[0];

      $sum=0;
      $count=0;
      $sumvl=0;//////
      $sumtemp=0;/////////
      if ($cnt==24)
         break;

      if ($echtime[$j]=='00')
```

```
        $echtime[$j]='24';
    }
}

 echo   "<h3 id='node".$nod."'>Node ".$nod."</h3><br/>";
 $prg="container".$nod
 ?>
<script type="text/javascript">
var chart;
$(document).ready(function() {
chart = new Highcharts.Chart({
chart: {
      renderTo: '<?php echo $prg;?>',
      defaultSeriesType: 'line',
      marginRight: 130,
      marginBottom: 40
},
title: {
      text: 'Average Humidity(%)-Temperature(°C)-Voltage(V) for the last 24 hours',
      x: -20 //center
},

xAxis: {
      title: {
            text: 'Time(24 hour Format)'
      },
      categories: [
        <?php
        for ($i=24;$i>=1;$i=$i-1){
          echo $echtime[$i].",";
        }
        for ($i=24;$i>=1;$i=$i-1){
          $echtime[$i]='';
        }
        ?>
      ]
},
yAxis: {
      title: {
            text: '<?php echo $lefty; ?>'
      },
      plotLines: [{
            value: 0,
            width: 1,
            color: '#808080'
      }]
},
tooltip: {
      formatter: function() {
      return '<b>'+ this.series.name +'</b><br/>'+
```

```
                this.x +': '+ this.y;
        }
},
legend: {
        layout: 'vertical',
        align: 'right',
        verticalAlign: 'top',
        x: 10,
        y: 100,
        borderWidth: 0
},
series: [
{
        name: 'Humidity(%)',
        data: [
                <?php
                for ($i=24;$i>=1;$i=$i-1){
                echo $echumtemp[$i].",";
                }
                for ($i=24;$i>=1;$i=$i-1){
                $echumtemp[$i]='';
                }
                ?>
              ]
},{
        name: 'Temperature(°C)',
        data: [
           <?php
                for ($i=24;$i>=1;$i=$i-1){
                echo $echtemp[$i].",";
                }
                for ($i=24;$i>=1;$i=$i-1){
                $echtemp[$i]='';
                }
                ?>

        ]
},{
        name: 'Voltage(V)',
        data: [
           <?php
                for ($i=24;$i>=1;$i=$i-1){
                echo $echvl[$i].",";
                }
                ?>

        ]
}
```

```
]
});


}
);
</script>
<div id="<?php echo $prg;?>" style="width: 720px; height:350px; margin: 0 auto"></div>
```

Interaction user of web site and web developer with email:

A user can send us a message through email, as it is shown below:

This is achieved with Send_Mail.php file:

```
$ToEmail = 'pfasoul@gmail.com';
$EmailSubject = 'Site contact form';
$mailheader = "From: ".$_POST["validemail"]."\r\n";
$mailheader .= "Reply-To: ".$_POST["validemail"]."\r\n";
$MESSAGE_BODY = "Name: ".$_POST["fullname"]."\r\n\n";
$MESSAGE_BODY .= "Email: ".$_POST["validemail"]."\r\n\n";
$MESSAGE_BODY .= "Comment: ".nl2br($_POST["message"])."\r\n";

mail($ToEmail, $EmailSubject, $MESSAGE_BODY, $mailheader) or die ("Failure");
```

Display Mesh Network

	result_time timestamp w	epoch integer	nodeid integer	parent integer	voltage integer	humid integer	humtemp integer	prtemp integer	press integer	taosch0 integer	taosch1 integer	accel_x integer	accel_y integer
737428	2015-02-10		6	0	411	1996	5355	27422	17683	65447	0	429	424
737429	2015-02-10		2	4	441	1730	5609	27579	19044	65484	0	419	448
737430	2015-02-10		4	0	458	1715	5575	31107	17560	65501	0	428	411
737431	2015-02-10		2	4	441	1732	5615	27584	19042	65485	0	411	446
737432	2015-02-10		4	0	458	1715	5576	31109	17559	65502	0	428	411
737433	2015-02-10		3	4	391	1311	5977	0	18932	65475	65434	416	413
737434	2015-02-10		6	0	411	1968	5338	27399	17688	65447	0	428	424
737435	2015-02-10		2	4	441	1726	5613	27583	19043	65485	0	411	446
737436	2015-02-10		4	0	458	1723	5577	31108	17559	65502	0	429	410
737437	2015-02-10		3	4	391	1288	5989	0	18926	65475	65434	416	413
737438	2015-02-10		2	4	441	1725	5610	27581	19043	65485	0	414	448
737439	2015-02-10		4	0	458	1726	5576	31108	17556	65502	0	429	411
737440	2015-02-10		3	4	391	1272	6002	0	18919	65475	0	415	413
737441	2015-02-10		6	0	411	1962	5338	27402	17688	65450	0	429	424
737442	2015-02-10		2	4	443	1719	5609	27578	19046	65485	0	411	445
737443	2015-02-10		4	0	458	1722	5575	31108	17559	65502	0	429	411
737444	2015-02-10		3	4	391	1260	6001	0	18925	65476	65435	416	413
737445	2015-02-10		6	0	411	1968	5338	27402	17688	65450	0	429	425
737446	2015-02-10		2	4	442	1716	5608	27576	19044	65485	0	414	448
737447	2015-02-10		4	0	458	1721	5575	31109	17559	65502	0	428	411
737448	2015-02-10		3	2	391	1250	6008	0	18928	65476	65435	416	413
737449	2015-02-10		2	4	441	1715	5606	27573	19043	65485	0	411	445

In this image, it is illustrated a depiction of our database and it is shown how our mesh network works. We observe that the node **3** sends his packets to base station through the node **4**, but when we change his topology the node **3** sends packets to base through the node **2**.