Ignacio Salan Padillo

Evaluation of Wireless Sensor Networks Technologies

Cranfield Health

MPhil Thesis

Cranfield University

Cranfield Health

MPhil Thesis

Academic Year 2008-2009

Ignacio Salan Padillo

Evaluation of Wireless Sensor Networks Technologies

Supervisor: Dr Conrad Bessant

September 2008

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Philosophy

TABLE OF CONTENTS

<u>**LIST OF FIGURES**</u>

<u>LIST OF TABLES</u>

I. ABSTRACT

Wireless sensor networks represent a new technology that has emerged from developments in ultra low power microcontrollers and sophisticated low cost wireless data devices. Their small size and power consumption allow a number of independent ‘nodes’ (known as Motes) to be distributed in the field, all capable of ad-hoc networking and multihop message transmission. New routing algorithms allow remote data to be passed reliably through the network to a final control point. This occurs within the constraints of low power RF transmissions in a congested 2.4GHz radio spectrum. Wireless sensor network nodes are suitable for applications requiring long term autonomous operation, away from mains power supplies, such as environmental or health monitoring. To achieve this, sophisticated power management techniques must be used, with the units remaining ‘asleep’ in ultra low power mode for long periods of time.

The main aim of this research described in this thesis is first to review the area and then to evaluate one of the current hardware platforms and the popular software used with it called TinyOS. Therefore this research uses a hardware platform designed from University of Berkeley, called the TmoteSky. Practical work has been carried out in different scenarios. Using Java tools running on a PC, and customized applications running on the Motes, data has been captured, together with information showing topology configuration and adaptive routing of the network and radio link quality information. Results show that the technology is promising for distributed data acquisition applications, although in time critical monitoring systems new power management schemes and networking protocols to improve latency in the system will be required.

1 INTRODUCTION AND LITERATURE REVIEW

A wireless sensor network is a technology that emerges as a consequence of the evolution of network technology along with microelectronics and micromechanical devices. It is a new concept, a view towards the future, a clear consequence of the new steps forward in the communications field. In few words a wireless sensor network, is a network that could contain from a couple to many small nodes with sensors attached and communications capabilities to transmit and receive information. The data acquired by these sensors is then transmitted along a network from one node to another until it is collected in a central unit, normally connected to a PC.

If the original expectations created around this new network concept come true, it will improve monitoring and control systems used nowadays in the environment, medical, industry, consumer, and military sectors. With the help of this technology such systems will now be capable of raising a huge amount of data and results which will be available for analysis in real time. This could lead to a new era of monitoring and controlling processes as they are occurring, something that it was not possible before in some cases without the intervention of persons and complex equipment. Another advantage in the use of wireless technology is the reduction in cost that cabling deployment has in current systems and also the possibility of performing measurements in inaccessible places.

The possible scenarios of applications of this new networks are various at this moment. Environmental monitoring could make use of this technology to control health parameters in the air (CO_2 levels), in water it would be useful to detect the presence of heavy metals and contaminants and the same for soil. A number of sensors deployed to control the presence of oil in soil around an oil refinery. A field equipped with this sensor network to monitor the conditions of the soil for agriculture purposes, a water reserve supported by these sensors could give information regarding to the health conditions of the water for public consumption.

The evaluation of this technology and the practical use of it will be necessary to outline its advantages and weaknesses compared to other alternatives. The results and conclusions of this report might help future researches in this topic. It will be a practical guide that will help others to set up similar systems and understand how this network operates.

The architecture will include hardware technical aspects and software, based on actual standards. It is significant and of relevance to consider the possibilities there are to implement such a system with the technology available now.

As mentioned earlier many authors define such networks with the term “ad-hoc”, this word comes from Latin and means literally “for this”, further meaning “for this purpose only”. In the communications field an ad-hoc network means, a local area network or other small network, especially one with wireless or temporary connections, in which some of the network devices are part of the network only for the duration of a communications session.

There is currently a wide range of different communication protocols around, some of them wireless, some others cable connected. Wireless protocols such as Bluetooth were designed to connect small devices close to the base station, targeted to computer appliances and short range radio transmission. Internet protocols (Cabled and Wireless based) are also very complex and need a huge amount of resources, to provide a great number of services. Therefore the research community along with the private sector and the organisations in charge of creating new standards have reached the conclusion that existing communications protocols cannot be applied to this new type of network.

1.1 REQUIREMENTS FOR A WIRELESS SENSOR NETWORK

Initial research into wireless network sensors was mainly motivated by military applications, with the Defence Advanced Research Projects Agency (DARPA), continuing to fund a number of prominent research projects (e.g., Smart Dust, Network of Embedded systems (NEST)).The type of applications considered by these projects led to a de facto definition of a wireless sensor network as a large-scale (thousands of nodes, covering large geographical areas), wireless, ad-hoc, multihop, un-partitioned network of homogenous, tiny (hardly noticeable), mostly immobile (once deployed), sensor nodes that would be randomly deployed in the area of interest, (Kay Römer et al. 2004). Further applications and projects do not always fit these requirements, but these are the basic constraints assumed by most developers.

To summarize the aims that a node and a network of this nature have to achieve, these are ideally the main characteristics:

» Low power consumption.

» Low cost nodes that use cheap and commonly available batteries.

» Small physical size to facilitate deployment.

» Compliance to standards and regulations.

» Single design for international markets.

» Ability to maintain time synchronization with other nodes and hopping messages to destination.

» Operate over wide temperature ranges especially for military applications or high temperature scenarios (i.e. a desert zones).

» Fault tolerant.

1.2 ARCHITECTURAL ISSUES

This section summarizes the solutions proposed so far by different organizations to cope with such a problem of designing the architecture that best fits all the requirements exposed previously. The following picture shows a generic architecture.

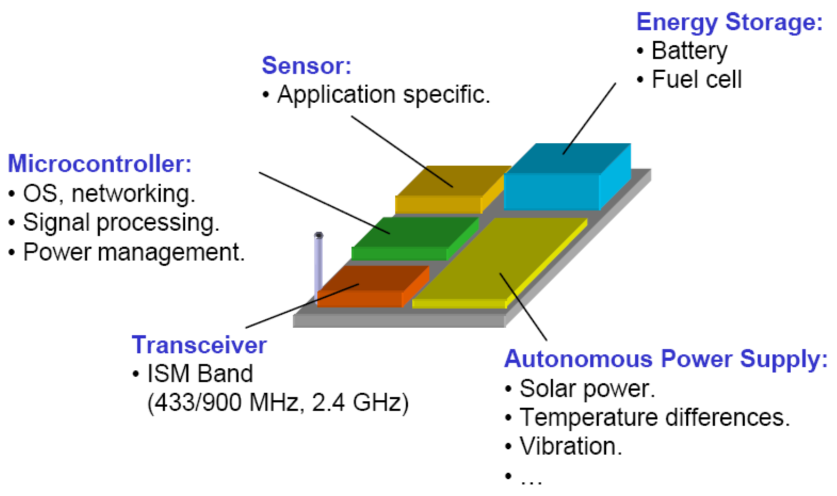


Figure 1. *The Architecture of a single node.*

Looking at Figure 1 where the generic node architecture is shown, it will include as main components: a microcontroller, a transceiver and a transducer. The development of microcontrollers make it possible to achieve low power consumption, reduced size while still allowing powerful computation to process signals and to control communications. One of the main attractions in the use of a current market microcontroller is the ability they have of entering into a state called “sleep mode” to save energy when no events occur. A transceiver is a transmitter/receiver in a single chip; this has been also a recent development in this field. The transducer, which is not shown in previous figure, will be the system in charge of processing the signals coming from the sensors.

1.2.1 Frequency of Radio Operation, Regulations and Range

An important aspect of such a system is the frequency band that will be assigned for these applications, as any radiating system has to run under proper regulations and standard that can vary depending on the country. For example in the US it is the Federal Commission of Communications (FCC) for commercial applications or OSM for defence applications. The organizations in charge of the Spectral Allocation, worldwide spectrum is controlled by International Telecommunication Union-Radio (ITU-R), and European Telecommunication Standard Institute (ETSI) for Europe. These organizations base their regulations most of the times on the standards published by the Institute of Electrical and Electronics Engineers (IEEE), the actual standard recommended for this system by this institute is the IEEE 802.15.4. The frequency of operation accorded worldwide for a wireless network system is 2.4 GHz, which belongs to the Industrial, Scientific and Medical (ISM) frequency band.

The industrial, scientific, and medical (ISM) radio bands were originally reserved internationally for non-commercial use of RF electromagnetic fields for industrial, scientific and medical purposes. The ISM bands are defined by the ITU-R in 5.138 and 5.150 of the Radio Regulations. Individual countries' use of the bands designated in these sections may differ due to variations in national radio regulations.

In recent years the ISM bands have also been used for license-free error-tolerant communications applications such as wireless LANs and Bluetooth, IEEE 802.11b/g wireless Ethernet also operates on the 2.4 GHz band:

The main benefit of this band is that no license is required, which reduces the cost of implementation and deployment of such a system. Disadvantages so far found are: the presence of many other services operating at the same band, security issues will be mandatory to avoid undesirable interferences and intrusion to the data being carried. The following graphs show some of the radio frequency propagation aspects of the frequency 2.4 GHz, (Theodore S. Rappaport, 1996).

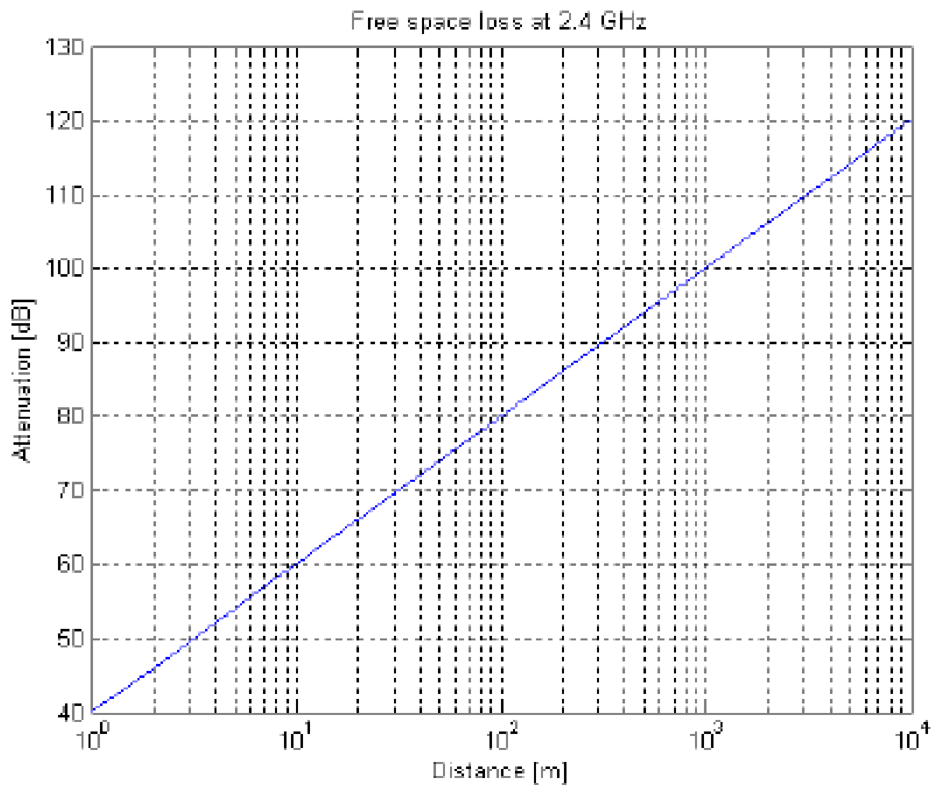


Figure 2. *Free space loss vs. distance at 2.4 GHz, when the first Fresnel zone is free of obstacles. (Theodore S. Rappaport, 1996)*

Free space losses versus distance shown by Figure 2, is an important parameter to take in account when designing the network. First Fresnel zone is defined as an area very close to the transmitter. A model to study the propagation in urban environments can be the Semi Deterministic COST 231-Walfisch-Ikegami. This approach takes into account heights of buildings, width of roads, building separation and road orientation. hb [m] is height of base station, hm [m] is height of client end, w [m] is width of roads, b [m] is building separation and φ [deg] is road orientation angle,

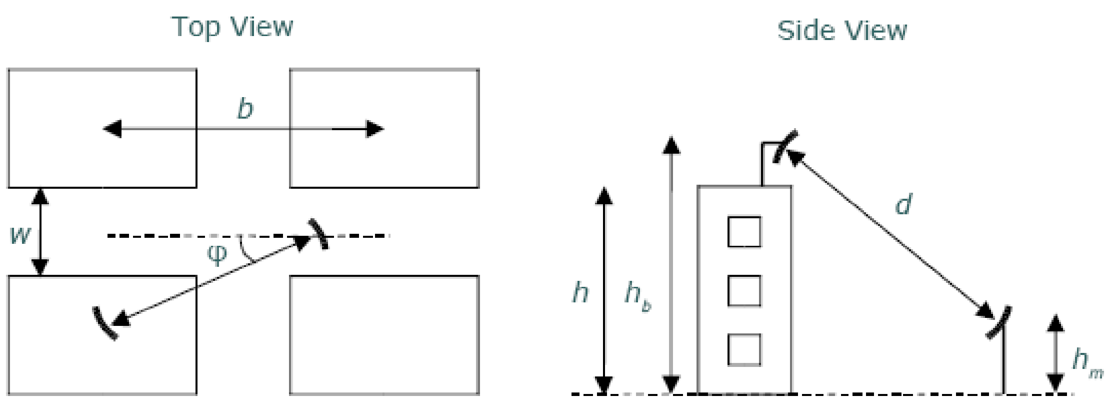


Figure 3. *Semi deterministic COST 231-Walfisch-Ikegami is suitable propagation model for built-up areas (Theodore S. Rappaport, 1996).*

To take account of losses resulting from floors and walls, the following equation can be used:

L (2.4GHz) = 40+20 log (d) + n f + m w (dB),

Where *n* and *m* are number of floors and walls between antennas and f and w are attenuation factors for floor and wall in dB. Typical attenuations through the different obstacles are presented in the following table.

Table 1. *The attenuation in indoors scenarios for 2.4 GHz.*

Obstacle	Attenuation [dB]
Floor	30
Brick wall with window	2
Office wall	6
Metal door in office wall	6
Cinder block wall	4
Metal door in brick wall	12.4
Brick wall next to metal door	3

In the next table regulations and limitations for this frequency are exposed along with the compliance document that describes in more detail these regulations.

Table 2 *Compliance table of power levels in different regions.*

Band	Location	Maximum Output Power	Maximum EIRP	Geographic Location	Compliance Document
ISM 2.4-2.483 GHz	Indoor and Outdoor	1000 mW	4000mW*	USA	FCC 15.247
ISM 2.4-2.483 GHz	Indoor and Outdoor	100 mW	NA	Europe	ETS 300-328

*For point to point applications

Being an unlicensed frequency band has the main advantage of reducing cost in the implementation of proprietary solutions, but it is necessary to design the system with care in order to avoid interferences due to other services operating in the same band of frequencies.

1.2.2 Network Topology

A communication network is composed of nodes, each of which has computing power and can transmit and receive messages over communications links, wireless or cabled. Therefore it is important to describe how the elements or nodes of a network are distributed physically. Depending on quality of service (QoS), the installation environment, economic considerations and the application, one of several basic network topologies may be used. The basic network topologies are shown in the next figure (F.L.Lewis *et al.*2004).

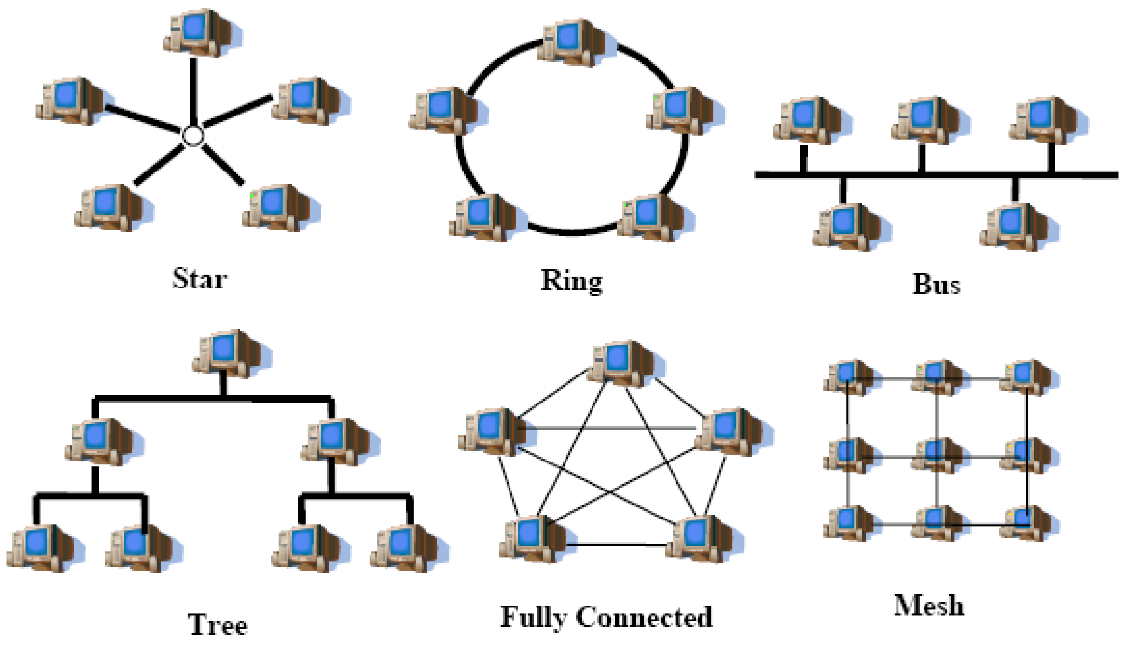


Figure 4. *Basic Network Topologies (F.L.Lewis et al.2004).*

Internet topology is a mixture of the star, tree and even fully connected diagram depending on the service and the area of the network. Fully connected networks suffer from problems of complexity; as additional nodes are added, the number of links increases exponentially. Therefore, for large networks, the routing problem is computationally intractable even with the availability of large amounts of computing power.

Mesh networks are regularly distributed networks that generally allow transmission only to a node's nearest neighbours. The nodes in these networks are generally identical, so that mesh nets are also referred to as peer-to-peer. Mesh nets can be good models for large-scale networks of wireless sensors that are distributed over a geographic region, e.g. personnel or vehicle security surveillance systems. Note that the regular structure reflects the communications topology; the actual geographic distribution of the nodes need not be a regular mesh. Since there are generally multiple routing paths between nodes, these nets are robust to failure of individual nodes or links. An advantage of mesh nets is that, although all nodes may be identical and have the same computing and transmission capabilities, certain nodes can be designated as ‘group leaders’ that take on additional functions. If a group leader is disabled, another node can then take over these duties.

All nodes of the star topology are connected to a single hub node. The hub requires greater message handling, routing, and decision-making capabilities than the other nodes. If a communication link is cut, it only affects one node. However, if the hub is incapacitated the network is destroyed. In the ring topology all nodes perform the same function and there is no leader node. Messages generally travel around the ring in a single direction.

In the bus topology, messages are broadcast on the bus to all nodes. Each node checks the destination address in the message header, and processes the messages addressed to it. The bus topology is passive in that each node simply listens for messages and is not responsible for retransmitting any messages. From the standard (IEEE 802.15.4, 2003) this figure shows what topologies are recommend in the design of a wireless sensor network which is include in what is called Low-Rate Wireless Personal Area Network (LR-WPAN).

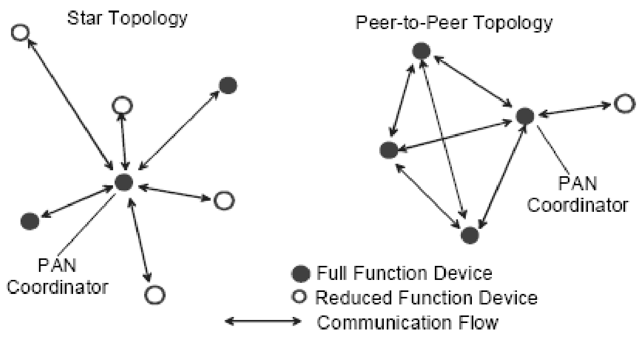


Figure 5. *Star and peer-to-peer topology examples.*

Applications that benefit from a star topology include home automation, Personal Computer (PC) peripherals, toys and games, and personal health care. The peer-to-peer topology also has a PAN coordinator; however, it differs from the star topology in that any device can communicate with any other device as long as they are in range for transmission. Peer-to-peer topology allows more complex network formations to be implemented, such as mesh networking topology. In the research work described later in this thesis, peer to peer topology is used but without the role of a coordinator.

1.2.3 Network Communications Protocols

This section summarizes the communications protocols proposed for this type of network, extracted from the literature consulted. In the context of data communication, a network protocol is a formal set of rules, conventions and data structure that governs how computers and other network devices exchange information over a network. In other words, protocol is a standard procedure and format that two data communication devices must understand, accept and use to be able to talk to each other.

Most communication protocols used in networks have a structure based on the Open Systems Interconnection model (OSI); this model was designed by the International Organization for Standardization (ISO) in 1984 under ISO document 7498.

The OSI model defines the communications process into 7 layers, which divides the tasks involved with moving information between networked devices into seven smaller, more manageable task groups. It is out of the scope of this document to describe in detail the functionality of each layer, but there is a brief explanation in the next table.

Table 3. *The OSI model and brief description of the different layers.*

Layer 7	**Application layer**	Responsible for managing the communication between applications. Provides standardized services to applications.
Layer 6	**Presentation layer**	Responsible for identifying the syntax of the data being transmitted. Provides code conversion, data compression and encryption services.
Layer 5	**Session layer**	Responsible for establishment and control of dialogues between software applications on different machines. Provide translation between names and address databases.
Layer 4	**Transport layer**	Responsible for end-to-end reliability from the transmitter to the receiver without any notice of intermediary equipment. Works in unit of messages.
Layer 3	**Network layer**	Responsible for routing of data unit called packets in the network, through intermediary equipments if necessary as far as the receiver. Works in unit of packets.
Layer 2	**Data Link layer**	Responsible for providing error-free transmission of hits on a physical interface. Provide mapping between Network Layer addresses and Data Link Layer addresses. Works in unit of frames.
Layer 1	**Physical layer**	Responsible for the electrical and mechanical interface for transmitting bits over a communication channel. Works in unit of bits.

The main task of a communication protocol is the routing of data packets along the network, where a node has to establish the route a message has to follow to reach the destination based upon parameters and conditions. It is this algorithm of deciding which route is best for the delivery of information that in most cases makes one protocol different from another. This function belongs to the network layer, the majority of Communications protocols for wireless sensor networks (WSNs) are based on the standard IEEE 802.15.4, that only defines the physical and data link layer, the upper layers are open to discussion, and different solutions that will be exposed in this document are being proposed.

These routing mechanisms have taken into consideration the inherent features of WSNs along with the application and architecture requirements. The task of finding and maintaining routes in WSNs is nontrivial since energy restrictions and sudden changes in node status (e.g., failure) cause frequent and unpredictable topological changes. To minimize energy consumption, routing techniques proposed in the literature for WSNs employ some well-known routing tactics as well as tactics special to WSNs, such as data aggregation and in-network processing, clustering, different node role assignment, and data-centric methods. Almost all of the routing protocols can be classified according to the network structure as flat, hierarchical, or location-based (Jamal N. Al-Karaki *et al.*2004). Furthermore, these protocols can be classified into multipath-based, query-based, negotiation-based, quality of service (QoS)-based, and coherent-based depending on the protocol operation.

In flat networks all nodes play the same role, while hierarchical protocols aim to cluster the nodes so that cluster heads can do some aggregation and reduction of data in order to save energy. In hierarchical-based routing, nodes will play different roles in the network. Location-based protocols utilize position information to relay the data to the desired regions rather than the whole network. I summarize the challenges and design issues that affect routing process in WSNs.

Node deployment in WSNs is application-dependent and can be either manual (deterministic) or randomized. In manual deployment, the sensors are manually placed and data is routed through predetermined paths. However, in random node deployment, the sensor nodes are scattered randomly, creating an ad hoc routing infrastructure. Communications between sensors is normally within a short range due to energy and bandwidth limitations; therefore it is most likely that a route will consist of multiple hops. Random deployments are of use when a network is just set up temporarily, as for example inside a tunnel where there is a fire, an evacuation situation that quickly needs different parameters to be evaluated and monitored during a certain period of time. Normally a manual deployment will be chosen when possible as it makes easier the control and maintenance of the network.

In a multihop WSN, each node plays a dual role as data sender and data router. The malfunctioning of some sensor nodes due to power failure can cause significant topological changes, and might require rerouting of packets and reorganization of the network. Data reporting can be classified as time-driven, event-driven, query-driven, or a hybrid of all these methods. In a time-driven delivery nodes transmit data at constant periodic times and enter sleep mode afterwards. In event-driven and query-driven methods nodes reacts to sudden and drastic changes in the value of sensed attribute or respond to a certain query generated by another node of the network. Normally in a WSN, all the nodes will have the same functionality but there is the possibility that in the same network some nodes have a different role. This network configuration makes necessary the presence of cluster heads to control transmissions towards the Base station or central unit. Since sensor nodes may generate significant redundant data, similar packets from multiple nodes can be aggregated to reduce the number of transmissions. Signal processing techniques can be used in a node to improve the quality of the signal received and has to pass to the next node in the route.

When sensor nodes are static, it is preferable to have table-driven routing protocols rather than reactive protocols. A significant amount of energy is used in route discovery. The following figures show the classification described (Jamal N. Al-Karaki *et al*.2004).

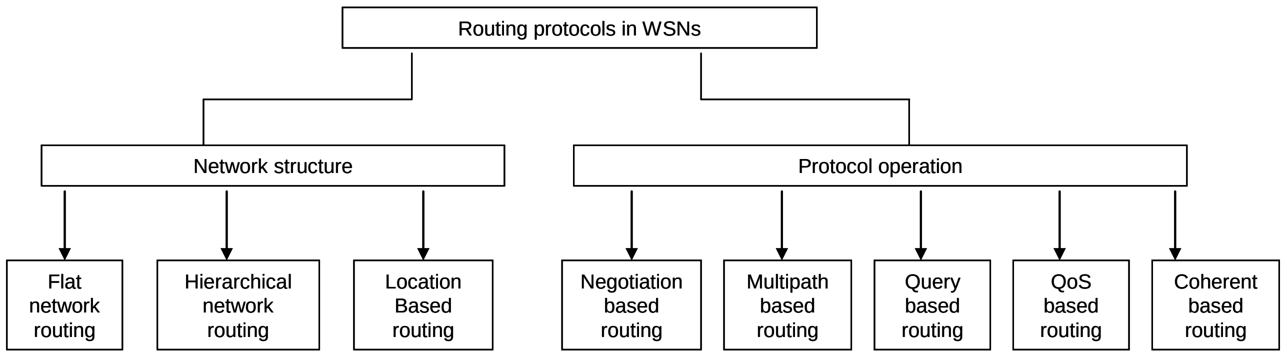
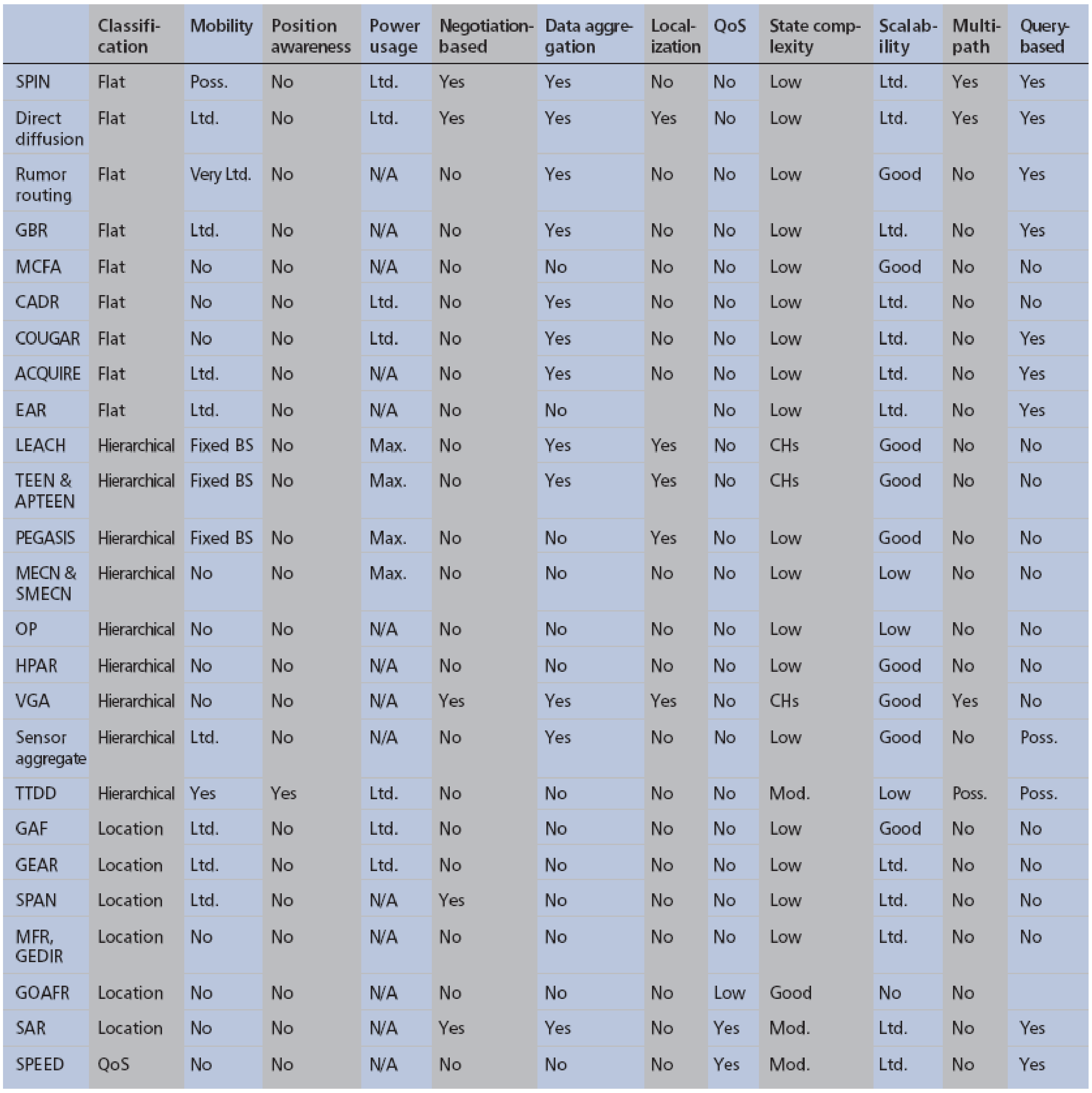


Figure 6. *Routing protocols in WSNs: a taxonomy (Jamal N. Al-Karaki et al.2004).*

Table 4 *Hierarchical vs. flat topologies routing (Jamal N. Al-Karaki et al.2004).*

Hierarchical routing	Flat routing
Reservation-based scheduling	Contention-based scheduling
Collisions avoided	Collision overhead present
Reduced duty cycle due to periodic sleeping	Variable duty cycle by controlling sleep time of nodes
Data aggregation by clusterhead	Node on multihop path aggregates incoming data from neighbors
Simple but non-optimal routing	Routing can be made optimal but with an added complexity.
Requires global and local synchronization	Links formed on the fly without synchronization
Overhead of cluster formation throughout the network	Routes formed only in regions that have data for transmission
Lower latency as multiple hops network formed by cluster- heads always available	Latency in waking up intermediate nodes and setting up the multipath
Energy dissipation is uniform	Energy dissipation depends on traffic patterns
Energy dissipation cannot be controlled	Energy dissipation adapts to traffic pattern
Fair channel allocation	Fairness not guaranteed

Table 5. *Classification and comparison of routing protocols in wireless sensor networks (Jamal N. Al-Karaki et al.2004).*



	Classifi-cation	Mobility	Position awareness	Power usage	Negotiation-based	Data aggre-gation	Local-ization	QoS	State comp-lexity	Scalab-ility	Multi-path	Query-based
SPIN	Flat	Poss.	No	Ltd.	Yes	Yes	No	No	Low	Ltd.	Yes	Yes
Direct diffusion	Flat	Ltd.	No	Ltd.	Yes	Yes	Yes	No	Low	Ltd.	Yes	Yes
Rumor routing	Flat	Very Ltd.	No	N/A	No	Yes	No	No	Low	Good	No	Yes
GBR	Flat	Ltd.	No	N/A	No	Yes	No	No	Low	Ltd.	No	Yes
MCFA	Flat	No	No	N/A	No	No	No	No	Low	Good	No	No
CADR	Flat	No	No	Ltd.	No	Yes	No	No	Low	Ltd.	No	No
COUGAR	Flat	No	No	Ltd.	No	Yes	No	No	Low	Ltd.	No	Yes
ACQUIRE	Flat	Ltd.	No	N/A	No	Yes	No	No	Low	Ltd.	No	Yes
EAR	Flat	Ltd.	No	N/A	No	No		No	Low	Ltd.	No	Yes
LEACH	Hierarchical	Fixed BS	No	Max.	No	Yes	Yes	No	CHs	Good	No	No
TEEN & APTEEN	Hierarchical	Fixed BS	No	Max.	No	Yes	Yes	No	CHs	Good	No	No
PEGASIS	Hierarchical	Fixed BS	No	Max.	No	No	Yes	No	Low	Good	No	No
MECN & SMECN	Hierarchical	No	No	Max.	No	No	No	No	Low	Low	No	No
OP	Hierarchical	No	No	N/A	No	No	No	No	Low	Low	No	No
HPAR	Hierarchical	No	No	N/A	No	No	No	No	Low	Good	No	No
VGA	Hierarchical	No	No	N/A	Yes	Yes	Yes	No	CHs	Good	Yes	No
Sensor aggregate	Hierarchical	Ltd.	No	N/A	No	Yes	No	No	Low	Good	No	Poss.
TTDD	Hierarchical	Yes	Yes	Ltd.	No	No	No	No	Mod.	Low	Poss.	Poss.
GAF	Location	Ltd.	No	Ltd.	No	No	No	No	Low	Good	No	No
GEAR	Location	Ltd.	No	Ltd.	No	No	No	No	Low	Ltd.	No	No
SPAN	Location	Ltd.	No	N/A	Yes	No	No	No	Low	Ltd.	No	No
MFR, GEDIR	Location	No	No	N/A	No	No	No	No	Low	Ltd.	No	No
GOAFR	Location	No	No	N/A	No	No	No	Low	Good	No	No	
SAR	Location	No	No	N/A	Yes	Yes	No	Yes	Mod.	Ltd.	No	Yes
SPEED	QoS	No	No	N/A	No	No	No	Yes	Mod.	Ltd.	No	Yes

The previous table presented in (Jamal N. Al-Karaki *et al*.2004) is a survey of WSNs communications protocols actually in use and under study; all of them present a different approach to the problem. I will describe in more detail some of them as the full detailed description can be consulted in the reference. The ones described are then considered more significant to this research.

(J.Kulik et al. 2002) proposed a family of adaptive protocols called Sensor Protocols for Information via Negotiation (SPIN) that disseminate all the information at each node to every node in the network assuming that all nodes in the network are potential BSs (base stations). This enables a user to query any node and get the required information immediately. An alternative could be: this technique is very useful where the nodes are in a fixed location that is precisely known by the user.

The Minimum Cost Forwarding Algorithm (MCFA) (Ye F. *et al* 2001), exploits the fact that the direction of routing is always known (i.e., toward the fixed external BS). Hence, a sensor node need not have a unique ID nor maintain a routing table. Instead, each node maintains the least cost estimate from itself to the BS. Each message to be forwarded by the sensor node is broadcast to its neighbours. When a node receives the message, it checks if it is on the least cost path between the source sensor node and the BS. If this is the case, it rebroadcasts the message to its neighbours. This process repeats until the BS is reached. This protocol might be very useful for this research as it is not intended to deploy the nodes randomly instead at fix locations after exhaustive study of scenario conditions.

The LEACH protocol introduces a hierarchical clustering algorithm for sensor networks, called Low Energy Adaptive Clustering Hierarchy (LEACH). LEACH is a cluster-based protocol, which includes distributed cluster formation. LEACH randomly selects a few sensor nodes as cluster heads (CHs) and rotates this role to evenly distribute the energy load among the sensors in the network. In LEACH, the CH nodes compress data arriving from nodes that belong to the respective cluster, and send an aggregated packet to the BS in order to reduce the amount of information that must be transmitted to the BS. This protocol is most appropriate when there is a need for constant monitoring by the sensor network. A user may not need all the data immediately. Hence, periodic data transmissions are unnecessary, and may drain the limited energy of the sensor nodes (M. Handy *et al*, 2002). Again this protocol is found very interesting as a powerful and smart technique to save power.

The protocol, called Power-Efficient Gathering in Sensor Information Systems (PEGASIS), is a near optimal chain-based protocol. The basic idea of the protocol is that in order to extend network lifetime, nodes need only communicate with their closest neighbours, and they take turns in communicating with the BS. When the round of all nodes communicating with the BS ends, a new round starts, and so on. This reduces the power required to transmit data per round as the power draining is spread uniformly over all nodes (Lindsey, S *et al,* 2002). This is another interesting communication protocol and set of techniques to maintain power consumption as minimum as possible.

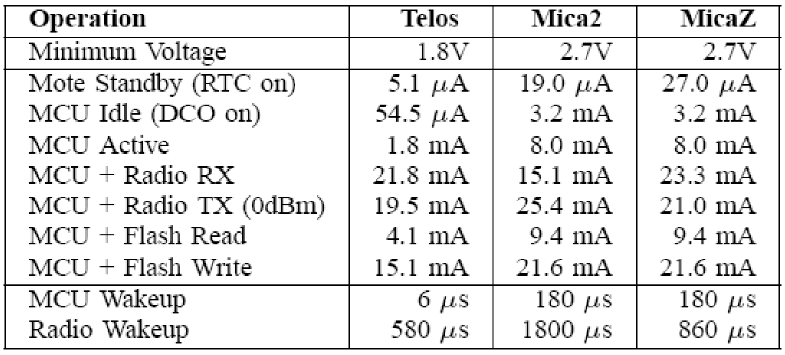
Another important issue for a communication protocol is the ability to avoid collisions between information packets along the network. When two packets are transmitted at the same time and collide, they become corrupted and must be discarded; retransmission might not be possible in some cases and also wastes too much energy. There is a classification of protocols based in this principle. Scheduled protocols are those where nodes access the channel in order to transmit information, at assigned fixed time slots, frequency or code. The Leach protocol uses Time Division Multiple Access (TDMA), where the channel is divided into time slots and these are assigned by the master to the nodes. There is also a Frequency Division Multiple Access (FDMA) and Code Division Multiple Access (CDMA), instead of assigning time slots, frequencies and codes identify when a node has to transmit.

Unlike scheduled protocols, contention based protocols do not divide the channel into sub-channels or pre-allocate the channel for each node to use. Instead a common channel is shared by all nodes and it is allocated on-demand. A contention mechanism is employed to decide when a node has the right to access the channel at any moment. These protocols can scale more easily across changes in node density or traffic load, can be more flexible as topology changes, there is no requirement to form clusters, and peer to peer communications is directly supported. As an example of this technique is the Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA), this one also avoids the hidden terminal problem. Suppose nodes a, b and c can only hear from their immediate neighbours. When node *a* is sending to *b*, node *c* is not aware of this transmission, and its carrier sense still indicates that the medium is idle. If node *c* starts to transmit now, node *b* will receive collided packets from both *a* and *c* (C.S.Raghavendra *et al.* 2004). The basic mechanism of CSMA-CA is all nodes listen before they start transmitting, if the channel is busy when they try to access it they booked a later period of time to retransmit. All nodes in the same transmitting area must have different listening periods. All these techniques to avoid collisions are implemented in the data link layer, and they are chosen to suit as best as possible the needs of the communication protocol used by a network.

1.2.4 <u>Power Management</u>

This section will present the importance for this technology of a good power management scheme. It is one of the most crucial aspects, as these networks are predicted to be unattended and have long time life of operation. Assuming the previous definition it is obvious that the power source of a node of such a network can not be mains power. Therefore the use of batteries and energy scavenging methods seem to be the best option in most cases, along with a communication protocol and architecture design optimized to save as much power as possible. To give an indication of how much power is consumed in such a system, the following table is presented, based on different hardware platforms (Mark Hempstead *et al*. 2005). The amount of current is directly proportional to the power consumed.

Table 6. *Measured Current consumption on different hardware platforms (Mark Hempstead et al. 2005). It is clear that the Telos uses the least power*



Operation	Telos	Mica2	MicaZ
Minimum Voltage	1.8V	2.7V	2.7V
Mote Standby (RTC on)	5.1 μA	19.0 μA	27.0 μA
MCU Idle (DCO on)	54.5 μA	3.2 mA	3.2 mA
MCU Active	1.8 mA	8.0 mA	8.0 mA
MCU + Radio RX	21.8 mA	15.1 mA	23.3 mA
MCU + Radio TX (0dBm)	19.5 mA	25.4 mA	21.0 mA
MCU + Flash Read	4.1 mA	9.4 mA	9.4 mA
MCU + Flash Write	15.1 mA	21.6 mA	21.6 mA
MCU Wakeup	6 μs	180 μs	180 μs
Radio Wakeup	580 μs	1800 μs	860 μs

The following graph shows a comparison of the potential output versus lifetime for different power sources (Shad Roundy *et al.* 2003).

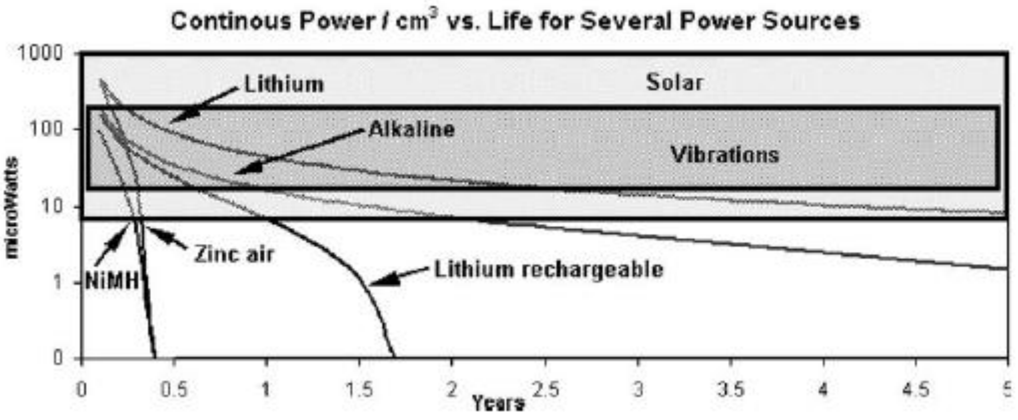


Figure 7. *Comparison of power from vibrations, solar, and various battery chemistries (Shad Roundy et al. 2003).*

The overall power scheme of a network would be a combination of different techniques applied in all the design stages. In some cases, it is possible for the node to obtain energy sufficient for its operation from its environment. Such methods are called “energy scavenging” and include extracting energy by photovoltaic, mechanical vibration and other means. There are also studies on how to obtain energy from temporal variations of air pressure and temperature (Edgar H. Callaway, 2003). Combining different power sources it is called hybrid power source. This concept in combination with some digital electronics to chose between one and another and recharge one of them at the same time might be an interesting power management technique to study.

These techniques rarely present the only power source but can be combined with usual power sources and extend considerably the life of the node. It is clear that an important stage in the design of such a network is to make a survey of possible energy sources in the environment and location of deployment. Microcontrollers used today normally have low power consumption levels and different states of operation. Tasks executed by a node can be totally separated and assigned to the different components without involving the use of the microcontroller, a transceiver can operate while the microcontroller is switched off or put to sleep and vice versa. According to the latest report from OFCOM about this technology energy saving is a key constraint for a bigger adoption of WSN in the industry. In next sections will be explained the different standards and solutions proposed by Industry and Research community.

1.3 IEEE 802.15.4

This is the standard that rules the working procedure of wireless sensor networks at the bottom layers of the network protocol. It specifies the Physical layer and the Data link layer also called the Medium Access Control (MAC) layer. Wireless Personal Area Networks (WPANs) are used to convey information over relatively short distances. Unlike wireless local area networks (WLANs), connections effected via WPANs involve little or no infrastructure. This feature allows small, power-efficient, inexpensive solutions to be implemented for a wide range of devices (IEEE 802.15.4).

A LR-WPAN is a simple, low-cost communication network that allows wireless connectivity in applications with limited power and relaxed throughput requirements. The main objectives of an LR-WPAN are ease of installation, reliable data transfer, short-range operation, extremely low cost, and a reasonable battery life, while maintaining a simple and flexible protocol. Some of the characteristics of an LR-WPAN are:

- Over-the-air data rates of 250 kb/s, 40 kb/s, and 20 kb/s
- Star or peer-to-peer operation
- Allocated 16 bit short or 64 bit extended addresses
- Allocation of guaranteed time slots (GTSs)
- Carrier sense multiple access with collision avoidance (CSMA-CA) channel access
- Fully acknowledged protocol for transfer reliability
- Low power consumption
- Energy detection (ED)
- Link quality indication (LQI)
- 16 channels in the 2450 MHz band, 10 channels in the 915 MHz band, and 1 channel in the 868 MHz band

Two different device types can participate in an LR-WPAN network; a Full-Function Device (FFD) and a Reduced-Function Device (RFD). The FFD can operate in three modes serving as a Personal Area Network (PAN) coordinator, a coordinator, or a device. An FFD can talk to RFDs or other FFDs, while an RFD can talk only to an FFD. An RFD is intended for applications that are extremely simple, such as a light switch or a passive infrared sensor; they do not have the need to send large amounts of data and may only associate with a single FFD at a time. Consequently, the RFD can be implemented using minimal resources and memory capacity (IEEE 802.15.4). The two possible network topologies have been described in figure 6.

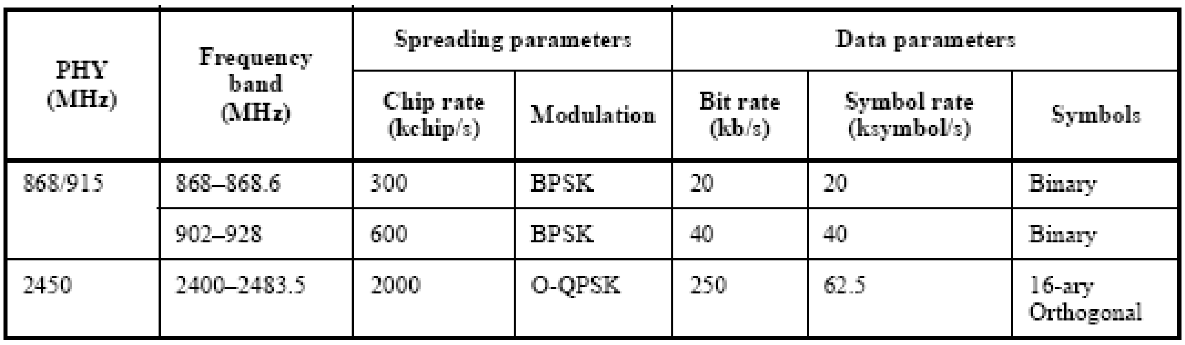
After an FFD is activated for the first time, it may establish its own network and become the PAN coordinator. All star networks operate independently from all other star networks currently in operation. This is achieved by choosing a PAN identifier, which is not currently used by any other network within the radio sphere of influence. Once the PAN identifier is chosen, the PAN coordinator can allow other devices to join its network; both FFDs and RFDs may join the network. Detailed procedure of joining the network can be consulted in the standard. In a peer-to-peer topology, each device is capable of communicating with any other device within its radio sphere of influence. One device will be nominated as the PAN coordinator, for instance, by virtue of being the first device to communicate on the channel. Further network structures can be constructed out of the peer-to-peer topology and may impose topological restrictions on the formation of the network.

The features of the Physical Layer (PHY) are activation and deactivation of the radio transceiver, ED, LQI, channel selection, Clear Channel Assessment (CCA), and transmitting as well as receiving packets across the physical medium. The radio shall operate at one of the following license-free bands:

– 868–868.6 MHz (e.g., Europe),
– 902–928 MHz (e.g., North America) or
– 2400–2483.5 MHz (worldwide).

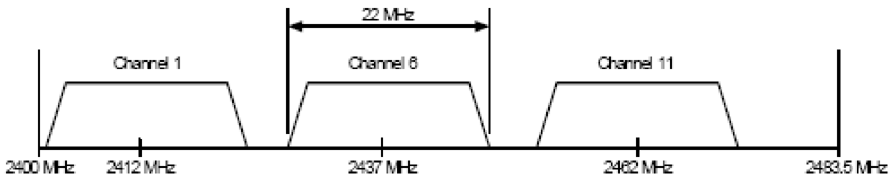
The MAC sublayer provides two services: the MAC data service and the MAC management service interfacing to the MAC sublayer management entity (MLME) Service Access Point (SAP) (known as MLME-SAP). The MAC data service enables the transmission and reception of MAC protocol data units (MPDUs) across the PHY data service. The features of the MAC sublayer are beacon management, channel access, GTS management, frame validation, acknowledged frame delivery, association, and disassociation. In addition, the MAC sublayer provides hooks for implementing application appropriate security mechanisms (IEEE 802.15.4).

Table 7. *Frequency bands and data rates (IEEE 802.15.4).*

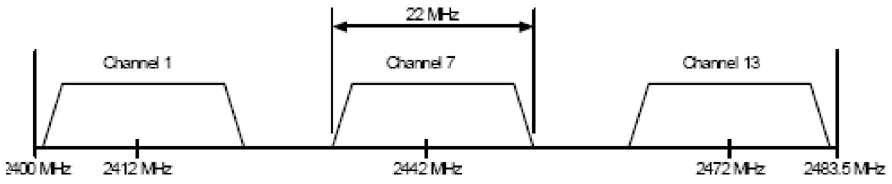


PHY (MHz)	Frequency band (MHz)	Spreading parameters		Data parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbol rate (ksymbol/s)	Symbols
868/915	868–868.6	300	BPSK	20	20	Binary
	902–928	600	BPSK	40	40	Binary
2450	2400–2483.5	2000	O-QPSK	250	62.5	16-ary Orthogonal

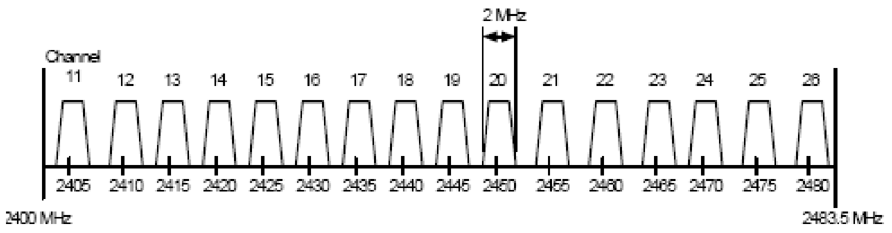
The standard is still under study and discussion to solve ambiguities, reduce complexities and cope with problems appearing with the deployment of such networks. It has also assigned different task groups to study other frequencies available in the ISM bands with better response to attenuations. It is a very important issue how this standard coexists with other services transmitting in the same frequency bands. There are several graphs showing the probability of packet information collision between other wireless services and WSNs at the end of the IEEE 802.15.4 document. The first conclusion one can extract looking at these graphs is probability decreases with distance between services. Next figure shows distribution of channels of other services and wireless sensor networks.



a) IEEE 802.11b North American channel selection (nonoverlapping)



b) IEEE 802.11b European channel selection (nonoverlapping)



c) IEEE 802.15.4 channel selection (2400 MHz PHY)

Figure 8 *IEEE 802.15.4 (2400 MHz PHY) and IEEE 802.11b channel selection (IEEE 802.15.4).*

Since the power is concentrated around the centre frequency, the in-band interference power is dependent on the offset between the centre frequencies of the IEEE 802.15.4 and the IEEE 802.11b. For example, if the centre frequency of the IEEE 802.15.4 is 2416 MHz and that of the IEEE 802.11b is 2418 MHz, then the centre frequency offset is 2 MHz. Then, the in-band interference power is about 17 percent of the total power of the IEEE 802.11b (Soo Young Shin, *et al*. 2005). Next figures extracted from the results of this article show the packet error rate (PER) in different situations and from simulated/empirical analysis.

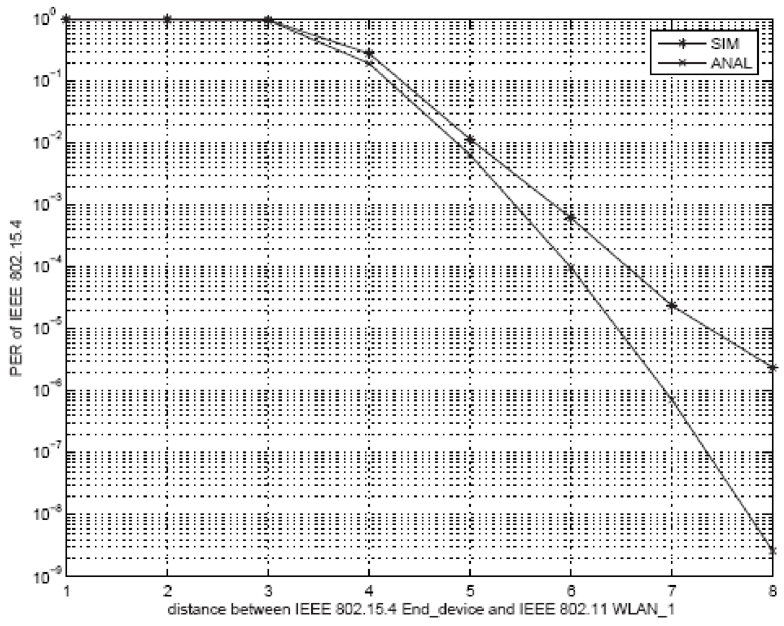


Figure 9. *PER of the IEEE 802.15.4 without considering the power spectral density of the IEEE 802.11b (Soo Young Shin, et al. 2005)*

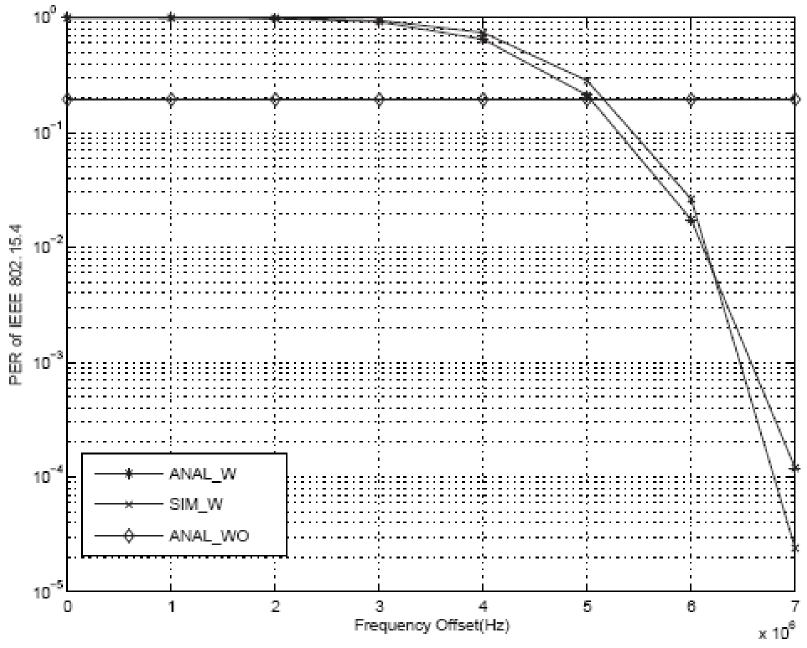


Figure 10. *PER of the IEEE 802.15.4 with the different center frequency offset to the IEEE 802.11b (Soo Young Shin, et al. 2005)*

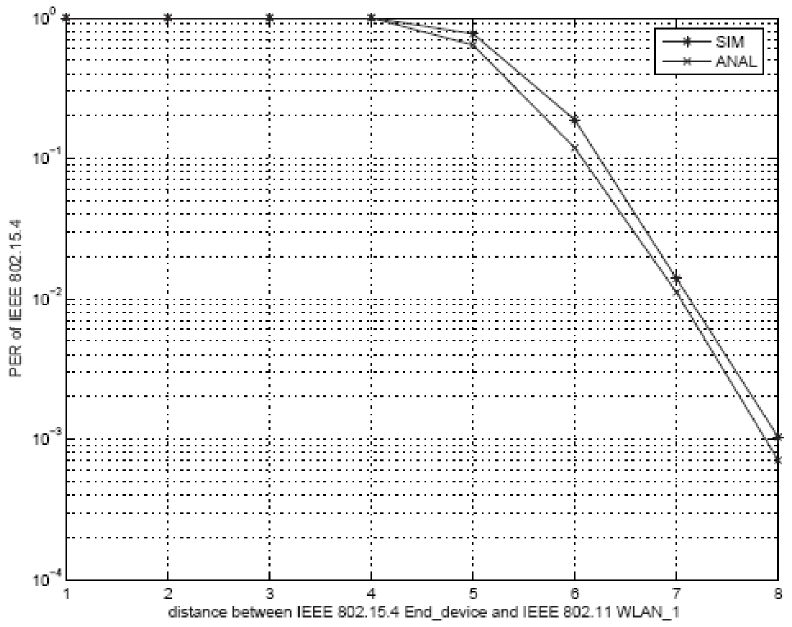


Figure 11. *PER of the IEEE 802.15.4 with considering the power spectral density of the IEEE 802.11b (Soo Young Shin, et al. 2005)*

If the distance between the IEEE 802.15.4 and 802.11b is longer than 8 m, the interference of the IEEE 802.11b does not affect the performance of the IEEE 802.15.4. If the frequency offset is larger than 7 MHz, the interference effect of the IEEE 802.11b is negligible to the performance of the IEEE 802.15.4. Therefore, three additional channels of the IEEE 802.15.4 such as 2420 MHz, 2445 MHz, and 2470 MHz can be used for the coexistence channels under the interference of the IEEE 802.11b (Soo Young Shin, *et al*. 2005).

This is a topic well opened to discussion and further practical experimentation as only analytical studies and simulation have been carried out, more practical experiments and the real deployment and observation of these networks in such scenarios coexisting with other services will be necessary.

1.4 ZIGBEE

Zigbee is a standard proposed by the industry to the problem of designing a wireless network sensor that achieves all the objectives set by the requirements of such a network. The ZigBee Alliance is developing a standard based on very low-cost, very low power consumption, two-way wireless communications. Solutions adopting the ZigBee standard are hoped to be embedded in consumer electronics, home and building automation, industrial controls, PC peripherals, medical sensor applications, toys and games (ZigBee Alliance, 2005). The name ZigBee is said to come from the domestic honeybee which uses a zig-zag type of dance to communicate important information to other hive members. This communication dance (the "ZigBee Principle") is what engineers are trying to emulate with this protocol, a bunch of separate and simple organisms that join together to tackle complex tasks.

The ZigBee stack architecture, which is depicted in Figure 13, is based on the standard Open Systems Interconnection (OSI) seven-layer model, but defines only those layers relevant to achieving functionality in the intended market space. The IEEE 802.15.4-2003 standard defines the lower two layers: the physical layer (PHY) and the Medium Access Control (MAC) sub-layer. The ZigBee Alliance builds on this foundation by providing the network (NWK) layer and the framework for the application layer, which includes the Application Support Sub-layer (APS), the ZigBee Device Objects (ZDO) and the manufacturer-defined application objects (ZigBee Alliance, 2005).

The routing protocol optimizes the shortest and most reliable path through the network and can dynamically change, so as to take evolving conditions into account. This enables an extremely reliable network, since the network can heal itself if one node is disabled. This is very similar to the redundancy employed in the Internet. ZigBee networks are primarily intended for low duty cycle sensor networks (<1%). A new network node may be recognized and associated in about 30 ms. Waking up a sleeping node takes about 15 ms, as does accessing a channel or transmitting data. ZigBee applications benefit from the ability to quickly attach information, detach, and go to sleep mode, which results in low power consumption and extended battery life.

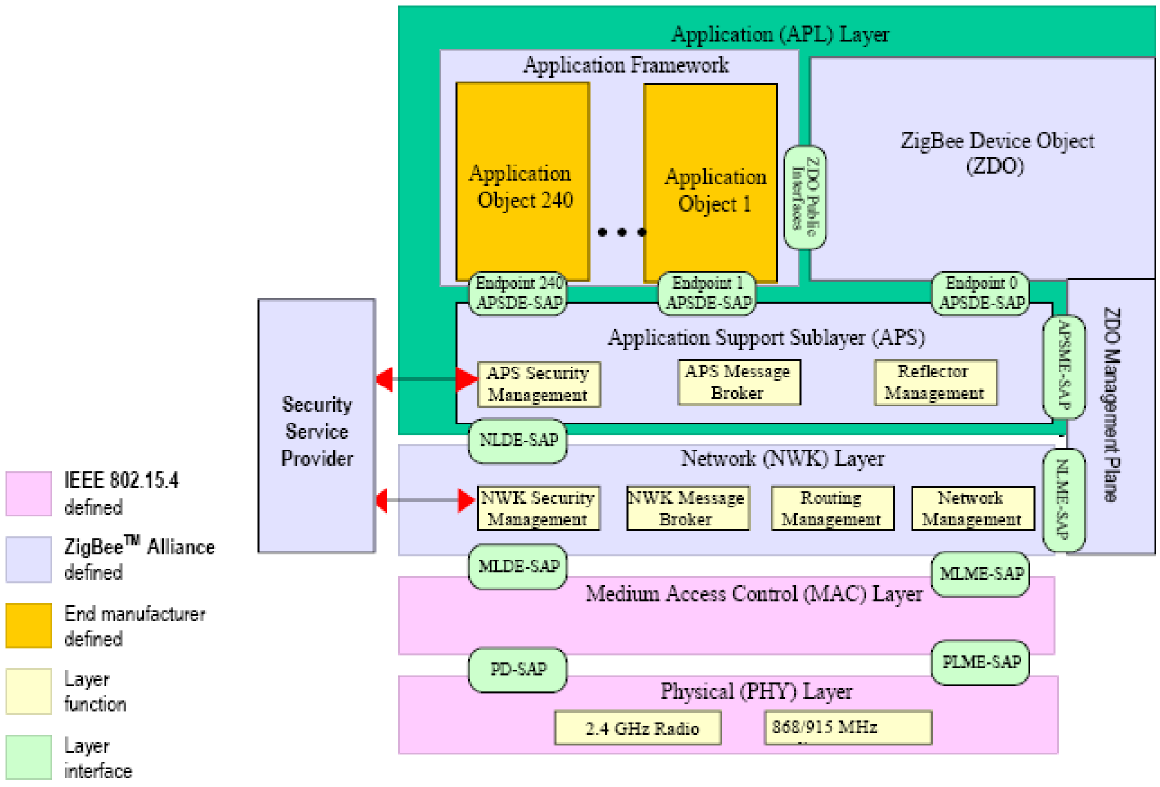


Figure 12. *Outline ZigBee stack architecture (ZigBee Alliance, 2005)*

In this standard is introduced what is known as profile, a collection of device descriptions, which together form a cooperative application. As an example, a thermostat on one node communicates with a furnace on another node. Together, they cooperatively form a heating application profile. Profiles are still in development and currently they do not cover all fields. The majority of profiles defined so far until today are for the home appliances market and the industrial market. The standard is also under improvement and further study. Latest standard published is called 'Smart Energy Profile Specification. This specification provides standard interfaces and device definitions to allow interoperability among ZigBee devices produced by various manufacturers of electrical equipment, meters, and Smart Energy enabling products (ZigBee Alliance, 2008).

1.5 TINYOS

Global efforts, led by the Computer Science Department at the University of California at Berkeley, have succeeded in developing an operating system with a component-based runtime environment designed to support embedded systems with a minimal amount of physical hardware in order to keep the overall system size very small. TinyOS is developed then by a consortium led by the University of California, Berkeley in co-operation with Intel Research concluding in the official presentation to the world by Jason Lester Hill with the publication of a thesis in 2003. It is an open source operative system in constant evolution for more information about current status and future releases look in www.tinyos.org.

TinyOS is a component based operating system designed to run in resource constrained wireless devices. It provides highly efficient communication primitives and fine-grained concurrency mechanisms to application and protocol developers. A key concept in TinyOS is the use of event based programming in conjunction with a highly efficient component model. TinyOS enables system-wide optimization by providing a tight coupling between hardware and software, as well as flexible mechanisms for building application specific modules. TinyOS has been designed to run on a generalized architecture where a single CPU is shared between application and protocol processing (Jason Lester Hill, 2003).

In an event based system, a single execution context is shared between unrelated processing tasks. In TinyOS, each system module is designed to operate by continually responding to incoming events. When an event arrives, it brings the required execution context with it. When the event processing is completed, it is returned back to the system.

TinyOS focuses on three high-level goals in sensor network system architectures (Philip Levis *et al*., 2004):

• Take account of current and likely future designs for sensor networks and sensor network nodes.

• Allow diverse implementations of both operating system services and applications, in varying mixes of hardware (in different mote generations) and software.

• Address the specific and unusual challenges of sensor networks: limited resources, concurrency interface intensive operation, a need for robustness, and application-specific requirements.

The need to handle high concurrency comes from the observation that sensor nodes predominantly process multiple information flows on the fly, as opposed to performing heavy computation. Nodes must simultaneously execute several operations at once, but have limited storage. Hardware events are interrupts, caused by a timer, sensor, or communication device. Tasks are a form of deferred procedure call that allows a hardware event or task to postpone processing. System modularity is based on a component model. A named component encapsulates some fixed-size state and a number of tasks. Components interact with each other strictly via function call interfaces, which are related groups of commands and events.

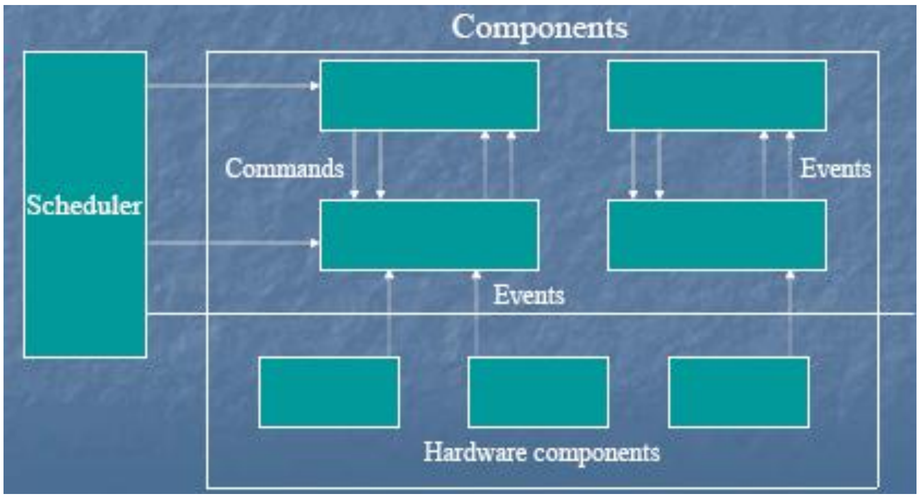


Figure 13. *Design diagram of a TinyOs application.*

TinyOS applications are written in a programming language called Nesc. It is an extension of C: produces efficient code for all the target microcontrollers that are likely to be used in sensor networks. C provides all the low-level features necessary for accessing hardware, and interaction with existing C code is simplified. Last but not least, many programmers are familiar with C. C does have significant disadvantages: it provides little help in writing safe code or in structuring applications. NesC addresses safety through reduced expressive power and structure through components. NesC's concurrency model should be extended to admit multiprocessors, blocking operations, and a more general notion of threads, as discussed above. Such an approach would lead to a rich set of concurrency primitives specifically tailored for component oriented programming of large-scale systems. (D. Gay *et al.*, 2003).

Next figure shows a screenshot of this application and the type of environment that is missed actually by developers. It corresponds to the efforts coming from the Eclipse community. (http://www.eclipse.org/).

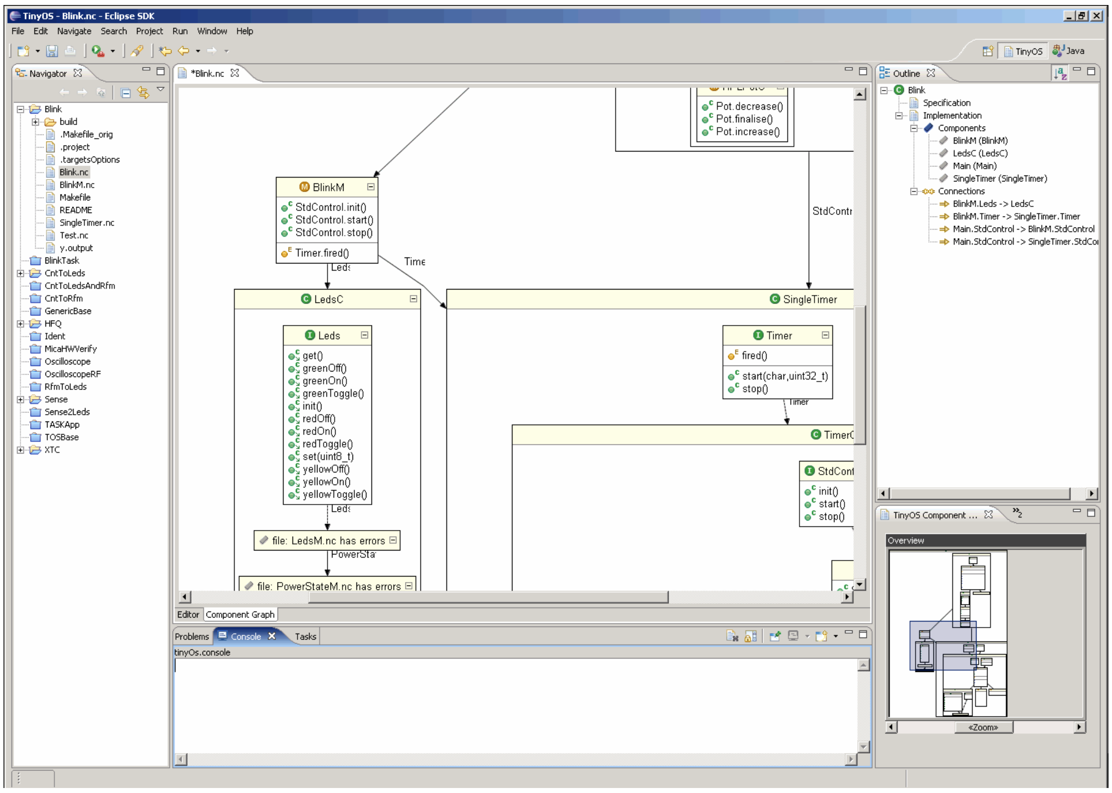
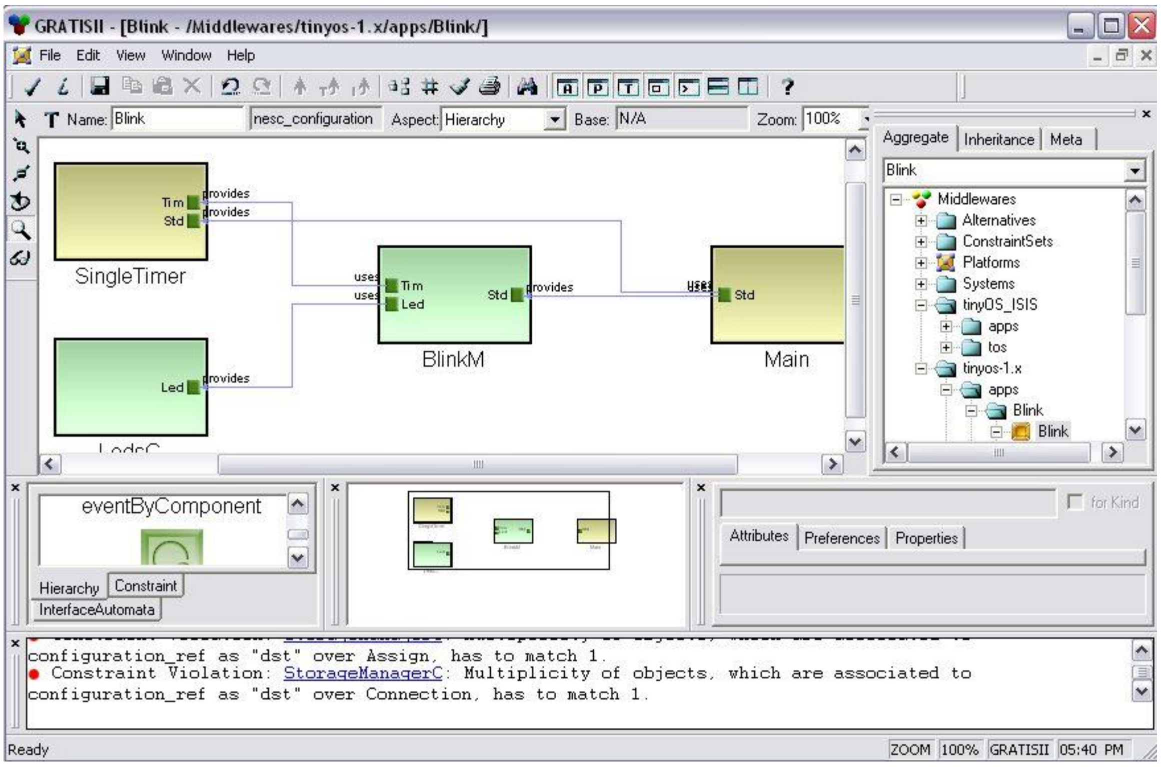


Figure 14. *Eclipse plug-in screenshot.*

There is another project from Vanderbilt University called “NEST: Network Embedded Systems Technology”. Researchers working in this project that ended in 2005 have developed middleware services, tools and applications for this technology. It is of particular interest for this project one software tool from this University called "GRATISII". This is a model-based approach to the development of applications based on TinyOS with NesC, an important NEST platform. Operative System (OS) and application component interfaces along with their interdependencies are captured in a graphical environment and the glue code that ties together the application and OS components are automatically generated. GRATIS II is a fully functional modelling, code generation and parsing environment developed using model integrated technology, specifically the Generic Modelling Environment (GME). There has been some practical work using this software tool although is not easy to understand how it works and how to create graphically new applications. The next screenshot shows this software tool that is very similar to the previous one presented.



```
configuration_ref as "dst" over Assign, has to match 1.
● Constraint Violation: StorageManagerC: Multiplicity of objects, which are associated to
configuration_ref as "dst" over Connection, has to match 1.
```

Figure 15 *GratisII software tool screenshot.*

Both Software tools presented are the most significant ones among the research community; the main idea behind them is to design new TinyOS applications in a graphical way. This makes sense as it would make easier for developers to picture new software code, and it is accorded with the way TinyOS is designed in terms of modules, components, interfaces etc.

1.6 APPLICATIONS AND PROJECTS

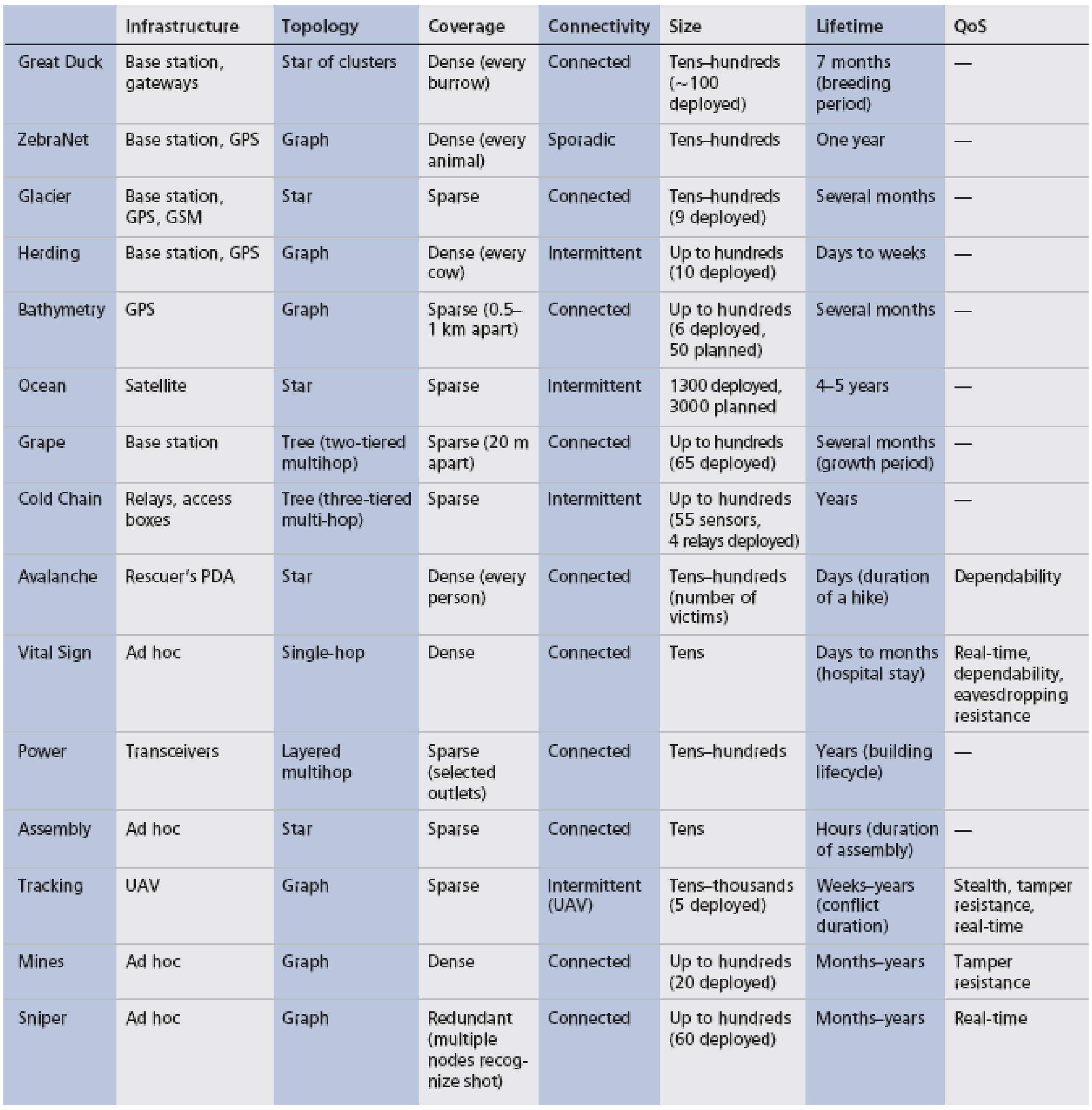
A summary of existing applications and projects is important for the purpose of this research, which will give to the reader a view of the level of acceptance of this technology at the present time. The following tables show some example applications carried out using wireless sensor networks mostly from (Kay Römer *et al.*, 2004). The authors of this article have classified these applications based on different aspects. Looking at the following tables 'Deployment' refers to the physical work of placing the nodes of a network in the scenario under study. 'Mobility' refers to the possibility of nodes changing their location in the network. 'Resources' classifies the networks according to the size of the nodes. 'Heterogeneity' means what sensors have been attached to the nodes and if they are all the same type. 'Modality' refers to the transmission medium used in the network.

Table 8. *Classification of the sample applications according to the design space (Kay Römer et al., 2004).*

	Deployment	Mobility	Resources	Cost	Energy	Heterogeneity	Modality
Great Duck	Manual, one-time	Immobile	Matchbox	~200 USD	Battery, solar	Weather stations, burrow nodes, gateways	Radio
ZebraNet	Manual, one-time	All, continuous, passive	Matchbox	—	Battery	Nodes, gateway	Radio
Glacier	Manual, one-time	All, continuous, passive	Brick	—	Battery	Nodes, base station	Radio
Herding	Manual, one-time	All, continuous, passive	Brick	~1000 USD	Battery	Homogeneous	Radio
Bathymetry	Manual, one-time	All, occasional, passive	Brick	—	Battery	Homogeneous	Radio
Ocean	Random, iterative	All, continuous, passive	Brick	~15000 USD	Battery	Homogeneous	Radio
Grape	Manual, one-time	Immobile	Matchbox	~200 USD	Battery	Sensors, gateway, base station	Radio
Cold Chain	Manual, iterative	Partly (sensors), occasional, passive	Matchbox (sensors), brick (relays)	—	Battery	Sensors, relays, access boxes, warehouse	Radio
Avalanche	Manual, one-time	All, continuous, passive	Matchbox	—	Battery	Homogeneous	Radio
Vital Sign	Manual	All, continuous, passive	Matchbox	—	Battery	Medical sensors, patient identifier, display device, setup pen	Radio, IR light (for setup pen)
Power	Manual, iterative	Immobile	Matchbox	—	Power grid	Sensor nodes, transceivers, central unit	Radio (sensor unidirectional)
Assembly	Manual, one-time	All, occasional, passive	Matchbox	~100 Euro	Battery	Different sensors	Radio
Tracking	Random (thrown from aircraft)	All, occasional, passive	Matchbox	~ 200 USD	Battery	Homogeneous	Radio
Mines	Manual	All, occasional, active	Brick	—	Battery	Homogeneous	Radio, ultrasound (for localization)
Sniper	Manual	Immobile	Matchbox with FPGA	~200 USD	Battery	Homogeneous	Radio

The next table is the continuation of the previous one classifying the applications according to the following parameters. Infrastructure means if there is some other communication system involved to transmit the data apart of the network itself. Connectivity outlines the fact of connections between nodes of the network, if nodes are isolated most of the time and enter the communication range of other nodes only occasionally, connectivity is sporadic. Quality of Service (QoS) is the last category of this table, making mention to the grade of quality some of these networks must support because of the application they are working with. It is real time if the network must be reporting a crucial parameter periodically.

Table 9. *Classification of the sample applications according to the design space (Kay Römer et al., 2004).*



	Infrastructure	Topology	Coverage	Connectivity	Size	Lifetime	QoS
Great Duck	Base station, gateways	Star of clusters	Dense (every burrow)	Connected	Tens–hundreds (~100 deployed)	7 months (breeding period)	—
ZebraNet	Base station, GPS	Graph	Dense (every animal)	Sporadic	Tens–hundreds	One year	—
Glacier	Base station, GPS, GSM	Star	Sparse	Connected	Tens–hundreds (9 deployed)	Several months	—
Herding	Base station, GPS	Graph	Dense (every cow)	Intermittent	Up to hundreds (10 deployed)	Days to weeks	—
Bathymetry	GPS	Graph	Sparse (0.5–1 km apart)	Connected	Up to hundreds (6 deployed, 50 planned)	Several months	—
Ocean	Satellite	Star	Sparse	Intermittent	1300 deployed, 3000 planned	4–5 years	—
Grape	Base station	Tree (two-tiered multihop)	Sparse (20 m apart)	Connected	Up to hundreds (65 deployed)	Several months (growth period)	—
Cold Chain	Relays, access boxes	Tree (three-tiered multi-hop)	Sparse	Intermittent	Up to hundreds (55 sensors, 4 relays deployed)	Years	—
Avalanche	Rescuer's PDA	Star	Dense (every person)	Connected	Tens–hundreds (number of victims)	Days (duration of a hike)	Dependability
Vital Sign	Ad hoc	Single-hop	Dense	Connected	Tens	Days to months (hospital stay)	Real-time, dependability, eavesdropping resistance
Power	Transceivers	Layered multihop	Sparse (selected outlets)	Connected	Tens–hundreds	Years (building lifecycle)	—
Assembly	Ad hoc	Star	Sparse	Connected	Tens	Hours (duration of assembly)	—
Tracking	UAV	Graph	Sparse	Intermittent (UAV)	Tens–thousands (5 deployed)	Weeks–years (conflict duration)	Stealth, tamper resistance, real-time
Mines	Ad hoc	Graph	Dense	Connected	Up to hundreds (20 deployed)	Months–years	Tamper resistance
Sniper	Ad hoc	Graph	Redundant (multiple nodes recognize shot)	Connected	Up to hundreds (60 deployed)	Months–years	Real-time

There follows a brief overview of the aim each of the projects listed in the table.

Bird observation on great duck island, A wireless sensor network (WSN) is being used to observe the breeding behaviour of a small bird called Leach's Storm Petrel on Great Duck Island, Maine, United States. These birds are easily disturbed by the presence of humans, so a WSN seems an appropriate way of better understanding their behaviour (Kay Römer *et al*., 2004).

Zebranet, A WSN is being used to observe the behaviour of wild animals within a spacious habitat (e.g., wild horses, zebras, and lions) at the Mpala Research Center in Kenya. Of particular interest is the behaviour of individual animals (e.g., activity patterns of grazing, graze-walking, and fast moving), interactions within a species, interactions among different species (e.g., grouping behaviour and group structure), and the impact of human development on the species. The observation period is scheduled to last a year or more. The observation area may be as large as hundreds or even thousands of square kilometres.

Glacier Monitoring, A sensor network is being used to monitor subglacier environments at Briksdalsbreen, Norway, with the overall goal of better understanding the Earth’s climate. Of particular interest are displacements and the dynamics inside the glacier. A lengthy observation period of months to years is required. Sensor nodes are deployed in drill holes at different depths in the glacier ice and in the till beneath the glacier. Sensor nodes are equipped with pressure and temperature sensors and a tilt sensor for measuring the orientation of the node.

Cattle Herding, A WSN is being used to implement virtual fences, with an acoustic stimulus being given to animals that cross a virtual fence line. Movement data from the cows controls the virtual fence algorithm that dynamically shifts fence lines. Such a system can reduce the overheads of installing and moving physical fences, and improve the usage of feedlots.

Bathymetry, A sensor network is being used to monitor the impact on the surrounding environment of a wind farm off the coast of England. Of particular interest here is the influence on the structure of the ocean bed (e.g., formation of sand banks) and on tidal activity. Sensor nodes are deployed on the ocean bed by dropping them from a ship at selected positions, their location being fixed on the ocean bed by an anchor.

Ocean Water Monitoring, The ARGO project is using a sensor network to observe the temperature, salinity, and current profile of the upper ocean. The goal is a quantitative description of the state of the upper ocean and the patterns of ocean climate variability, including heat and freshwater storage and transport. Intended coverage is global, and observation is planned to last for several years. Measurement data is available almost in real time. The project uses free-drifting profiling sensor nodes equipped with temperature and salinity sensors. The nodes are dropped from ships or planes.

Grape Monitoring, A WSN is being used to monitor the conditions that influence plant growth (e.g., temperature, soil moisture, light, and humidity) across a large vineyard in Oregon, United States. The goals include supporting precision harvesting (harvesting an area as soon as the grapes in it are ripe), precision plant care (adapting the water/fertilizer/pesticide supply to the needs of individual plants), frost protection, predicting insect/pest/fungi development, and developing new agricultural models. In a first version of the system, sensor nodes are deployed across a vineyard in a regular grid about 20 m apart. A temperature sensor is connected to each sensor node via a cable in order to minimize false sensor readings due to heat disseminated by the sensor nodes.

Cold Chain Management, The commercial Securifood system is a WSN for monitoring the temperature compliance of cold chains from production, via distribution centres and stores, to the consumer. Clients receive an early warning of possible breaks in the cold chain.

Rescue of Avalanche Victims, A WSN is being used to assist rescue teams in saving people buried in avalanches. The goal is to better locate buried people and to limit overall damage by giving the rescue team additional indications of the state of the victims and to automate the prioritization of victims (e.g., based on heart rate, respiration activity, and level of consciousness). For this purpose, people at risk (e.g., skiers, snowboarders, and hikers) carry a sensor node equipped with an oximeter (a sensor that measures the oxygen level in blood) that permits heart rate and respiration activity to be measured.

Vital Sign Monitoring, Wireless sensors are being used to monitor vital signs of patients in a hospital environment. Compared to conventional approaches, solutions based on wireless sensors are intended to improve monitoring accuracy while also being more convenient for patients.

Power Monitoring, A WSN is being used to monitor power consumption in large and dispersed office building. The goal is to detect locations or devices that are consuming a lot of power to provide indications for potential reductions in power consumption.

Parts Assembly, A WSN is being used to assist people during the assembly of complex composite objects such as dot- yourself furniture. This saves users from having to study and understand complex instruction manuals, and prevents them from making mistakes.

Tracking Military Vehicles, A WSN is being used to track the path of military vehicles (e.g., tanks). The sensor network should be unnoticeable and difficult to destroy. Tracking results should be reported within given deadlines.

Self-Healing Mine Field, Anti-tank landmines are being equipped with sensing and communication capabilities to ensure that a particular area remains covered even if the enemy tampers with a mine to create a potential breach lane. If tampering is detected by the mine network, an intact mine hops into the breach using a rocket thrusters.

Sniper Localization, A WSN is being used to locate snipers and the trajectory of bullets, providing valuable clues for law enforcement. The system consists of sensor nodes that measure the muzzle blast and shockwave using acoustic sensors.

There is an article where this technology is presented as an alternative to study the welfare of animals (Ian McCauley *et al*, 2005). The SECOAS project for oceanographic study developed at the University of Kent. The project partners are BT exact plc, Intelisys Ltd, Plextek Ltd, University of Essex, University College London, and the University of East Anglia. The project started in 2003 and will finish in 2006 (http://www.cs.kent.ac.uk/projects/secoas/).

The project FloodNet, primary goal is to demonstrate a methodology whereby a set of sensors monitoring the river and functional floodplain environment at a particular location are connected by wireless links to other nodes to provide an "intelligent" sensor network (http://envisense.org/floodnet/floodnet.htm). GlacsWeb, the aim of this project at the University of Southampton is to be able to monitor glacier behaviour via different sensors and link them together into an intelligent web of resources. Probes are placed on and under glaciers and data collected from them by a base station on the surface. Measurements include temperature, pressure and sub glacial movement (http://envisense.org/glacsweb.htm).

It is interesting to mention a project called “codeblue” and the paper that describes the experience developing a combined hardware and software platform for medical sensor networks (Victor Shnayder, *et al.* 2005), http://www.eecs.harvard.edu/~mdw/proj/codeblue/. It is presented initial results for the CodeBlue prototype demonstrating the integration of medical sensors with the publish/subscribe routing substrate. They have experimentally validated the prototype on a 30-nodesensor network testbed, demonstrating its scalability and robustness as the number of simultaneous queries, data rates, and transmitting sensors are varied. They also study the effect of node mobility, fairness across multiple simultaneous paths, and patterns of packet loss, confirming the system's ability to maintain stable routes despite variations in node location and data rate.

The projects discussed here represent some of the research effort by scientific institutions. Some are commercial applications and two of them are military research projects (Self-Healing Mine Field and the sniper location). In most cases only presentations or summaries are available and not the full detail implementation. In the military case only monitoring purposes are required with no mention of transmitting an alarm. The Code Blue project handles a scenario where time response could be critical like in an Oil leakage detection system, but at the moment is aimed to improve monitoring of patients in a hospital or residence. Because it is a research project is running on parallel to existing systems and it does not implement any technique towards transmitting alarms. In this line are the rest of projects commented their objective to monitor and to control systems, covering huge geographical areas. They are alternatives techniques in their fields as in most present solutions to observe a system without the presence of a person and therefore disrupting such systems, this is the case of animal observation, agricultural monitoring and environment survey. Therefore what it has been presented in this report is just to give a view of what other research groups, companies or institutions are doing using this technology.

1.7 THESIS AIMS AND OBJECTIVES

The aim of this thesis is to evaluate the application of wireless sensor technology in environmental monitoring projects. The result of this research will serve as a practical guide to future research work carried out in this area. Practical work with a hardware platform and existing software will be necessary to achieve this.

In a scenario like an Oil storage facility where oil leakage poses a serious environmental hazard, alarms have to be triggered and attended as quickly as possible. This technology could help to handle such a critical situation. To design a system capable of such a task it is necessary to accomplish various objectives.

- Research the field of wireless sensor networks in order to identify a suitable hardware and software platform to start the practical work.
- Test the technology to evaluate it's capabilities as an alternative to existing monitoring and control systems already in use. The experiments and practical work will try to recreate scenarios or case studies where this technology could pose an advantage versus other systems and techniques.
- Develop software tools and hardware necessary to do the practical work and evaluate this process.

2 PROCUREMENT AND EVALUATION OF DEVELOPMENT PLATFORM

The literature review presented this technology and gave some theoretical background about it. This section will present the chosen hardware platform and discusses the software designed along with the hardware like TinyOS, NesC and Java.

2.1 TMOTE HARDWARE PLATFORM

The first task in this project after researching the field was to choose the hardware platform to start experimenting with. The next table shows different hardware platforms available in 2005 and their main technical specifications.

Table 10. *Most popular hardware platforms for wireless sensor networks at a glance 2005.*

	Manufacturer	Power Consumption C/R/T (mA)	Power sleep (μA)	CPU type & clock freq (MHz)	Memory kB		Radio Standard
					SRAM	Flash	
Imote	Intel	15/24/24	250	32bit ARM @12	64	512	BT
Micaz	Crossbow	8/20/18	27	8bit Atmel @8	4	128	802.15.4
TmoteSky	Moteiv	**1/20/18**	**6**	16bit TI @8	10	1024	802.15.4
Mica2	Crossbow	8/10/27	19	8bit Atmel @8	4	128	300-900Mhz
Imote 2	Intel	40/20/18	100	32b XS@13(104)	256	32000	802.11

The reasons behind choosing TmoteSky or Telos, which it is the common name used for this hardware platform, are based on the specifications highlighted in the previous table. The hardware platform with a microprocessor MSP430 from Texas Instruments consumes less power when it is just active, when is receiving or transmitting. It is also the platform consuming less power when in sleep mode, which it will be the main operational mode of the platform in order to save power. This mode of operation is also in line with the requirements and characteristics of this technology, where nodes get the sensor readings, transmit them if necessary and then back to sleep until next reading is available. Using this platform the bottom layers of the Communication protocol stack (Physical layer and Link layer) of the standard 802.15.4 are already implemented by the transceiver from Chipcon. This is important as it could be compatible with other devices using the same standard.

During 2007 Moteiv Company decided to discontinue support for the Tmote Sky platform as the company joined a software company called Sentilla. They are focused now in developing a wireless sensor framework in Java with the main objective of providing end user applications for this technology. Zigbee Alliance is today an organization with more than 100 members, and some of the finest companies and individuals in the industry. The direction taken by the participants of this organization has been to produce specific devices with wireless capabilities. Products examples are thermostats, lights, central heating and air conditioning control units, and metering devices to monitor gas and electricity consumption. They have focused mainly in the home automation market. There also members dedicated to the end user software applications, raw hardware platforms, development kits, network gateways and complete solutions which include software and hardware ready to use.

As shown in figure 16 the board has fixed locations for different types of sensors to measure parameters such as, relative humidity, temperature, photo synthetically active radiation light, total solar light, the internal temperature of the microcontroller and the internal voltage. A decision was made to purchase the hardware with two of these sensors (internal voltage and temperature), providing the opportunity to try any sensor and also reducing the price per unit.

Next picture and description shows the main technical aspects of the hardware platform.

Summary of Technical Aspects:

- 250kbps 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver
- Interoperability with other IEEE 802.15.4 devices
- 8MHz Texas Instruments MSP430 microcontroller (10k RAM, 48k Flash)
- Integrated ADC, DAC, Supply Voltage Supervisor, and DMA Controller
- Integrated onboard antenna with 50m range indoors / 125m range outdoors
- Ultra low current consumption
- Fast wakeup from sleep (<6us)
- Hardware link-layer encryption and authentication
- Programming and data collection via USB
- 16-pin expansion support and optional SMA antenna connector
- TinyOS support : mesh networking and communication implementation

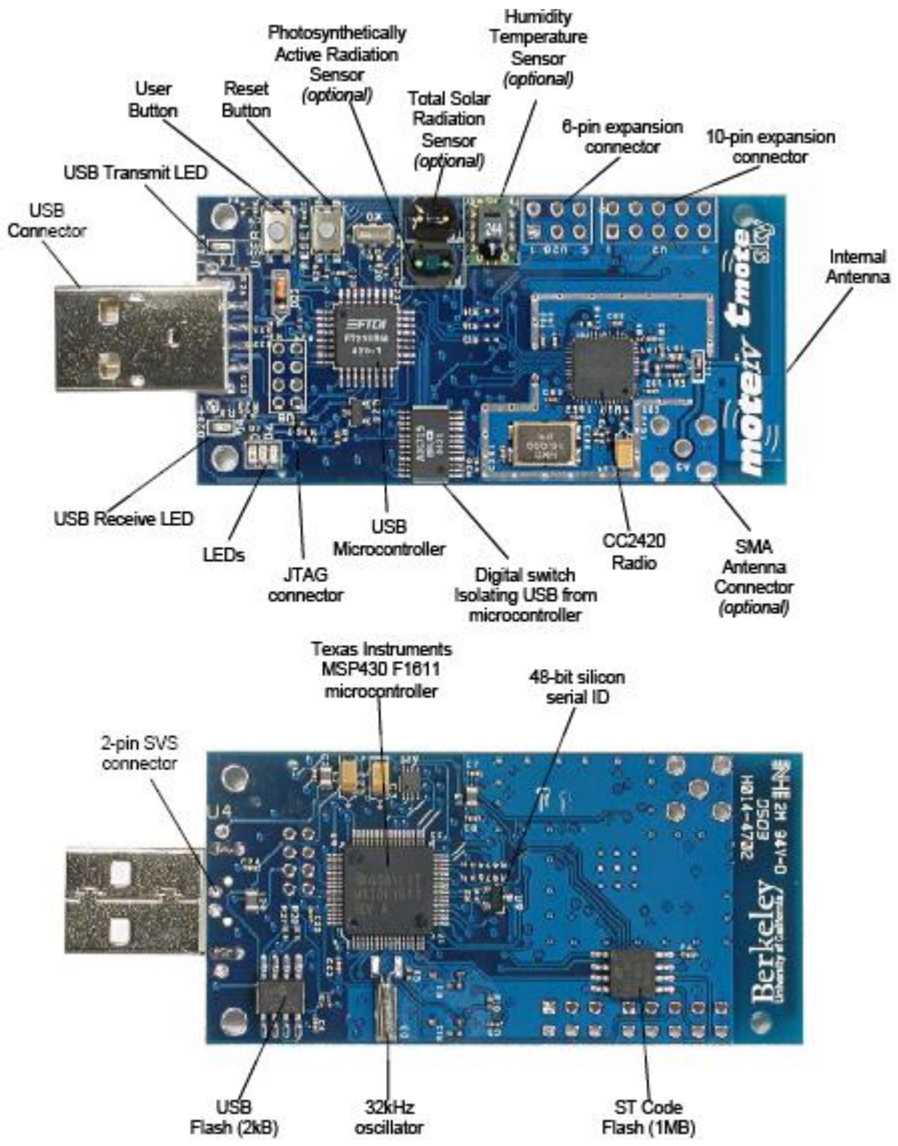

Photosynthetically Active Radiation Sensor (optional)
Humidity Temperature Sensor (optional)
User Button
Reset Button
Total Solar Radiation Sensor (optional)
6-pin expansion connector
10-pin expansion connector
USB Transmit LED
USB Connector
Internal Antenna
USB Receive LED
LEDs
JTAG connector
USB Microcontroller
Digital switch Isolating USB from microcontroller
CC2420 Radio
SMA Antenna Connector (optional)
Texas Instruments MSP430 F1611 microcontroller
48-bit silicon serial ID
2-pin SVS connector
USB Flash (2kB)
32kHz oscillator
ST Code Flash (1MB)

Figure 16. *Tmote Sky a node running TinyOS, front and back of the module.*

<u>2.2</u> TINYOS PRACTICAL WORK

In section 1.5 has been presented briefly the operating system that runs in the hardware platform of this project the TmoteSky. The decision behind adopting TinyOS comes after looking to alternative systems. TinyOS is open source code which facilitates greatly the labour of producing new software tools, understanding of the technology and support in general. It is definitely the option adopted by the majority of the research, scientific, and academic community. The flexibility and the cost of this solution is probably the main quality versus some of the alternatives like Zigbee, where only the standard is open source code and other tools have to be purchased. There are not public forums using this technology just private companies and industrial firms. Each hardware platform manufacturer using Zigbee provides the software communication stack; it is the consumer who produces the application. There is not much room to try different routing techniques or new algorithms to hop messages from one node to another. After purchasing some hardware units, the author took several software tutorials. There is a main web page for TinyOS with questions, tutorials and help archives, http://www.tinyos.net/.

Before going in detail with the practical work carried using this operative system the following figure shows graphically the architecture of the system in terms of software.

PC

A PC will have Cygwin software installed which makes possible to program nodes, run java tools and contains the TinyOS libraries and source code. Cygwin software is also open source and it just serves as a platform for TinyOS. It is an interface between the user or other PC applications and the Wireless Sensor Network. Tools written in Java are used to process the information coming from the network. Nodes are connected to the PC either to program them or to transfer the data coming from the network.

Node

A node of the network is programmed with an application written in NesC (TinyOS programming Language). Therefore it contains a reduced version of TinyOS as it will just execute the tasks written in the program.

It is interesting to have a look after TinyOS is installed on a PC to the folder structure and the brief description of what each folder contains, next figure shows this. It is based on a distribution of TinyOS released by Moteiv, the company that manufactures the hardware platform currently used in this project.

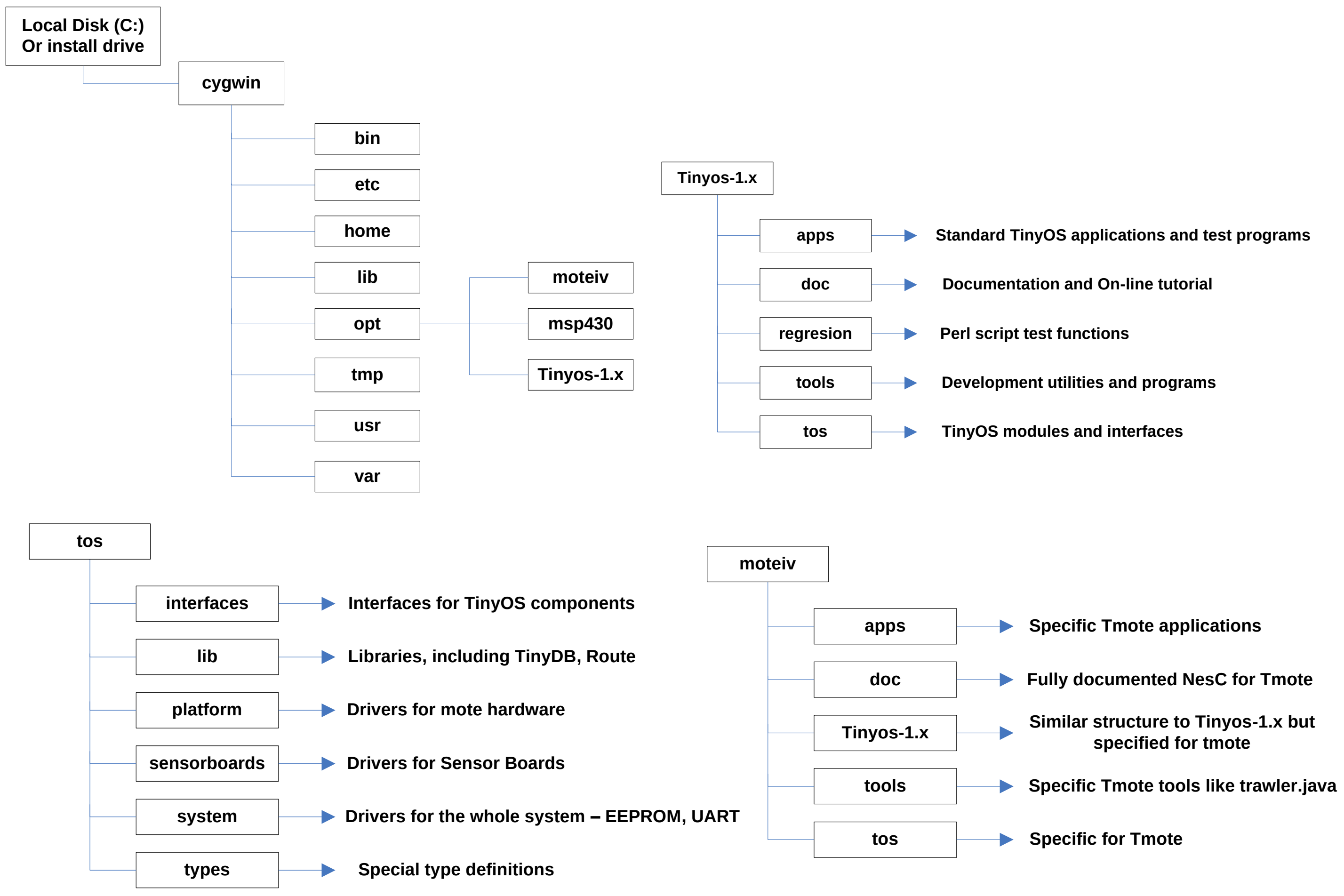

Local Disk (C:)
Or install drive
cygwin
bin
etc
home
lib
opt
tmp
usr
var
moteiv
msp430
Tinyos-1.x
Tinyos-1.x
apps
Standard TinyOS applications and test programs
doc
Documentation and On-line tutorial
regresion
Perl script test functions
tools
Development utilities and programs
tos
TinyOS modules and interfaces
tos
interfaces
Interfaces for TinyOS components
lib
Libraries, including TinyDB, Route
platform
Drivers for mote hardware
sensorboards
Drivers for Sensor Boards
system
Drivers for the whole system – EEPROM, UART
types
Special type definitions
moteiv
apps
Specific Tmote applications
doc
Fully documented NesC for Tmote
Tinyos-1.x
Similar structure to Tinyos-1.x but specified for tmote
tools
Specific Tmote tools like trawler.java
tos
Specific for Tmote

Figure 17. *Block diagram with the TinyOS folder structure and brief description of them.*

It is very important towards the development of new components of TinyOS to understand how to use contents in each folder. Under Moteiv folder there is now a folder called Nesdoc where is fully documented all NesC interfaces functions and modules. This is a very fast changing environment and demands constant update and tracking of any new features, tools and applications. The practical work with TinyOS has involved the study of software code inside each of these folders, as the majority of it is not explained anywhere. As mentioned earlier the interface to run and compile TinyOS is called Cygwin that is a Linux-like environment for Windows.

A sensor connected trough USB port to a pc acts as a base station collecting the packets coming over the radio. The communication establishes follows a serial line communication protocol. It is loosely based on the PPP in HDLC-like framing described in RFC-1662. It uses the same framing and escape sequences and a 16-bit CRC. The protocol always assumes Address and Control field compression, for more details refer to TinyOS web site. Although Tmote hardware platform is connected to the computer trough USB port the operative system uses a virtual COM port driver provided by FTDI. The user sees the device as connected to a COM port.

2.2.1 TinyOS Message

The structure of the data packets has been part of the experimental work carried out, as it has been identified as a crucial key to develop further applications. There is a lack of information in this area from TinyOS community and the manufacturers of hardware platforms. The description that follows tries to explain about the structure of these packets in more detail.

A TinyOS data packet has a maximum length of 255 bytes; the raw data packet is wrapped on both ends by a frame synchronization byte of with a value 0x7E. This is used to detect the start and end of a packet from the stream. The raw data packet uses an escape byte of value 0x7D. This is needed in case a byte of payload data is the same as a reserved byte code, such as the frame synch byte of value 0x7E. There are different types of data packets; in fact there is flexibility in changing the structure as needed.

Here it is described the structure of TinyOS message, oscope message packet and surge message packet.

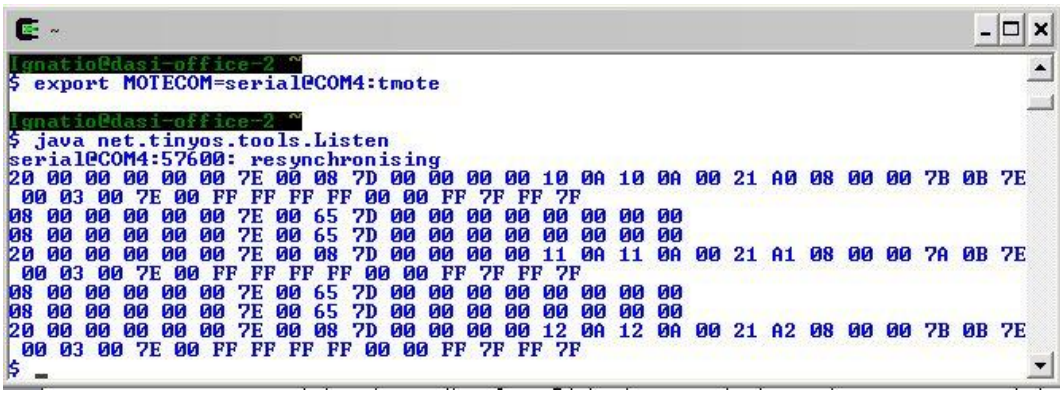
The so called ‘TinyOS message’ is the basic structure of any data packet transmitted over the radio; other data structures created are a combination of the basic one and the one defined for a certain application. The structure of the packets is defined by a heading file written in C language that is included in the NesC program. For a TinyOS message the file is called ‘AM.h’. It is located under folder “\cygwin\opt\tinyos-1.x\tos\platform\telos\AM.h”. One of the most important parts in the code is the one describing the structure,

```
typedef struct TOS_Msg      (TOS_Msg = TinyOS Message)
{
  /* The following fields are transmitted/received on the radio. */
  uint8_t length;
  uint8_t fcfhi;
  uint8_t fcflo;
  uint8_t dsn;
  uint16_t destpan;                          Standard Header
  uint16_t addr;
  uint8_t type;
  uint8_t group;
  int8_t data[TOSH_DATA_LENGTH];

  /* The following fields are not actually transmitted or received
   * on the radio! They are used for internal accounting only.
   * The reason they are in this structure is that the AM interface
   * requires them to be part of the TOS_Msg that is passed to
   * send/receive operations.
   */
  uint8_t strength;
  uint8_t lqi;
  bool crc;
  bool ack;
  uint16_t time;
} __attribute((packed)) TOS_Msg;
```

2.2.2 Oscope Message

This message type will be used to describe in more detail the fields in the data structure packet. ‘Oscope’ is an application that gets the readings from the ADC and sends them to the Base Station. Using a java tool called ‘Oscope.java’ it is possible to display graphically those readings. The experiment consists in programming one Tmote with an application called ‘OscopeIgn’ which is transmitting the reading from Light detector, the internal temperature, and the internal voltage level of the microcontroller. Using a Java tool called ‘Listen.java’ the packets are display in hexadecimal values in cygwin command prompt window. Tmote connected to the pc is running an application called TOSBase. Some configurations commands are needed in order to capture and display the data coming. These are the commands type in the cygwin bash window.



```
Ignatio@dasi-office-2 ~
$ export MOTECOM=serial@COM4:tmote

Ignatio@dasi-office-2 ~
$ java net.tinyos.tools.Listen
serial@COM4:57600: resynchronising
20 00 00 00 00 00 7E 00 08 7D 00 00 00 00 10 0A 10 0A 00 21 A0 08 00 00 7B 0B 7E
 00 03 00 7E 00 FF FF FF FF 00 00 FF 7F FF 7F
08 00 00 00 00 00 7E 00 65 7D 00 00 00 00 00 00 00 00
08 00 00 00 00 00 7E 00 65 7D 00 00 00 00 00 00 00 00
20 00 00 00 00 00 7E 00 08 7D 00 00 00 00 11 0A 11 0A 00 21 A1 08 00 00 7A 0B 7E
 00 03 00 7E 00 FF FF FF FF 00 00 FF 7F FF 7F
08 00 00 00 00 00 7E 00 65 7D 00 00 00 00 00 00 00 00
08 00 00 00 00 00 7E 00 65 7D 00 00 00 00 00 00 00 00
20 00 00 00 00 00 7E 00 08 7D 00 00 00 00 12 0A 12 0A 00 21 A2 08 00 00 7B 0B 7E
 00 03 00 7E 00 FF FF FF FF 00 00 FF 7F FF 7F
$ _
```

Figure 18. C*ygwin bash shell command window for Oscope message experiment.*

Oscope message is a result of the combination of TOS message and a structure defined for this application in the file Oscope.h, located under folder “cygwin\opt\tinyos-1.x\tos\lib\Oscope\Oscope.h”. Here is highlighted the main part of this file.

```
typedef struct OscopeMsg
{
  uint16_t sourceMoteID;
  uint16_t lastSampleNumber;
  uint16_t channel;
  uint16_t data[OSCOPE_BUFFER_SIZE];
      } OscopeMsg_t;
```

The data captured,

<u>1st packet</u>

1A 01 08 5D FF FF FF FF 0A 7D 00 00 92 18 00 00 AC 01 A3 01 10 01 7D 01 BC 01 6F 01 43 01 A6 01 AB 01 1F 01

<u>2nd packet</u>

1A 01 08 5E FF FF FF FF 0A 7D 00 00 92 18 01 00 7D 0A 5F 0A 3A 0A 7F 0A 96 0A 7A 0A 7F 0A 4B 0A 63 0A 86 0A

<u>3rd packet</u>

1A 01 08 5F FF FF FF FF 0A 7D 00 00 92 18 02 00 A7 0F A7 0F A7 0F A7 0F A7 0F A8 0F A7 0F A7 0F A7 0F AA 0F

Description of fields as follows of first packet received,

Sample Message Hexadecimal Values of 1st packet

1A 01 08 5D FF FF FF FF 0A 7D	00 00 92 18 00 00 AC 01 A3 01 10 01 7D 01 BC 01 6F 01 43 01
Standard header	**Oscope Message**

Length = 1A = 26 bytes which it is the number of bytes we have after 7D which is also the last byte defined by TOS_Msg structure.

Frame Control Field High Byte (fcfhi) = 01 note that these fields are defined by the standard IEEE 802.15.4 for more details refer to it.

Frame Control Field Low Byte (fcflo) = 08.

Data Sequence Number (dsn) = 5D. It is observed that this number is changing subsequently with the transmission of every packet. If an acknowledgment is requested, the recipient device shall copy the DSN received in the data or MAC command frame into the DSN field of the corresponding acknowledgment frame.

Destination PAN identifier (dustpan) = FF FF . The destination PAN identifier field is 16 bits in length and specifies the unique PAN identifier of the intended recipient of the frame. A value of “0XFFFF” in this field shall represent the broadcast PAN identifier, which shall be accepted as a valid PAN identifier by all devices currently listening to the channel. PAN (Personal Area Network).

Destination address field (addr) = FF FF . The destination address field is either 16 bits or 64 bits in length, according to the value specified in the destination addressing mode subfield of the frame control field, and specifies the address of the intended recipient of the frame. A 16 bit value of “0XFFFF”in this field shall represent the broadcast short address, which shall be accepted as a valid short address by all devices currently listening to the channel.

Message Type (type) = 0A. Active Message (AM) unique identifier for the type of message it is. Typically each application will have its own message type.

Group ID = 7D. Unique identifier for the group of Tmotes participating in the network. The default value is 125(0x7D). Only motes with the same group ID will talk to each other. And after this byte comes the data payload that is also defined by the OscopeMsg structure.

Source Mote ID = 00 00. Source Address which corresponds to the one programmed.

Last Sample Number = 92 18. This number changes subsequently after three packets have been transmitted. As in our case there are three channels defined in our application, so each sample refers to the reading of the three channels. It has been observed that the difference between the previous and next sample is 10, because of ten readings being transmitted.

Channel = 00 00. As it can be observed it changes subsequently with each packet transmitted, we have in this case three channels.

After this field of the message it comes the readings transmitted in endian format which means that for example A7 0F will have to be read really as 0F A7. The oscope component works in a way that fills the packet with ten readings for a channel and then transmits it.

ADC Readings = AC 01 A3 01 10 01 7D 01 BC 01 6F 01 43 01 A6 01 AB 01 1F 01. 10 Readings of 16 bits each one.

2.2.3 Surge Message

Surge is an application that enables multihop routing in the network; the routing algorithm is based on link quality indicator. Routes are chosen depending on the value of this parameter. The message passes from one node to another until reaching destination, it is called multihop because of the number of nodes (hops) will pass a message before reaching target. To make this possible there are two special structures for this type of messages defined by file surge.h and multihop.h, under folders “\cygwin\opt\tinyos-1.x\apps\SurgeTelos\Surge.h” and “\cygwin\opt\tinyos-1.x\tos\lib\MultiHopLQI\Multihop.h”

```
typedef struct SurgeMsg {
  uint16_t type;
  uint16_t reading;
  uint16_t parentaddr;
  uint32_t seq_no;
      } SurgeMsg;


typedef struct MultihopMsg {
  uint16_t sourceaddr;
  uint16_t originaddr;
  int16_t seqno;
  int16_t originseqno;
  uint16_t hopcount;
  uint8_t data[(TOSH_DATA_LENGTH - 10)];
      } TOS_MHopMsg;
```

The following data description has been made using three motes all of them running TelosSurge application under “cygwin\opt\tinyos-1.x\apps\SurgeTelos”. Each node is programmed with a different source address, 0,1,and 2 subsequently. Same Java tool as mention before is used and these are the packets captured.

1st packet

```
14 00 00 00 00 00 7E 00 11 7D 00 00 00 00 00 00 06 00 00 00 00 00 EB 0A 7E
00 61 00 00 00
```

2nd packet

```
14 21 08 7A FF FF 7E 00 11 7D 00 00 01 00 5F 00 5F 00 01 00 00 00 93 0A 00
00 59 00 00 00
```

3rd packet

```
14 21 08 6F FF FF 7E 00 11 7D 00 00 02 00 54 00 54 00 01 00 00 00 8E 0A 00
00 4F 00 00 00
```

Sample Message Hexadecimal Values of 3rd packet

14 21 08 6F FF FF 7E 00 11 7D	00 00 02 00 54 00 54 00 01 00	00 00 8E 0A 00 00 4F 00 00 00
Standard header	**Multihop Message**	**Surge Message**

Descriptions as follows of the first packet, note that the final structure of the data packet is a combination of TOS message, Multihop message and Surge message.

Length = 14. This is 20 bytes is exactly what we have after 7D.

Frame Control Field High Byte (fcfhi) = 00.

Frame Control Field Low Byte (fcflo) = 00

Data Sequence Number (dsn) = 00

Destination PAN identifier (dustpan) = 00 00

Destination address field (addr) = 7E 00

Message Type (type) = 11. Active Message (AM) unique identifier in this case 17 as defined in surge.h.

Group ID = 7D. After this byte comes the data payload that is also defined by the Surge.h structure.

Sourceaddr = 00 00. Address of the previous hop (ignore this)

Originaddr = 00 00. Address of the origin of the message, note that changes consecutively as three nodes are involved in the communication.

Seqno = 00 00. Sequence number of the previous hop of multihop messages (ignore this).

Originseqno = 06 00. Sequence number from the origin of multihop messages.

Hopcount= 00 00. Hopcount.

Type = 00 00. The type of message that indicates the action, known values are:

- SURGE_TYPE_SENSORREADING (0x00) – The message contains sensor data.
- SURGE_TYPE_ROOTBEACON (0x01) –
- SURGE_TYPE_SETRATE (0x02) – Changes the rate a mote will send data.
- SURGE_TYPE_SLEEP (0x03) – Puts the mote to sleep.
- SURGE_TYPE_WAKEUP (0x04) – Wakes mote.
- SURGE_TYPE_FOCUS (0x05) – Causes mote to chirp.
- SURGE_TYPE_UNFOCUS (0x06) – Returns mote to normal (unfocused mode).

No broadcast mode is implemented in actual version of SurgeTelos so the sleep, set rate, root beacon, sleep, wake up, focus and unfocused functions are not possible to be used.

Reading = EB 0A. The sensor data because the first packet corresponds to the base station does not have any sensor data. In the structure this field appears before the Parent Address but has been changed in the program. ADC reading from the sensor.

Parent Address = 7E 00. The address of the parent node this field changes consequently as there three nodes involved in the communications. Address of the parent in the multihop tree

The sequence number = 61 00 00 00. Sequence number of Surge messages

2.3 NESC PROGRAMMING

This section will summarize the practical work carried out programming in NesC language at the early stages of this research. But first it is necessary to give a brief description of NesC aspects, which are presented in the next table.

NesC Concept	Description
Application	A NesC application consists of one or more components linked ("wired") together to form a run-time executable.
Component	Components are the basic building blocks in NesC applications. There are two types of components: modules and configurations. A TinyOS component can provide and uses interfaces.
Module	A component that implements one or more interfaces.
Configuration	A component that wires other components together, connecting interfaces used by components to interfaces provided by others. This is called wiring; the idea is that a developer can build an application as a set of modules, wiring together those modules by providing a configuration. Furthermore every NesC application is described by a top-level configuration that specifies the component in the application and how they invoke one another.
Interface	An interface is used to provide an abstract definition of the interaction of two components. This concept is similar to Java in that an interface should not contain code or wiring. It simply declares a set of functions that the interface's provider must implement—commands—and another set of functions the interfaces' requires must implement—events. In this way it is different than Java interfaces which specify one direction of call. NesC interfaces are bi-directional. For a component to call the commands in an interface it must implement the events of that interface. A single component may require or provide multiple interfaces and multiple instances of the same interface. These interfaces are the only point of access to the component.

Table 11. *Brief Description of NesC main concepts.*

NesC syntax is very similar to C language, every time the system compiles an application there is a C language conversion in the process. This concept is shown below, in Figure 19.

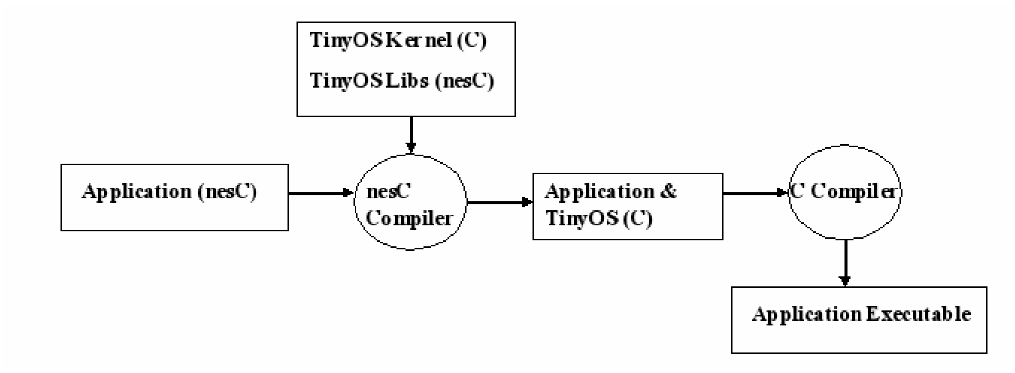


Figure 19 *Description of the compilation process.*

There is not a development environment for NesC like those seen with other programming languages. Some Universities and research communities are currently working on this. A tool called Notepad for programmers is used to develop and create new NesC applications (http://www.pnotepad.org/).

The very first application tried is one called ‘Oscilloscope’; this application comes with the TinyOS distribution. It sends over the radio the readings of all sensors available on the platform to a node connected to the PC and running an application called ‘TOSBase’. The original ‘Oscilloscope’ application was modified producing two new applications first called ‘OscilloscopeIgn’ and the second called ‘Glasshouseadc0’. The first one is like the original but just sends three sensor readings over the radio (Light, internal voltage and internal temperature). The second one is again similar to the original but in this case the light sensor was connected through the external connectors, it also sends the internal temperature and voltage. See the Documentation CD for more detail about these files, next it is shown the header file that makes possible to connect a sensor to the external connectors. This opens the door to use any type of sensor or a sensor board with multiple sensors.

```
#ifndef _H_External_h
#define _H_External_h

#include "MSP430ADC12.h"

// PAR, Photosynthetically Active Radiation connected to ADC0
enum
{
  TOS_ADC_PAR_PORT = unique("ADCPort"),

  TOSH_ACTUAL_ADC_PAR_PORT = ASSOCIATE_ADC_CHANNEL(
    INPUT_CHANNEL_A0,
    REFERENCE_VREFplus_AVss,
    REFVOLT_LEVEL_1_5
  )
};

#define MSP430ADC12_PAR ADC12_SETTINGS( \
           INPUT_CHANNEL_A0, REFERENCE_VREFplus_AVss, SAMPLE_HOLD_4_CYCLES, \
           SHT_SOURCE_ACLK, SHT_CLOCK_DIV_1, SAMPCON_SOURCE_SMCLK, \
           SAMPCON_CLOCK_DIV_1, REFVOLT_LEVEL_1_5)

#endif//_H_Hamamatsu_h
```

The following graph was taken as a result of an experiment, where three motes were used. One of the Tmotes was programmed with ‘OscilloscopeIgn’ and the other with ‘Glasshouseadc0’ (Documentation CD), both units using Photo synthetically Active Radiation light detector. In one of the units is connected to the PAR holder in the board and in the other to ADC0 using external connections. The third mote connected to the PC is running TOSBase, this is the graph displayed using Java application called ‘oscilloscope.java;’ which comes in any TinyOS distribution.

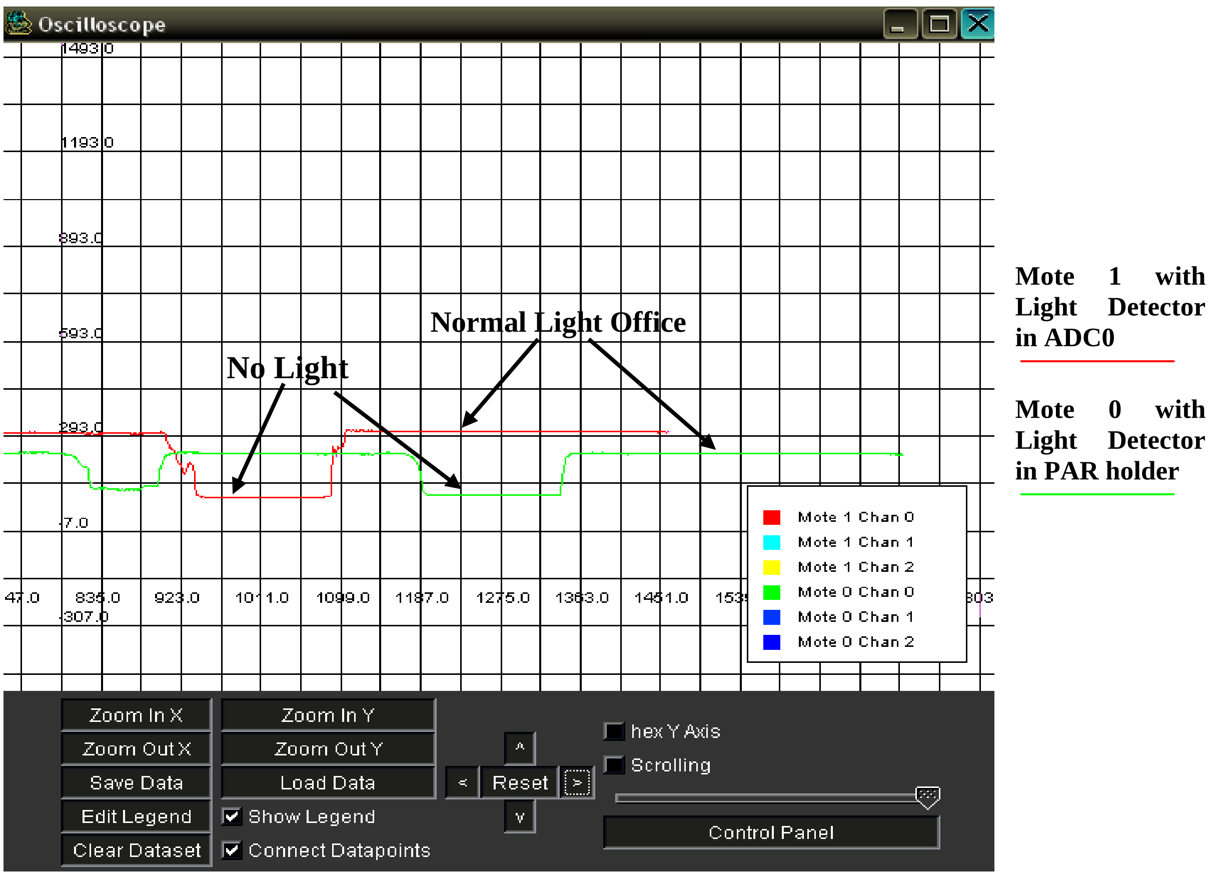


Figure 20. *Experiment results with two motes with light detector sensors connected in different places in the TmoteSky hardware platform.*

Each sensor reading is the result of the ADC sampling process using 12 bits; in this case the value of the light detector when there is no light is 47. According to the previous header file reference voltage is set to 1.5 volts, sampling frequency is 8 KHz.

The following expression can be used to calculate the current generated by the sensor and going through the resistor and ADC0.

$$I\,(A) = \text{value} * (\text{vref}/4096)\ 10^{-5}$$

The internal voltage sensor uses the microcontroller's 12-bit ADC. To convert the raw value of the ADC to the corresponding voltage, perform the calculation:

Internal Voltage (V) = value/4096 * Vref where Vref = 1.5V

The internal voltage sensor monitors Vcc/2, so multiply the resulting voltage value by 2 to get mote's supply voltage (Vcc). Similar to the internal voltage, the internal temperature sensor is an un-calibrated thermistor that is sampled using the microcontroller's 12-bit ADC. Note that the sensor is not calibrated, although the following formula works well for most applications:

T = (Vtemp - 0.986)/0.00355 Temperature is specified for degrees Celsius.

The following figure displays an application called ‘Trawler’ which comes in TinyOS distribution released by Moteiv. The display window represents the network topology with link quality indicators, the numbers below each branch or route of communication. Node 0 is the base station connected to the computer, node 1 is the unit with external antenna connected and node 2 is the unit with onboard antenna. All three nodes are running an application called ‘Delta’, which is similar to Oscilloscope with multihop capabilities; it will be described in more in next section. Packets are routed through the network from node to node to the Base Station, which is the node connected to the PC. This application comes with Moteiv’s TinyOS distribution. Figure 21 is with the three nodes together within a distance of less than 1 meter to the base station (node 0). Figure 21-b the nodes 1 and 2 are separated from base station (node 0) a distance of 25 meters approximately.

Figure 21-a

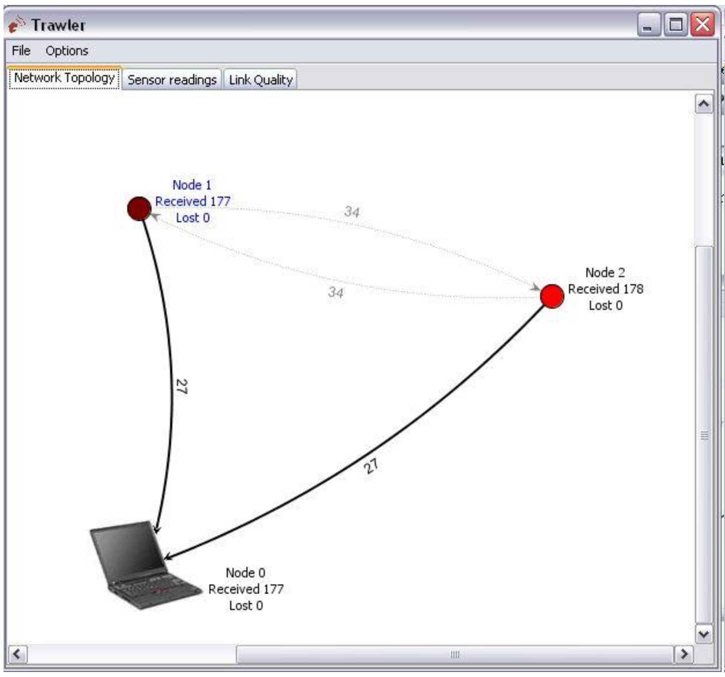


Figure 21-b

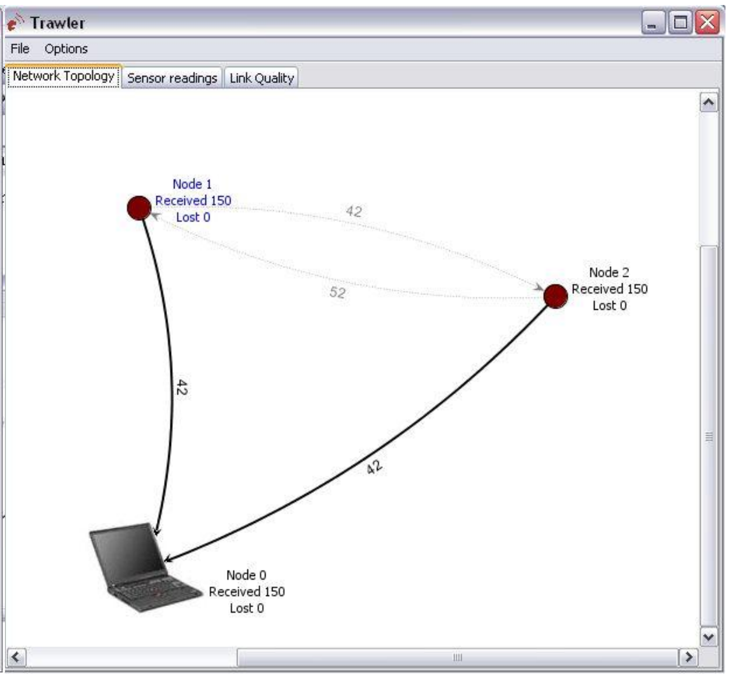
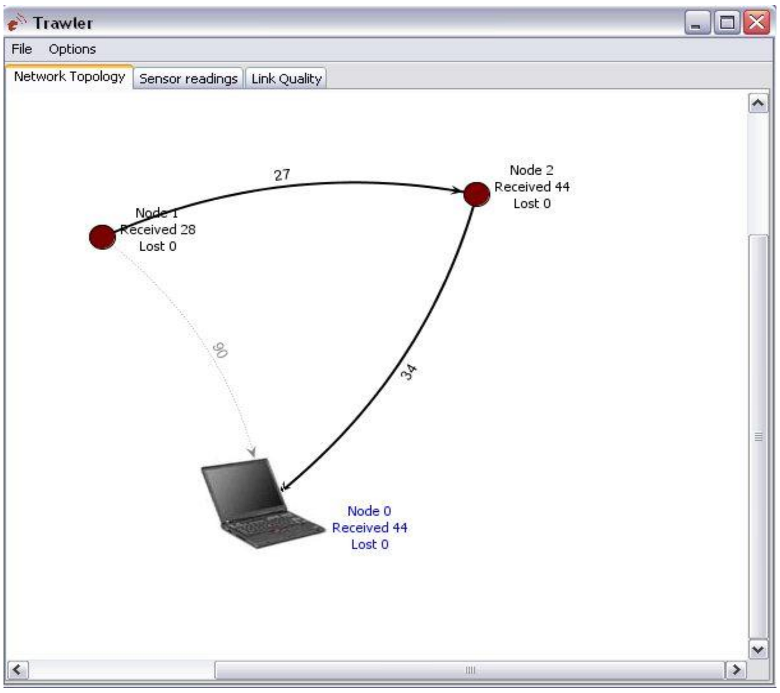


Figure 21-a. *Nodes 1 and 2 within 1 meter from base station*. **Figure 21-b.** *Nodes 1 and 2 at 25 meters from base station.*

Link Quality Indicator parameter also known as LQI is described in standard 802.15.4. The LQI measures the received energy level and/or SNR for each received packet. When energy level and SNR information are combined, they can indicate whether a corrupt packet resulted from low signal strength or from high signal strength plus interference. In the standard it is defined as a value between 0 and 255, being the latest one sign of good link conditions and signal to noise ratio (SNR). In the Trawler Java application has been observed that is the opposite of the standard definition, a low value indicates good transmission conditions while bigger values poor link quality indicator and SNR.

An experiment consisting of connecting a small solar cell panel to the board was carried out. The solar cell panel provided 3V under clear sky conditions. The application is the same as in the previous figure. This time node 1 is in a distance within 25 meters from node 0 and node 2 is in a distance within 100 meters from node 0.



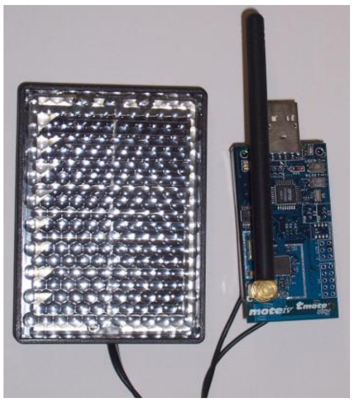


Figure 22. *Experiment using solar cell panel connected to node 1.*

And this completes the early experiments with the hardware platform chosen and programming in NesC.

2.4 DEVELOPMENT OF SOFTWARE TOOLS

The language used in TinyOS to develop the software tools in charge of post processing the data coming to a central unit connected to a PC is Java. Although there are not consistent tools to collect, store and display the data coming from the network in TinyOS. All developers have at the moment is a collection of basic functions, and modules already written that can make possible the development of more complex applications. An attempt to produce a C++ simple GUI proved to be a very difficult task due the lack of any work in this direction and then discarded as an option for this project. Basically the most difficult part was to create a complete new interface to handle communications with this hardware platform using the serial port. The existence of this interface written in Java ready to be used and easy to modify demonstrated to be a more convenient option, taking into account the time scale to complete this project.

The first tool created is called ‘tools.java’ which makes it easier to load other useful existing Java tools such as ‘oscope.java’ and ‘messagecentre.java’, a shortcut is created which can be accessed from the desktop. This shortcut sets the cygwin environment for the user; it detects the mote connected to the PC and sets the serial COM port to point to that mote and starts the application.

In the next chapter more customized applications and tools designed to carry out some of the practical work are described in more detail. The software written in Java has been developed using Netbeans IDE (http://www.netbeans.org/).

3 EXPERIMENTAL WORK AND RESULTS

This section will cover the experimental work carried out using this new technology. The starting point is the set up of a trial network measuring light in the Glass houses at Silsoe, then the setup of a network where nodes will have an existing sensor developed at Cranfield University, and at the end a transmission range test.

3.1 GLASS HOUSE TRIAL NETWORK

The aim of this experiment is to set up a trial network and evaluate how it communicates to the master PC using existing tools. It will also outline the battery consumption of the nodes when transmitting data constantly. A simple practical example like measuring Photosynthetic light emissions in a Glass House which could serve as a survey to optimize the growing of plants inside. Photosynthetic light has the physical characteristic of occurring in the region used by plants for photosynthesis. This is in the wavelength range of 400nm to 750 nm. The simplicity comes because the hardware platform has already the facility of connecting a sensor of this type. Therefore there is no need of adding any extra circuitry to condition the sensor signal, keeping the design as it comes from manufacturer. Some of the applications exposed previously will be used in here.

The network proposed will consist of 5 units, 4 units deployed inside the glass house and one of them acting as a base station connected to a computer in the office next to the network. All of them will be fitted with a Photosynthetic led diode detector as mention earlier.

In three of the nodes the external connectors provided will be used, this will serve as an experience to find out how any other type of sensor could be fitted to the board, and how the software needs to be modified to meet this setup.

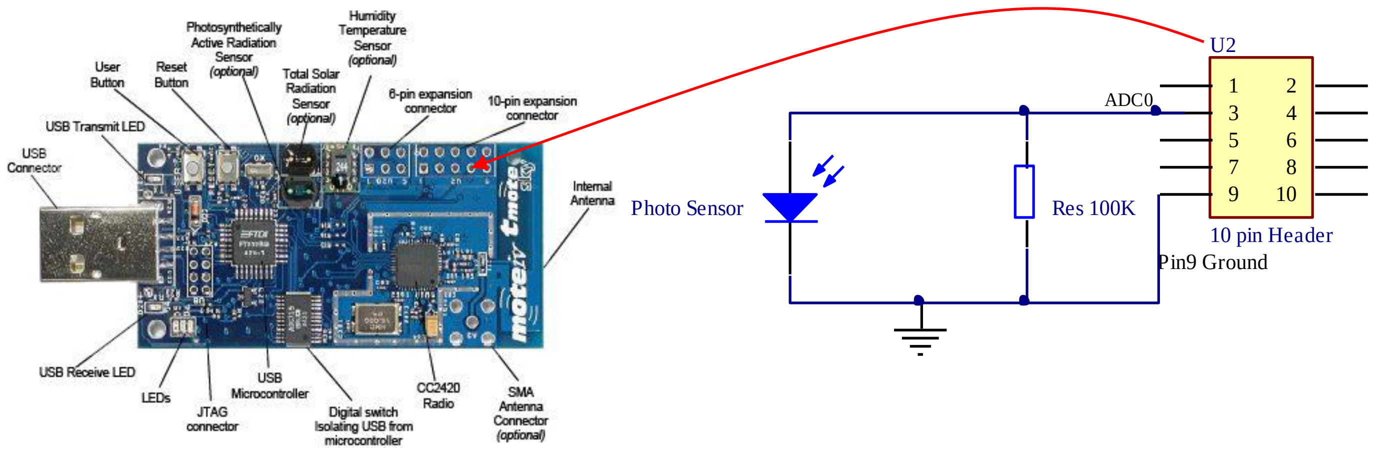

Photosynthetically Active Radiation Sensor (optional)
Humidity Temperature Sensor (optional)
User Button
Reset Button
Total Solar Radiation Sensor (optional)
6-pin expansion connector
10-pin expansion connector
USB Transmit LED
USB Connector
Internal Antenna
USB Receive LED
LEDs
JTAG connector
USB Microcontroller
Digital switch Isolating USB from microcontroller
CC2420 Radio
SMA Antenna Connector (optional)
U2
ADC0
Photo Sensor
Res 100K
1
2
3
4
5
6
7
8
9
10
10 pin Header
Pin9 Ground

Figure 23. *External Sensor connection schematics.*

Next pictures will show the scenario and location of each node of the network, one of them (node 4) will have an external antenna, the rest will be using the circuit printed antenna. There will be also one running with a solar cell panel attached to show how it performs compared to the others. The node chosen for this will be node 1 as the glass house is oriented to the sunset, so it will be the last one with enough light till complete darkness.

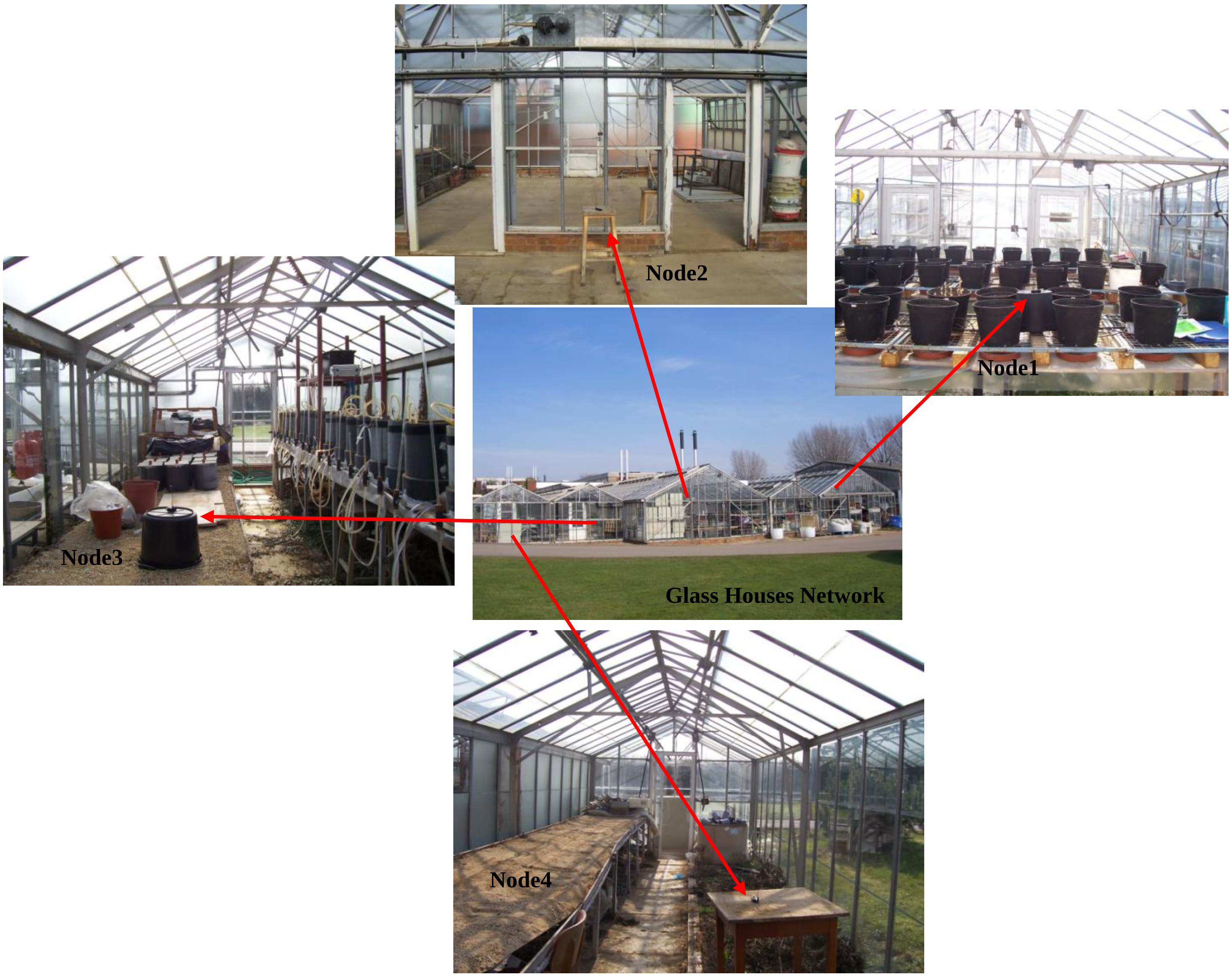


Figure 24. *Location of the nodes of the glass house trial network.*

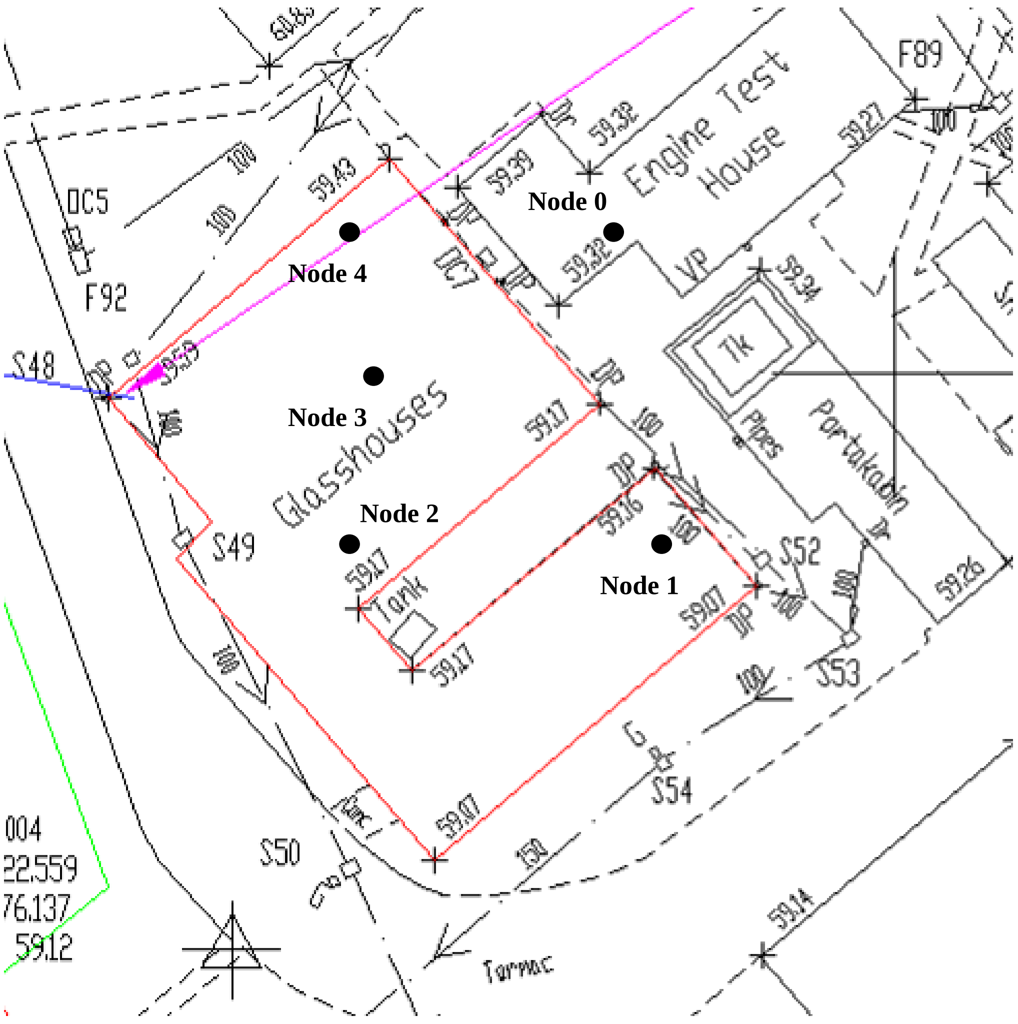


Figure 25. *Location of the nodes of the glass house trial network, marked with black dots.*

When locating the nodes it has been preferred to locate them in different houses than to form a network inside one of them. The reason for this is the belief that there will be some distance between them to test the quality of transmission and also to measure more light variations. First experimental results have to do with the quality of transmission within this network configuration. There are three different groups of software applications tested in this experiment.

The first experiment is carried out programming all nodes with an existing application of TinyOS distribution called 'SurgeTelos'. This program implements a communication protocol based on a tree algorithm routing, which relies on two pieces of information: a parent node identifier, and a hop-count or depth from the tree root. A routing tree is built via local broadcast from the root followed by selective retransmission from its descendents. A node routes a packet by transmitting it with the parent as the designated recipient. The parent does the same to its parent, until the packet reaches the root of the tree for more details see (Philip Levis, *et al*. 2004).

This application has been tested in order to display the network topology form by the nodes participating in the network using ‘Surge’ a Java tool available in TinyOS distribution. The next figures will show how the network topology is configured in the glass house trial network. Several figures are shown as network topology changes in time according to the variation of the link quality indicator.

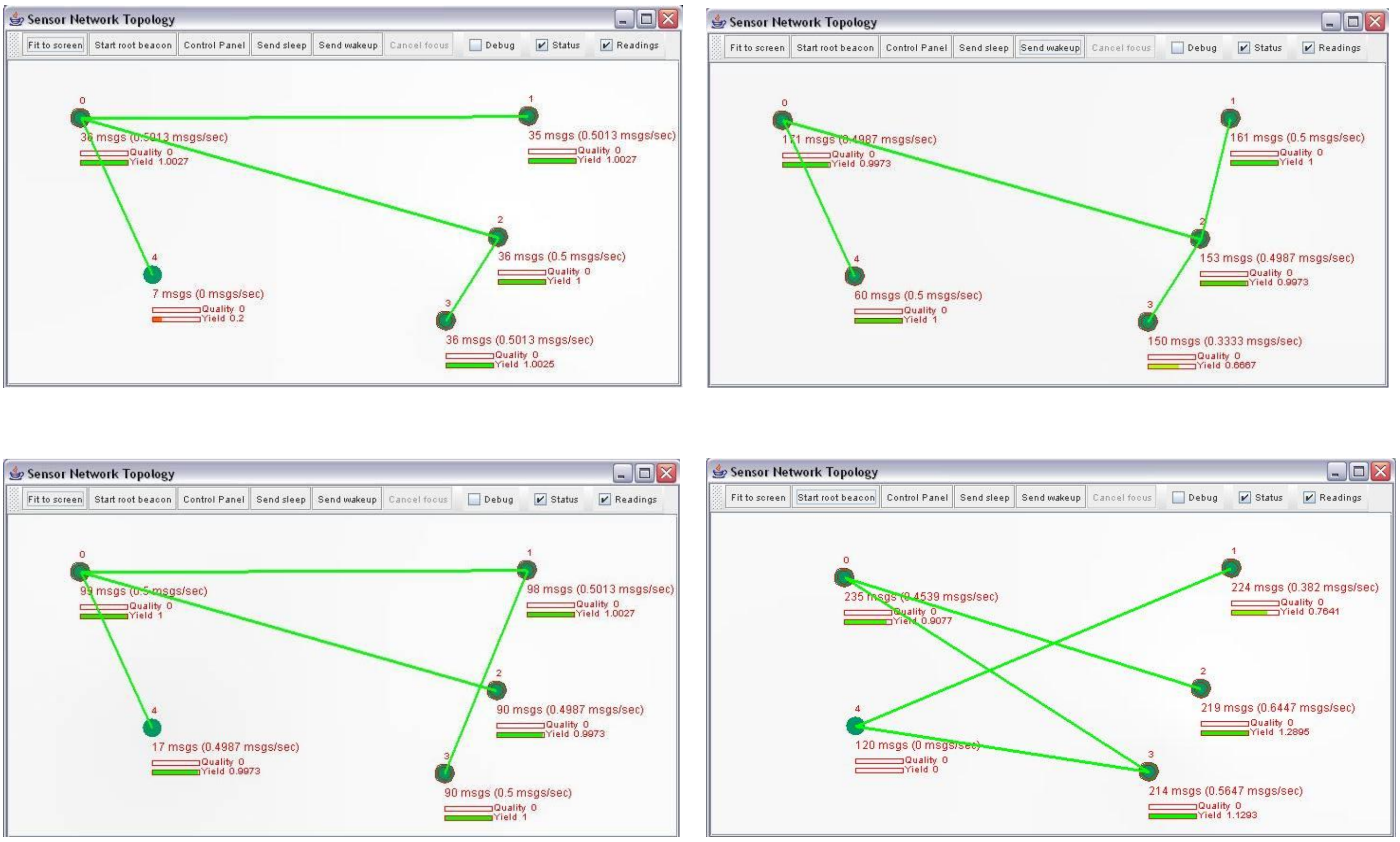


Figure 26. Glass House trial network topology configurations

These results have been taken during a period of 20 minutes restarting every 5 minutes Node0, which forces the network to calculate and establish new routes. All nodes are running 'SurgeTelos' application and assigned when programming them with an identifier number. Node0 is the one connected to the PC, location of nodes same as the one exposed in figure 25. Node 2 transmission performance is very poor compared to the rest of the nodes. Looking at the deployment in figure 25, is further from the base station (node 0) than any other node. The distances in between nodes and in between nodes and the base station are not greater than 25 metres in any case, which is bellow any maximum distance recommended in the technical specifications. The explanation to the bad performance comes when taking into account the obstacles and metallic structure within the glass house in between that node and the base station. The first node is closer to the door of the glass house, the electromagnetic waves after this obstacle travel under free space conditions until hitting the wall of the Engine house where the node 0 is connected to the laptop.

This second experiment involved two applications called ‘DeltaIgnacio’ and ‘DeltaIgnacioPAR’, which are a modification of original ‘Delta’ application that comes with Moteiv’s TinyOS distribution. Delta implements a communication protocol called Sensornet Protocol (SP). This protocol is based in three main operations, data transmission, data reception and neighbour management. This last operation is possibly the main difference respect to the previous one ‘SurgeTelos’. In this case there is not one parent node to transmit the message. Each node has a neighbour table where it keeps information about link quality and power scheduling (when the node should be awake or asleep). Transmission of data it also quite different as in this protocol there is a message pool where messages are left by nodes, and messages can contain information about the status of other nodes for more details see (Joseph Polastre, *et Al*. 2005).

The software called ‘DeltaIgnacioPAR’ is a modification of original Delta which sends the internal temperature. Therefore it was changed to send the readings of a sensor connected to the PAR holder on the board. And ‘DeltaIgnacio’ was tailored to send the readings of a sensor connected to the ADC0 channel. This makes necessary to rewire the current component driving the internal temperature sensor to the sensor used in each case. As mentioned earlier all units are fitted with Photosynthetic Active.

The Java tool used on the PC to display the following graphs is called 'Trawler', which comes with Moteiv's TinyOS distribution.

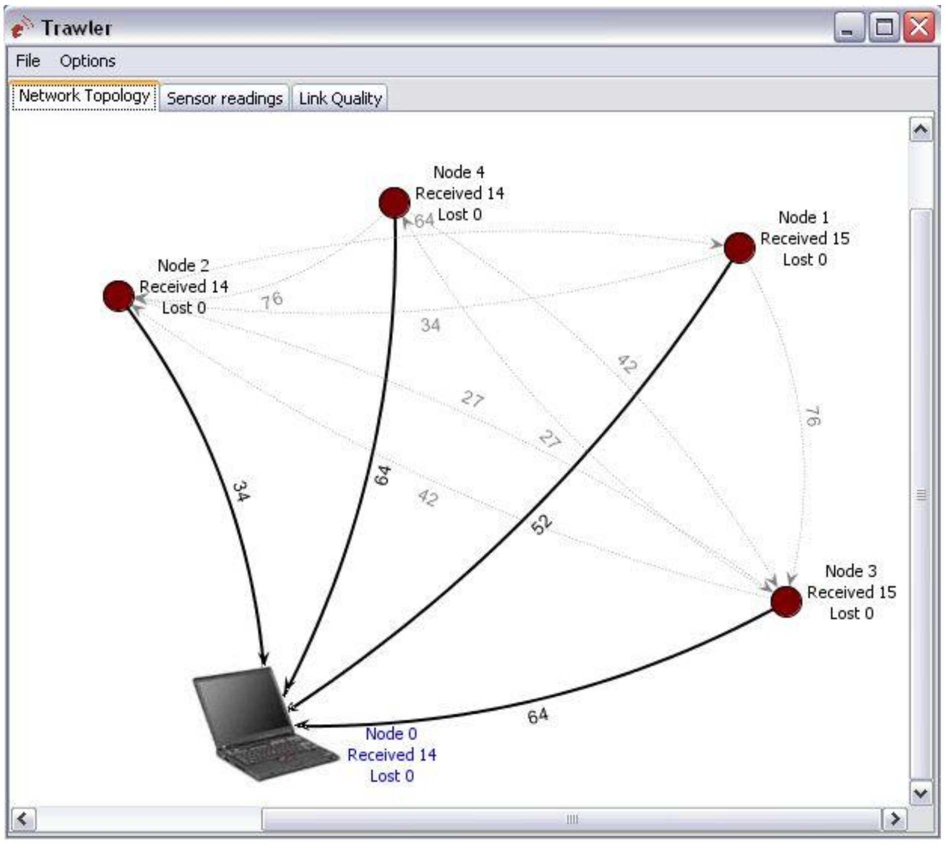


Figure 27. *Measurement result for LQI in glass house network.*

Final experimental work is carried out programming the nodes with 'OscilloscopeIgn' and 'Glasshouseadc0'. The application called 'Tools.java' was used to capture data from time to time. A node connected to the PC running 'TOSBase' was in charge of handling the data coming from the network and passed to the PC through the serial port. The network has been running three days and over a weekend. During the first day node 1 was running on batteries and the other two days was connected and powered by a solar cell panel.

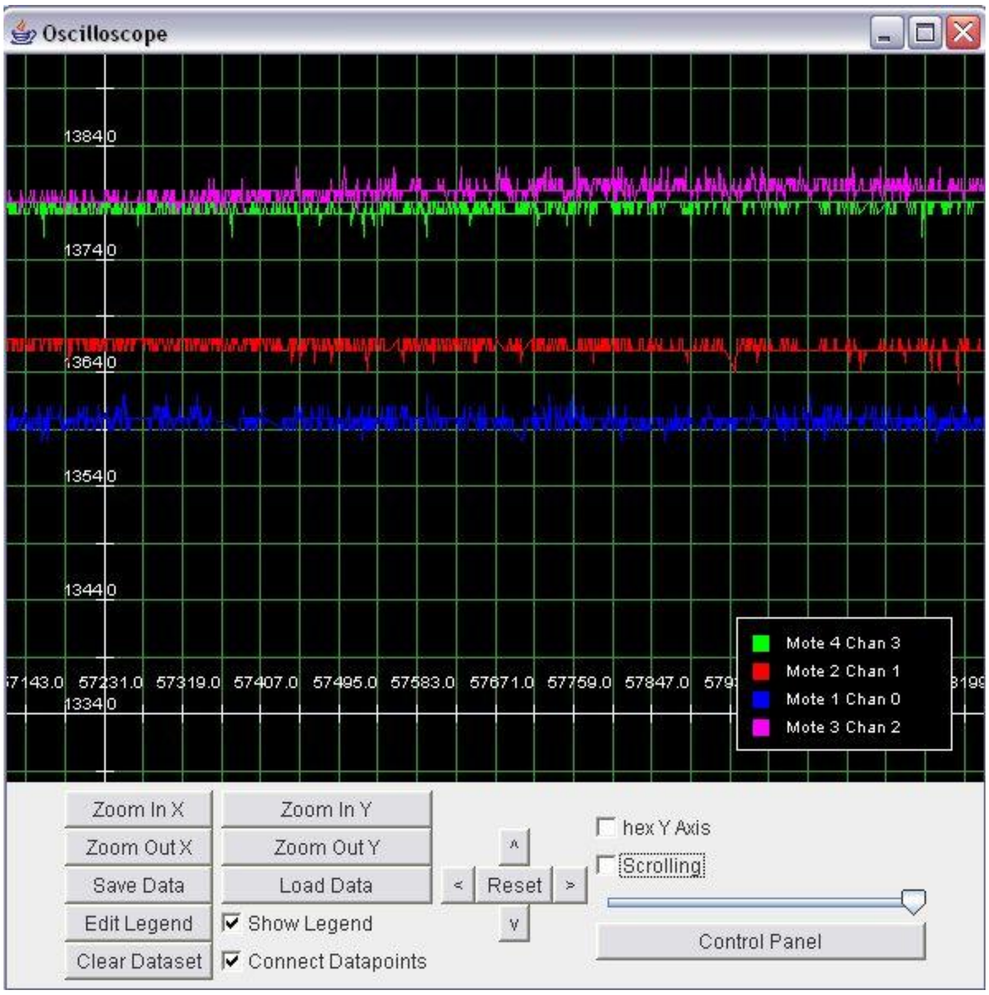


Figure 28. *Sensor readings from day1 at 18:32.*

As explained previously Osiloscope.java is a tool to display the sensor readings coming from a network node connected to the PC and acting as a gateway. Oscilloscope needs another java component to be running which is called ‘Serialforwarded.java’, which acts as an interface. Y-axis represents the ADC counts of the sensor readings sampled and the X-axis the sample number.

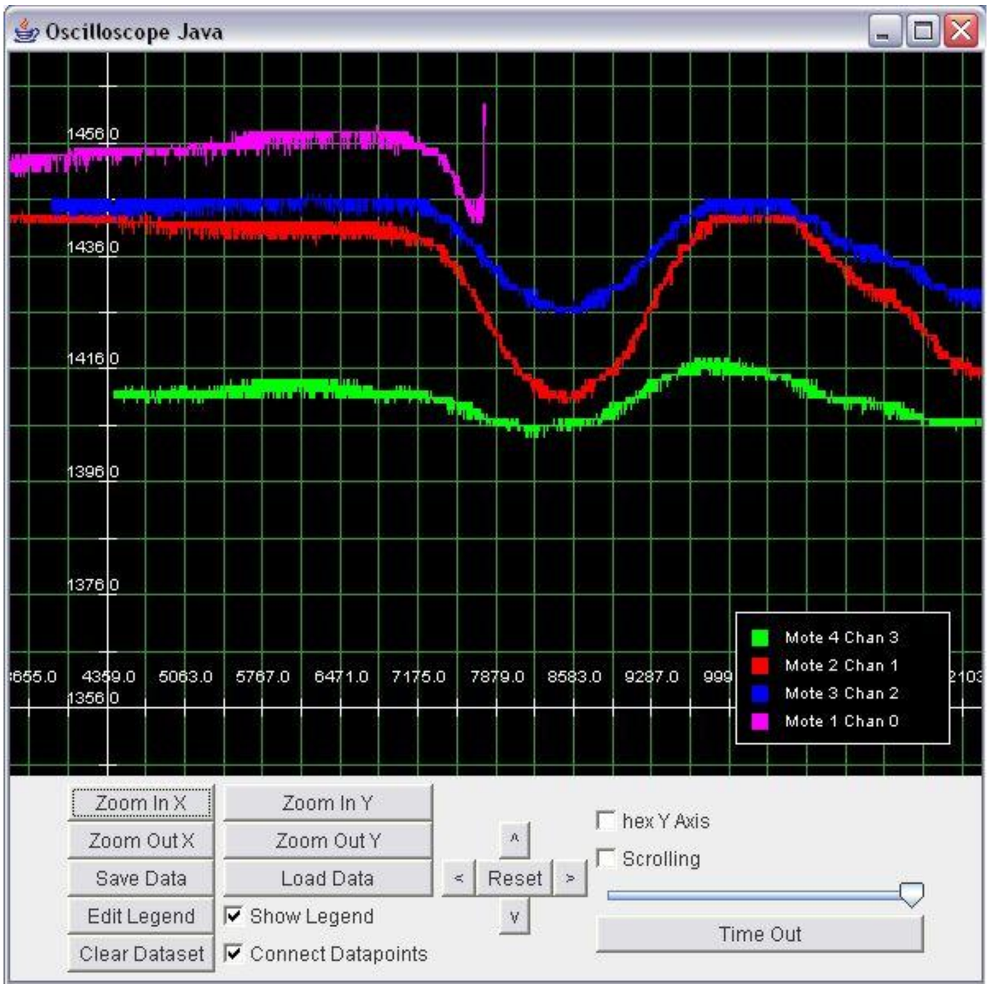


Figure 29. *Sensor readings from day2 at 18:14.*

Transmission link to node 1 is lost as there was not enough sun light to power the node.

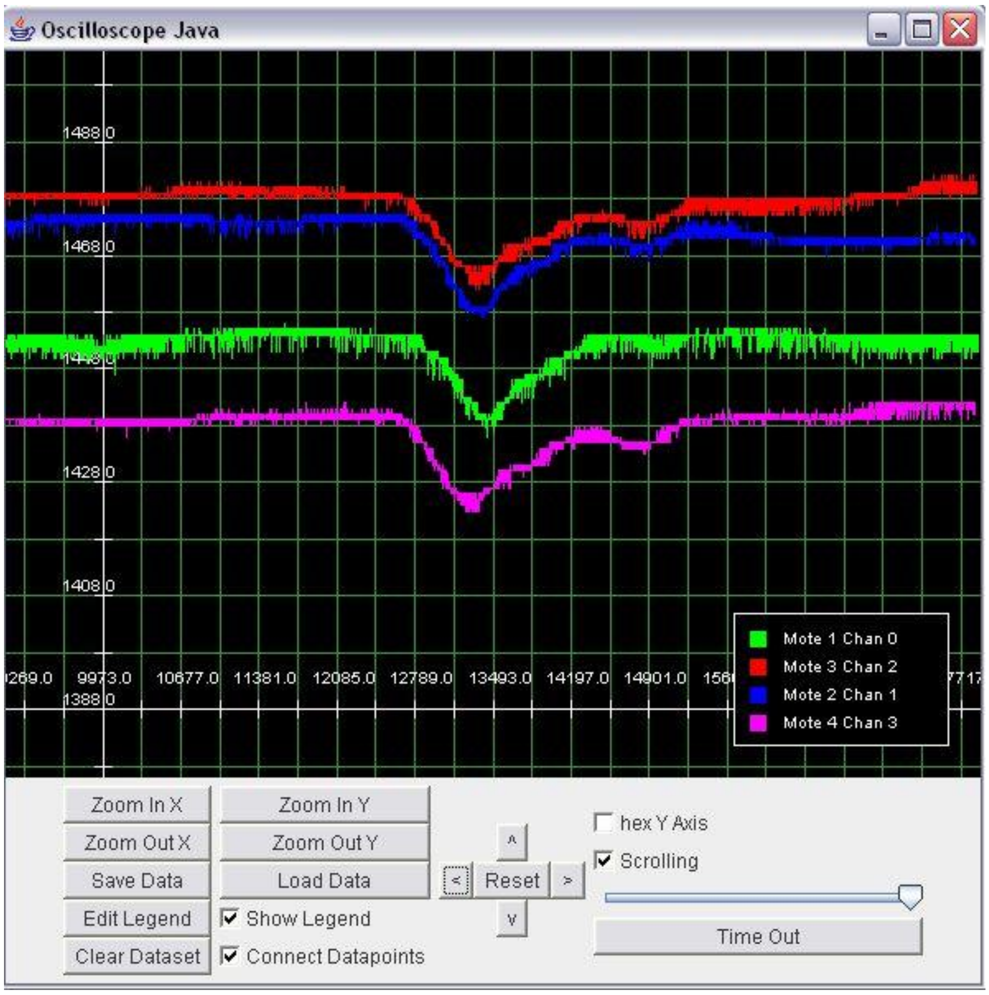


Figure 30. *Sensor readings from day2 at 16:44.*

These are just some of the graphical results obtained the whole set of graphs can be found in the documentation CD accompanying this report. They display the readings of photosynthetic light emissions during the period of three days. The graphs were taken randomly and in some cases where light was obstructed by clouds to show the variations of the sensor readings, which proved to be very sensitive. There is another graph displayed here with the readings of light when no light is present, measurement taken inside office, covering the sensor temporary with a piece of cardboard.

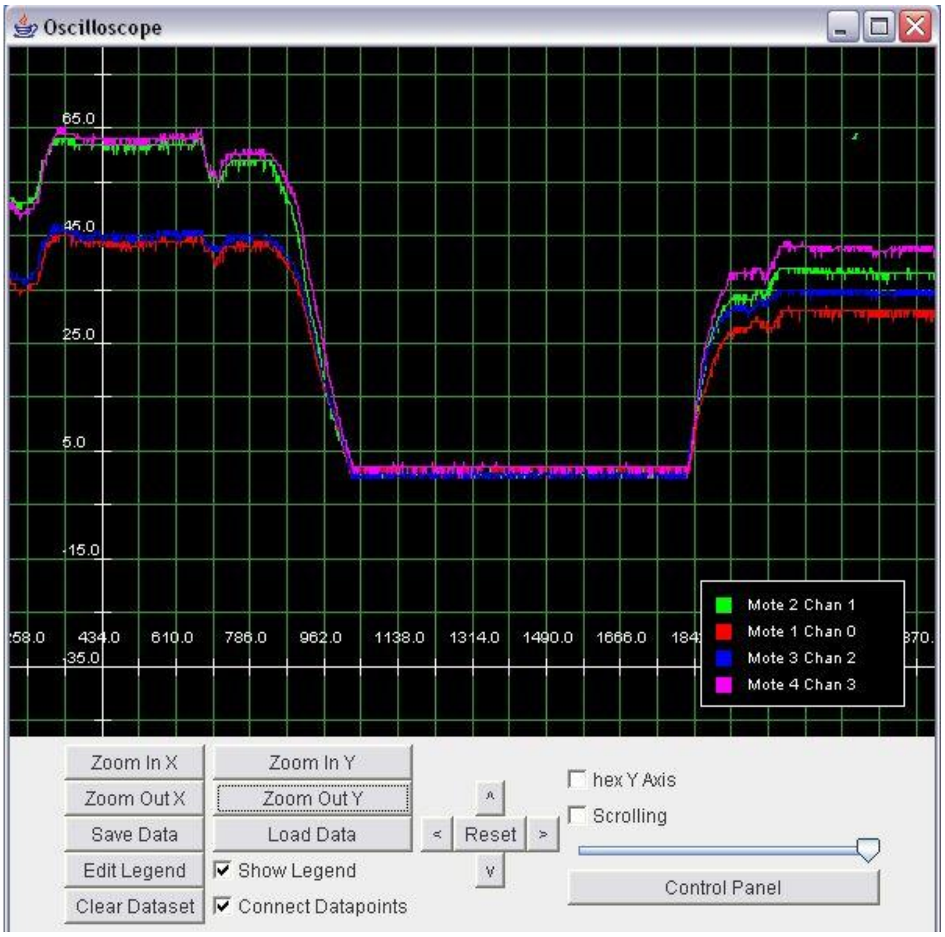


Figure 31. *Sensor readings when no light is present.*

3.2 OIL SENSOR EXPERIMENTS

This section describes experimental work using the oil detector sensor developed at Silsoe (Ritchie, L, *et al*, 2000). Connection of the oil detector sensor to a Tmote Sky unit requires a conditioning signal circuit and the development of new TinyOS applications. The aim of this practical work is to test the hardware platform and software in a real scenario, an Oil Leak Detection system using wireless sensor networks technology. Next picture shows the oil detector sensor used.

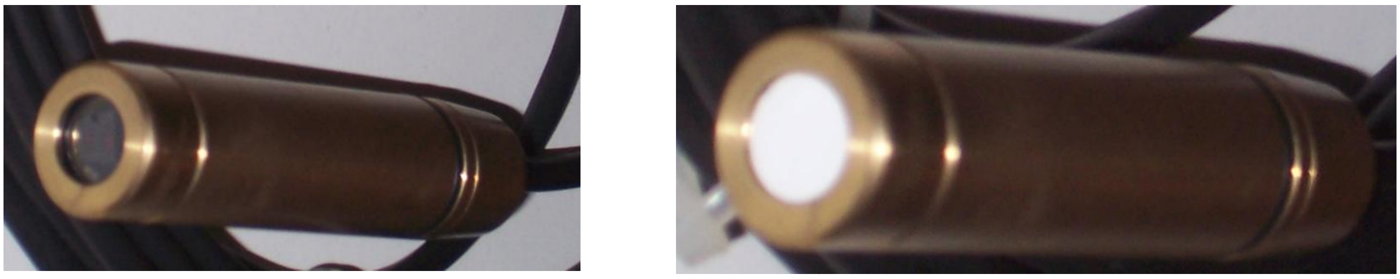


Figure 32. Oil detector sensor with and without the membrane**.**

Basically the oil detector sensor is based on an infrared emitter and detector, which is covered with a membrane as shown in the above picture. Infrared light reflected on the target is measured to determine if there is the presence of oil. When the top of the sensor is cover with the membrane the reflection measured is constant, if the membrane enters in contact with oil it becomes transparent and then a sudden change in the reflection measured occurs. The sensor output is connected to the ADC (analogue to digital converter, channel 1) of the TmoteSky.

The following sensor board was designed to accommodate the readings coming from the oil sensor and to drive the sensor from the software application running in the unit. It sits on top of the hardware unit therefore headers are named the same as in the unit schematic.

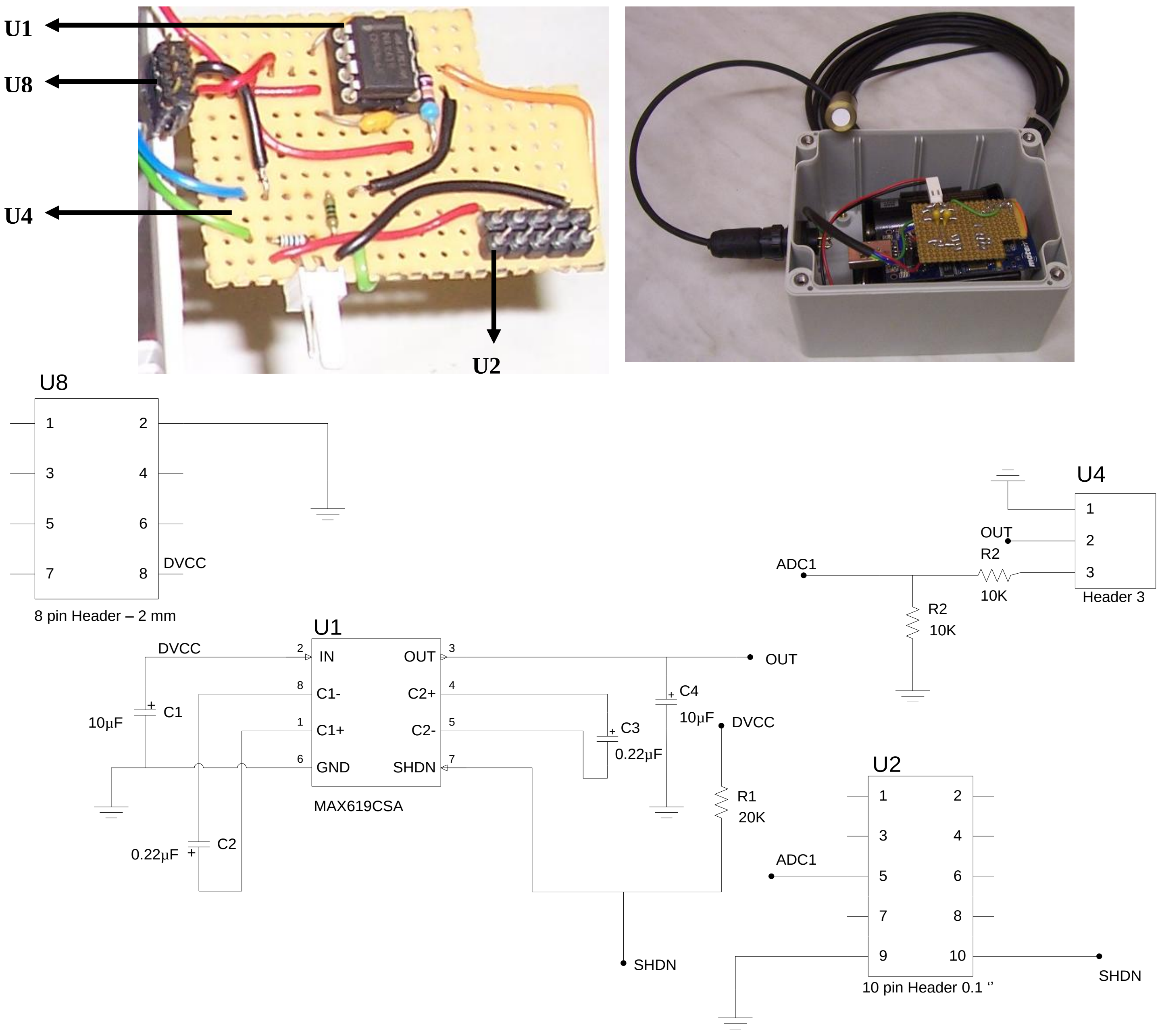


Figure 33. *Sensor board pictures and circuit schematic.*

Basically the oil sensor needed an input voltage to power it greater that the one the unit can provide using header U8. Therefore a Maxim integrated circuit (MAX619CSA) was chosen to double the voltage available. See contents of Documentation CD for more technical specifications details. This circuit also adds the possibility of shutting down the sensor on demand which it is quite useful when the readings are not needed using 'SHDN' input signal. This can be implemented in the software and it is in line with one of main aims of using this technology which is to save as much power as possible. For these experiments the sensor is active all the time.

Two programs were developed for these experiments with the oil sensor; they are called ‘DeltaOil’ and ‘DeltaOilv1.0’. The first one is a modification of original ‘Delta’ with the possibility of connecting the sensor board designed to the hardware platform. An interface is needed to make ADC1 an input for this sensor. The second application introduces the feature of sending the readings when they are above a threshold value. The setup of ADC1 is defined in a header file called ‘ExtenalOil.h’, see documentation CD for full programs listings.

```
#ifndef _H_ExternalOil_h
#define _H_ExternalOil_h

#include "MSP430ADC12.h"

// Oil leakeage detector sensor connected to ADC1
enum
{
  TOS_ADC_PAR_PORT = unique("ADCPort"),

  TOSH_ACTUAL_ADC_PAR_PORT = ASSOCIATE_ADC_CHANNEL(
    INPUT_CHANNEL_A1,
    REFERENCE_VREFplus_AVss,
    REFVOLT_LEVEL_2_5
  )
};

#define MSP430ADC12_PAR ADC12_SETTINGS( \
           INPUT_CHANNEL_A1, REFERENCE_VREFplus_AVss,
SAMPLE_HOLD_4_CYCLES, \
           SHT_SOURCE_ACLK, SHT_CLOCK_DIV_1, SAMPCON_SOURCE_SMCLK, \
           SAMPCON_CLOCK_DIV_1, REFVOLT_LEVEL_2_5)

#endif
```

Next figure shows graphically the working procedure of this sensor and how sudden is a change in the reflection measured when the membrane enters in contact with Oil. Two units are used to obtain this graph - one running ‘DeltaOil’ and the other running the same application but numbered as 0 when programming the unit. Doing this the unit is set automatically as the gateway between the network and the end user PC. The Java tool used to display the data is called 'Trawler'.

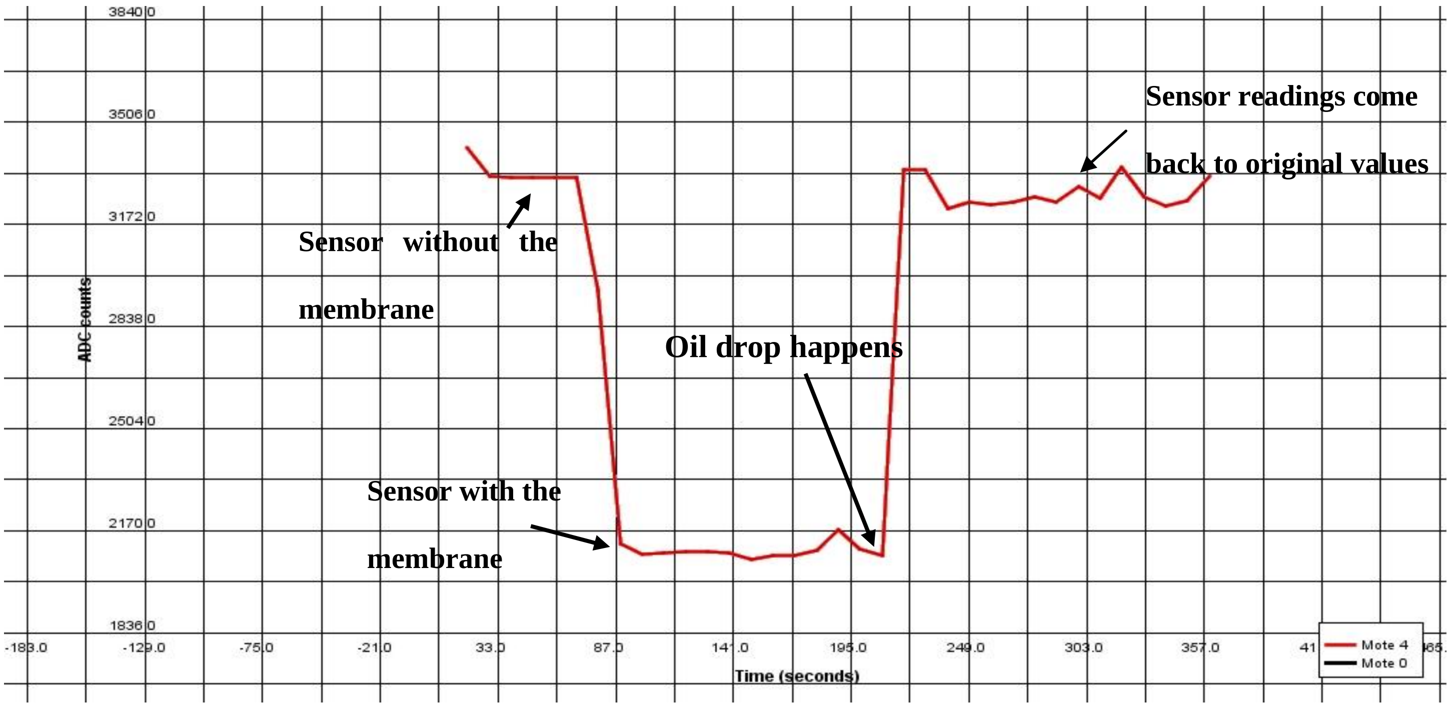


Figure 34. Oil Detector sensor response in different situations.

As it can be appreciated from the graph above where values from the sensor are represented there is a big step in values when the sensor has the membrane and when it does not or this becomes transparent as a consequence of dropping some oil on it. Next figure shows the same experiment this time with three units forming the network. The three of them running 'DeltaOil' and one acting as the base station, a tiny amount of oil is dropped into each sensor consecutively. It can be observed from both graphs shown how fast is the response captured by the unit.

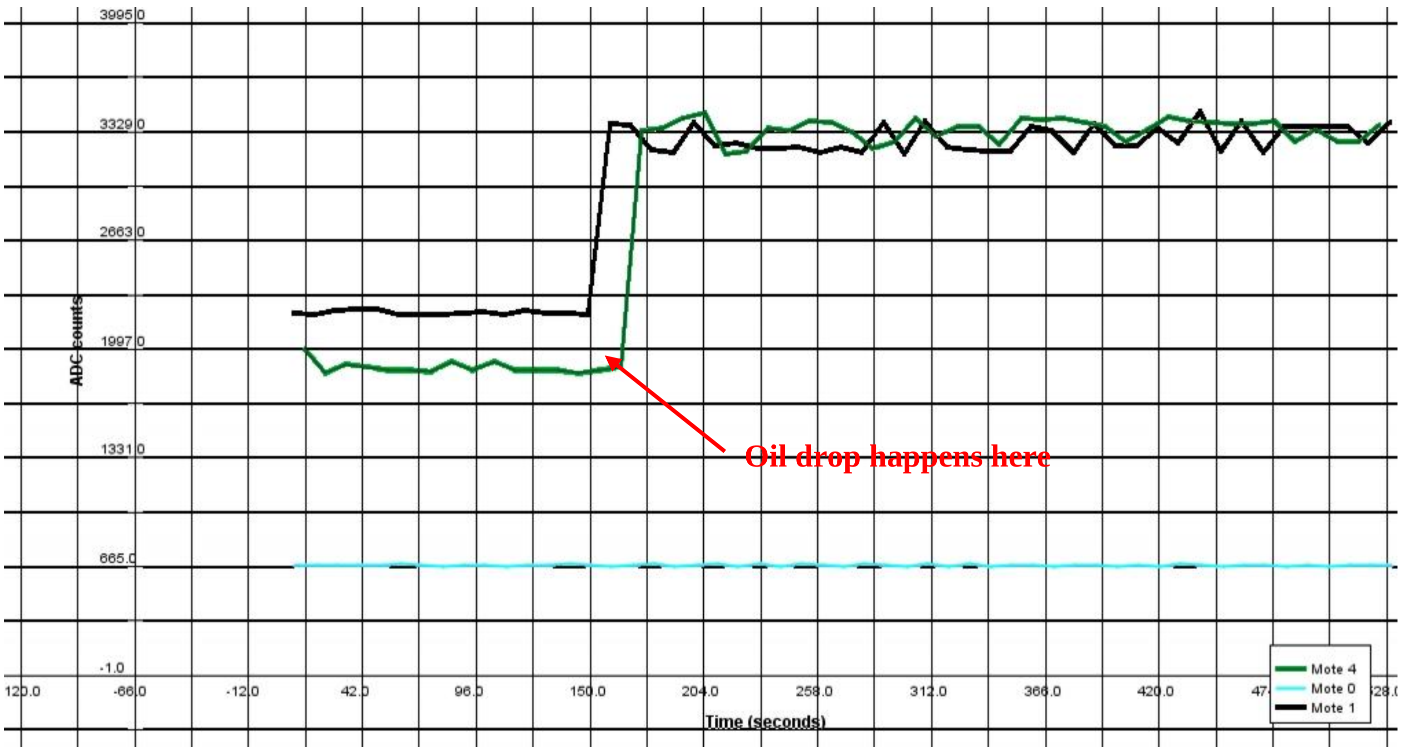


Figure 35. Oil detector sensor response when three motes are participating in the network.

The idea of setting a threshold value in the code is to make the unit transmit only when oil is detected. This practice in conjunction with toggling the sensor ON and OFF using the shutdown line mentioned before is in line with saving as much power as possible. This is a feature that this technology is trying to put forward as an advantage compared to other current monitoring systems. Therefore a value of 2504 is set as threshold. This value is far above from the level recorded when the oil sensor has an intact/cleaned membrane (see Figure 34). The part of the code responsible of controlling when data is ready for a transmission and the graph obtained is shown.

```
async event result_t ADC.dataReady(uint16_t data) {

    m_adc = data;

call  P63.setHigh();//P63 corresponds to Pin10 in Header U2 which turns
on the Sensor

    if (m_adc>=0x9c8) {   //setting a threshold value adccounts = 2504

        call Leds.greenOn();
```

```
              post sendData();
           }
        else {
              call Leds.greenOff();
        }
            return SUCCESS;
          }
```

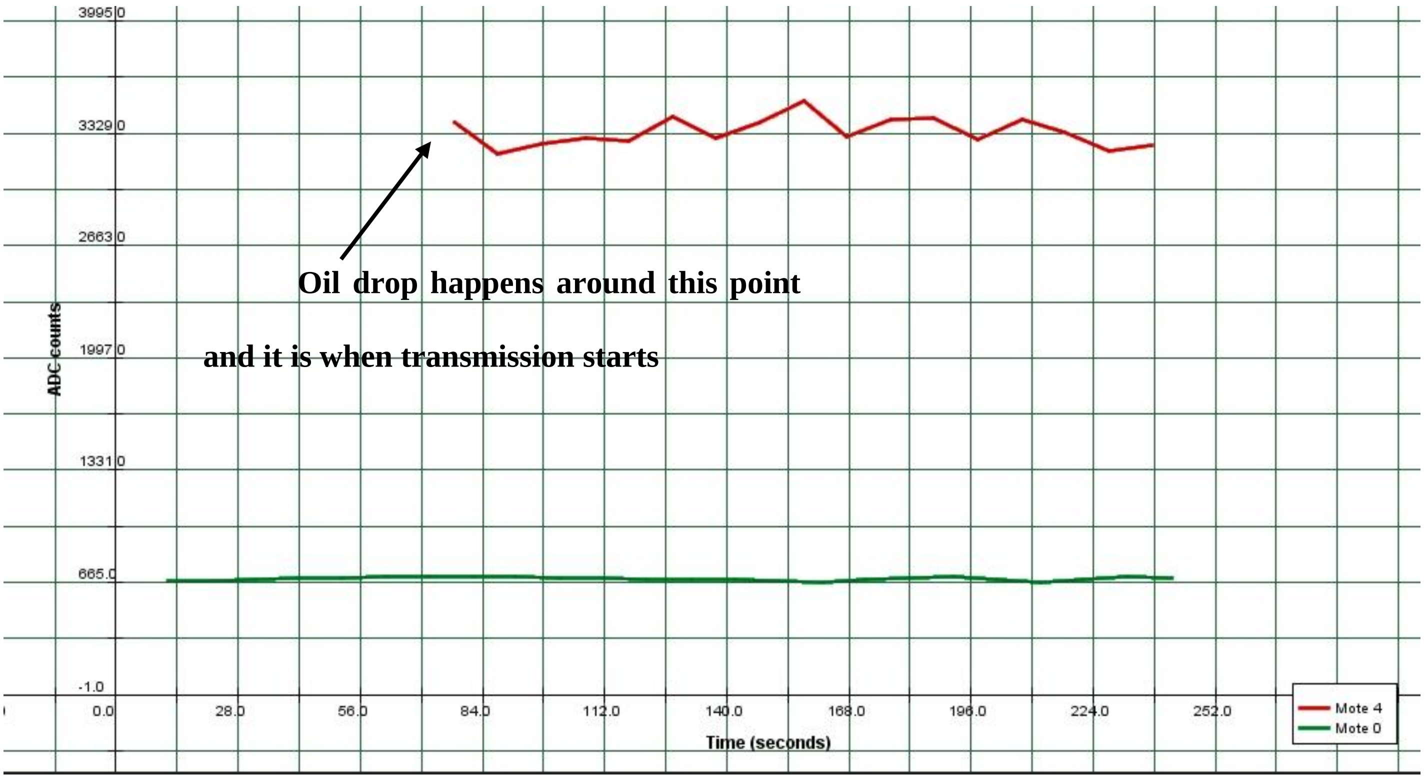


Figure 36. Graph results when transmission happens when significant data is captured.

As expected the node does not start transmitting until threshold level is not reached. The ADC is being read at a constant rate, and it has been set up like this for this experiment because the main aim was to demonstrate a simple modification in the code to avoid transmitting redundant information. In the future this rate can be delayed in order to save power, putting the node into sleep mode at scheduled intervals, and read at certain intervals and if proceed then transmit the information back to the base station.

3.3 SIGNAL STRENGH EXPERIMENTS

The last section has to do with the practical work carried out to determine the range of the hardware platform chosen. According to manufacturer and technical specifications the maximum transmission range distance from one unit to another should be 125 metres. A new graphical user interface is designed to display the readings coming from the network.

Full details of programs and graphs can be found in the CD submitted with this report; here there will be a brief description to outline the findings of doing such an experiment. In terms of hardware there is not any extra circuit needed in this case. Because the idea is just to transmit a packet from one unit to another used as a target with information about the Signal Strength Indicator (RSSI) and the Link Quality Indicator (LQI) and wait for the response from the target with the same information. The software in this case is form by an application called 'Pong' and the Java tool specially designed for this experiment is called 'RSSI'.

The reason for producing a new Java tool instead of re-using any of the existing ones is because 'Pong' application uses its own messages types. The work carried when studying the types of packets and how they are built will help quite a lot in the practical work presented in this section. This application basically set two units to transmit packets to each other constantly. The structure of these packets is defined and shown here in a header file called 'PongMsg.h'.

```
enum
{
  AM_PINGMSG = 7,
  AM_PONGMSG = 8,
};

typedef struct PongMsg
{
  uint16_t src;
  uint8_t src_rssi;
  uint8_t src_lqi;
  uint16_t dest;
  uint8_t dest_rssi;
  uint8_t dest_lqi;
} PongMsg_t;

typedef struct PingMsg
{
  uint16_t src;
} PingMsg_t;
```

As it can be seen in this file two messages are defined which contains the node Id information, the Signal Strength and the Link Quality Indicator from source and destination.

The following figure shows the folder containing the main Java classes used in this thesis.

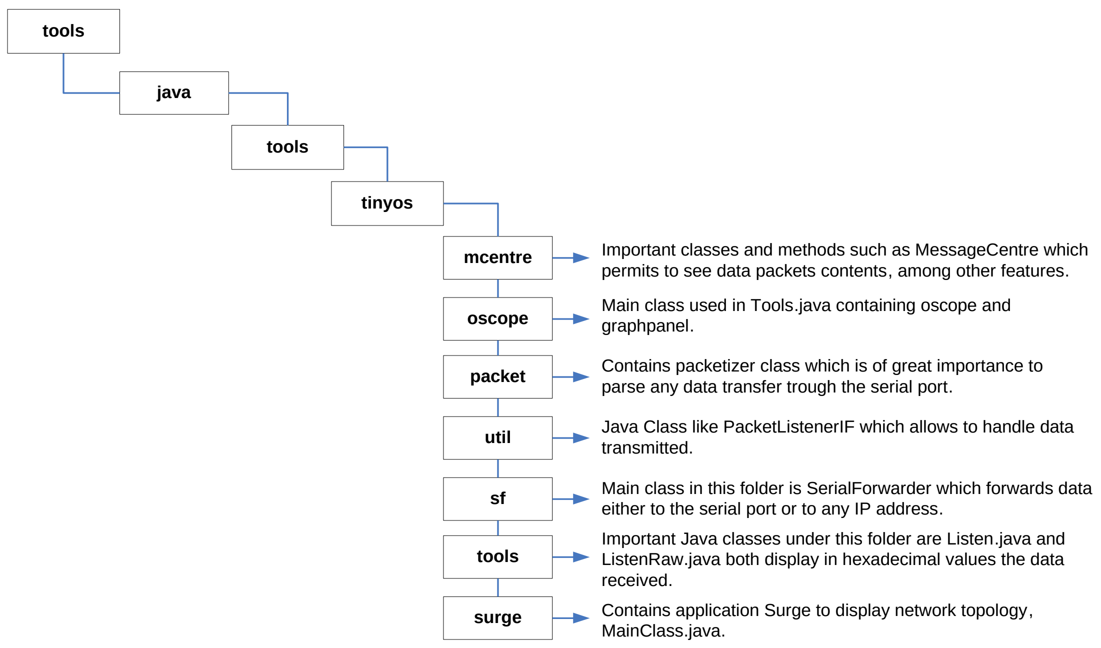

tools
java
tools
tinyos
mcentre
Important classes and methods such as MessageCentre which permits to see data packets contents, among other features.
oscope
Main class used in Tools.java containing oscope and graphpanel.
packet
Contains packetizer class which is of great importance to parse any data transfer trough the serial port.
util
Java Class like PacketListenerIF which allows to handle data transmitted.
sf
Main class in this folder is SerialForwarder which forwards data either to the serial port or to any IP address.
tools
Important Java classes under this folder are Listen.java and ListenRaw.java both display in hexadecimal values the data received.
surge
Contains application Surge to display network topology, MainClass.java.

Figure 37. *Folder structure of main Java classes and methods.*

Inside the folder called message there is a very important Java class that is called MoteIF.java. MoteIF provides an application-level Java interface for receiving messages from, and sending messages to, a mote through a serial port, TCP connection, or some other means of connectivity. Generally this is used to write Java programs that connect over a TCP or serial port to communicate with a 'TOSBase' or 'GenericBase' mote. The default way to use 'MoteIF' interface is to create an instance of this class and then register one or more 'MessageListener' objects that will be invoked when messages arrive. For example:

```
MoteIF mif = new MoteIF();
mif.registerListener(new FooMsg(), this);
```

The default 'MoteIF' constructor uses the MOTECOM environment variable to determine how the Java application connects to the mote. For example, a MOTECOM setting of "serial@COM1" connects to a base station using the serial port on COM1. It is also possible to send messages through the base station mote using,

```
MoteIF.send();
```

To make possible for the Java tool to handle the data coming through the serial port where one of the nodes is connected, two classes are defined. The one called ‘SerialPortReceivingPaint’ handles the graphical representation of the data coming through the serial port. It is shown here briefly here as full source code can be found in the documentation CD.

```
public class SerialPortReceivingPaint implements MessageListener {
        MoteIF mote;
        public SerialPortReceivingPaint() {
            mote = new MoteIF();
                  mote.registerListener(new PongMsg(), this);
```

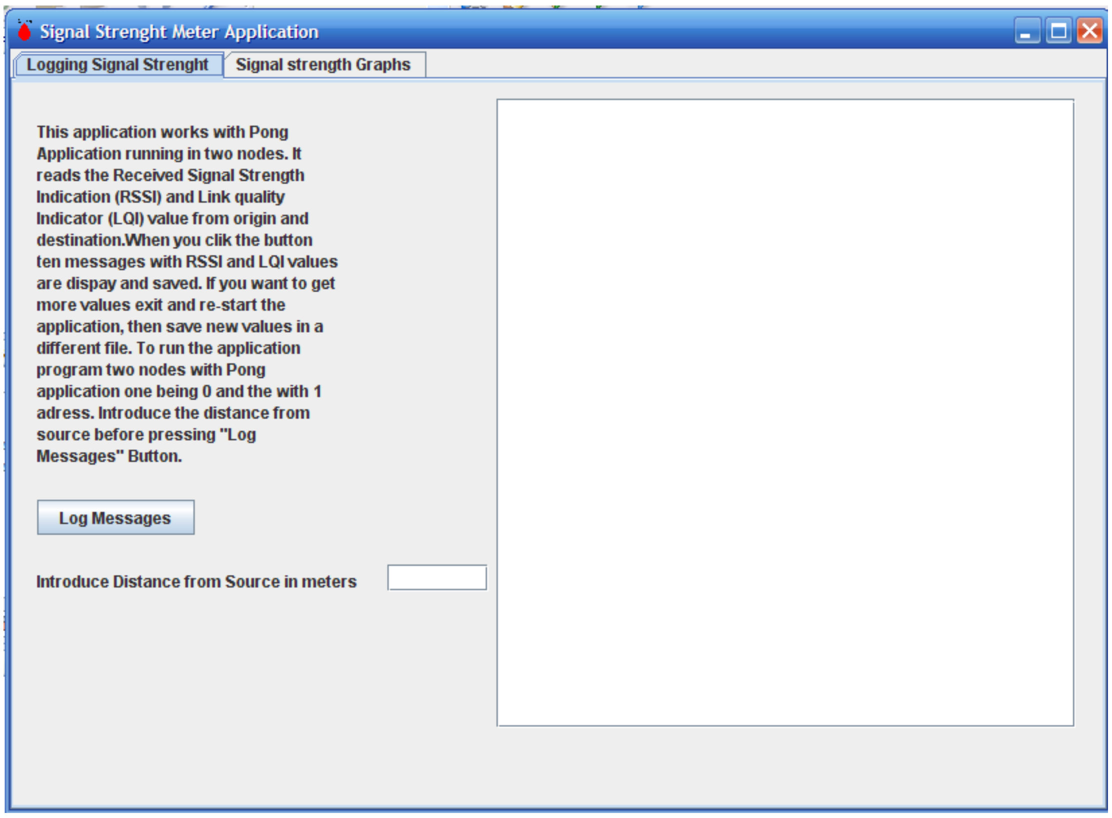
The other class called ‘SerialPortReceivingFile’ deals with the logging of the data into a text file named and saved into a chosen location in the PC by the user. Again it is shown just few lines of this class.

```
public class SerialPortReceivingFile implements MessageListener {
        boolean condition = true;
        MoteIF mote;
        FileOutputStream out; // declare a file output object
              PrintStream p; // declare a print stream object
```

The start of sending packets from source to destination is triggered by the GUI, once the user decides if to graphically display the data or to log it in a file. Methods defined in each of the classes mentioned earlier implement this functionality, here it just one of them.

```
public void sendpingmessagetopaint() {
            int i;
            PingMsg ping = new PingMsg();
            ping.set_src(0x007E);
            try {
                mote.send(MoteIF.TOS_BCAST_ADDR, ping);
            } catch (IOException ex) {
                ex.printStackTrace();
            }
```

The graphical user interface has been kept very simple in appearance as the real aim of this practical exercise was to prove how a new Java tool can be designed if needed. There are plenty of existing classes that can be used directly or derive new ones from, which gives Java a great advantage in TinyOS over any other programming language. Next figure shows how it looks the Java tool.



This application works with Pong Application running in two nodes. It reads the Received Signal Strength Indication (RSSI) and Link quality Indicator (LQI) value from origin and destination.When you clik the button ten messages with RSSI and LQI values are dispay and saved. If you want to get more values exit and re-start the application, then save new values in a different file. To run the application program two nodes with Pong application one being 0 and the with 1 adress. Introduce the distance from source before pressing "Log Messages" Button.

Introduce Distance from Source in meters

Figure 38. Signal strength application.

When user pressed Log Messages button a dialog appears giving the option of where to save the data to, this action also triggers the sending of the packets. The distance of the measurement can be introduced and it is saved along with the values, date and time of the experiment. The second tab as it has been mentioned above it displays the incoming data graphically in real time.

Every time a node receives a message automatically transmits back another. Both nodes involved in the communication are sending each other ping messages and replying to each other with a pong message. Then node acting as the base station sends the values to the computer trough the serial port. Several different scenarios are tested:

1. Base station with External Antenna of 6dBi and destination node inside the plastic box.
2. Base station with External Antenna of 6dBi and destination node outside the plastic box.
3. Base station with Microstrip Patch Antenna and destination node inside the plastic box.
4. Base station with Microstrip Patch Antenna and destination node outside the plastic box.

The destination node is tested inside a plastic box and without it, as this will be the normal enclosure of the units. The idea is to observe if the model chosen will have any effect on the transmission.

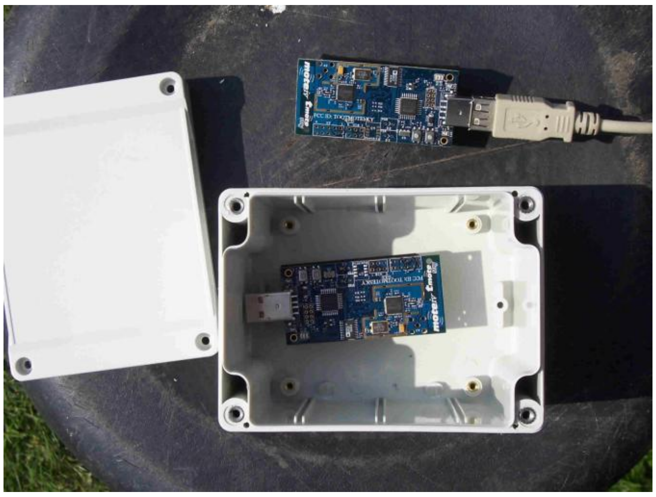


Figure 39 Hardware platforms on field.

The field is marked every 10 metres the base station is fixed and connected to the computer, the destination is moved with every measurement taken to the next position until no signal is received. Both source and destination are in a straight line clear of any obstacles. Sometimes restarting of both units is necessary to repeat measurement. The weather condition when the experiments are carried out is in a sunny windy day.

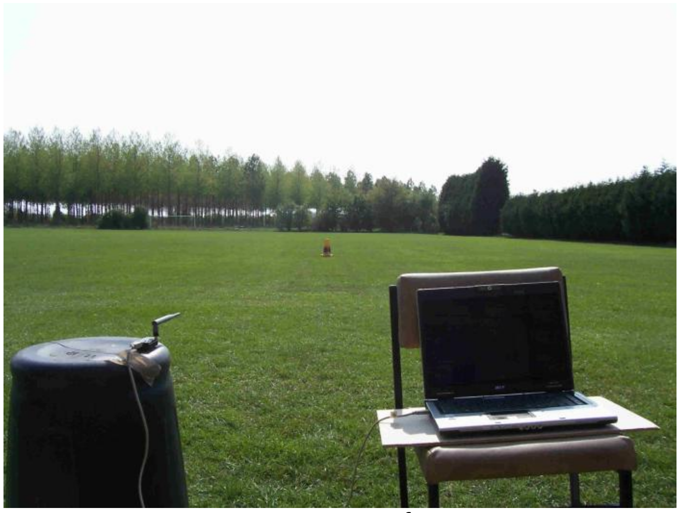


Figure 40. Picture of Experiment.

Graphs obtained reveal that the maximum range of transmission for this hardware unit is less than the one specified by manufacturer.

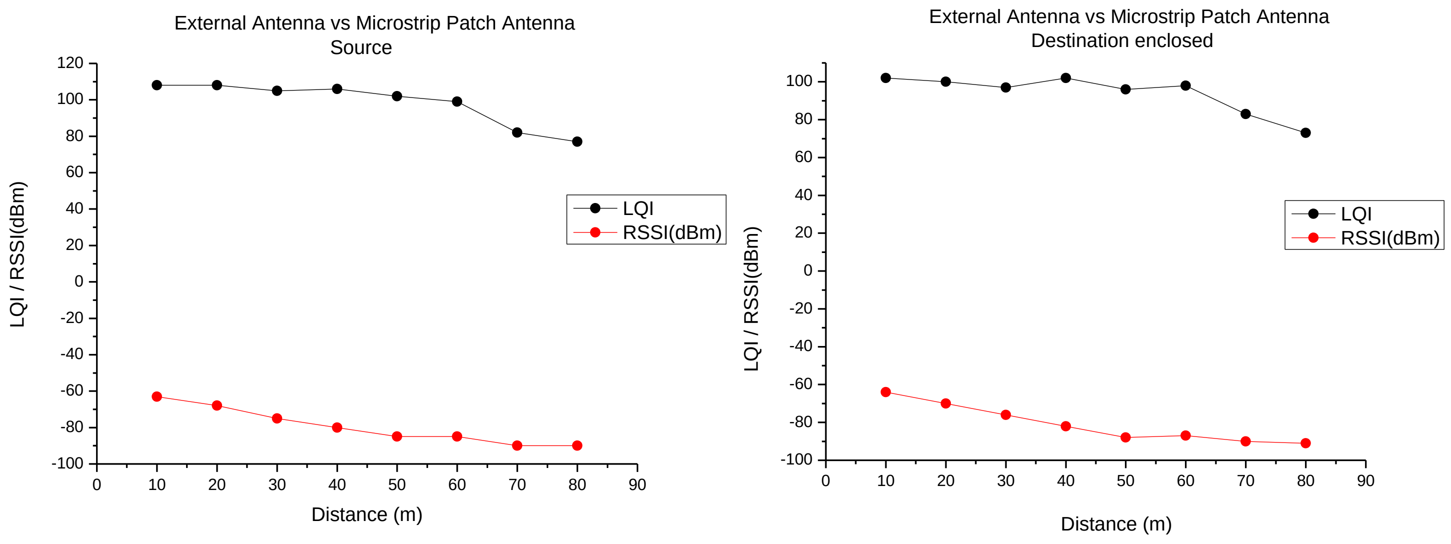


Figure 41. Graph results when Base Station is using an external antenna vs node using microstrip patch antenna and inside the enclosure.

After 80 metres signal is lost, the two parameters decrease with distance as it was expected. Again the increase at 60 metres is due to the radiation pattern of the external antenna. Performance of the microstrip patch antenna within the enclosure is very poor, although the enclosure box is made of plastic which should not be absorbing any energy.

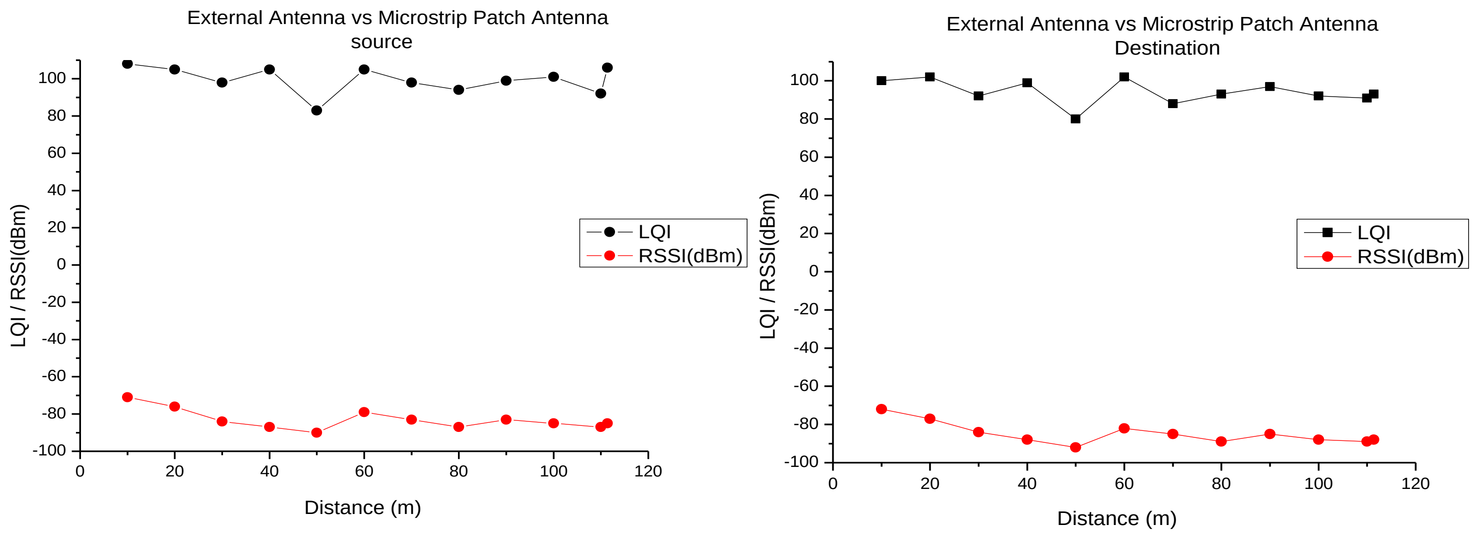


Figure 42. Graph results when Base Station is using an external antenna vs node using microstrip patch antenna and outside the enclosure.

The LQI and RSSI looks at 60 metres like they are increasing and then 10 metres after they start decreasing again that is due to the electromagnetic radiation pattern of the 6dBi external antenna. After 80 metres remains constant and then transmission is lost for distances greater than 115metres. This is the best case scenario, when it is achieved the greatest distance between nodes.

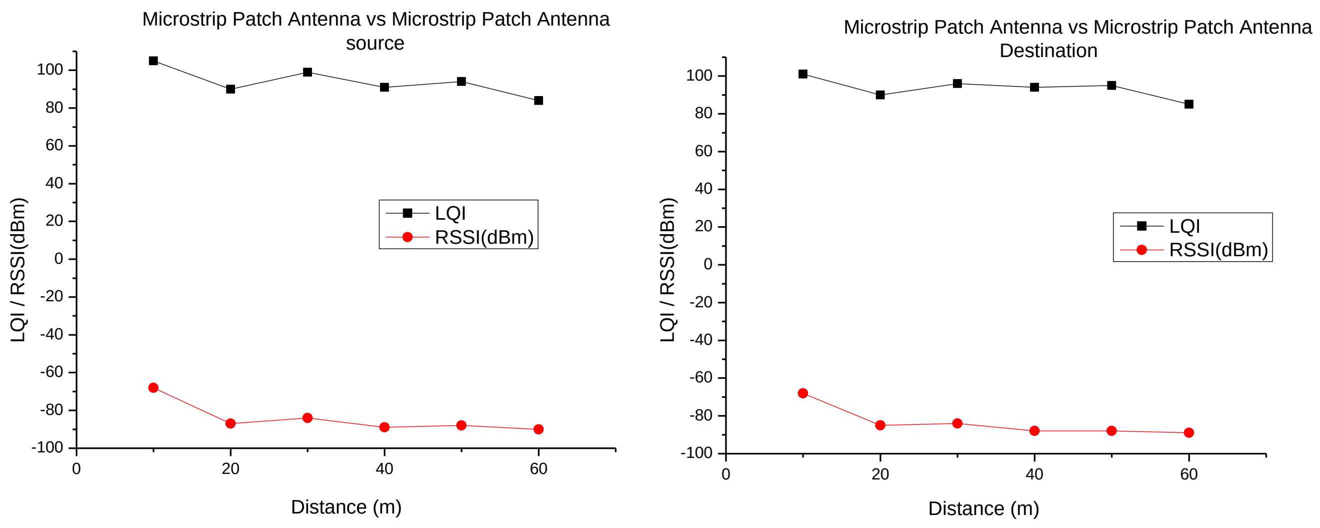


Figure 43. Graph results when Base Station is using the microstrip antenna vs node using microstrip patch antenna and inside the enclosure.

Transmission signal remains almost constant with the different distances, but performance is very poor. After 60 metres the signal is lost, increases as for example at 30 metres are due to the radiation pattern of the microstrip patch antennas.

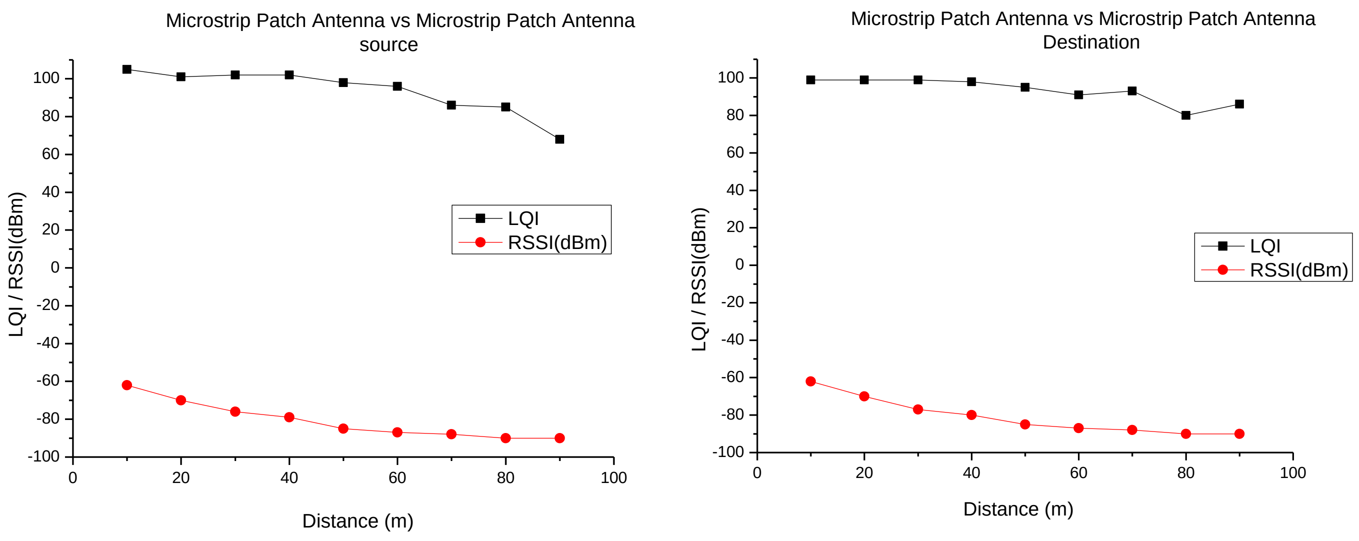


Figure 44. Graph results when Base Station is using the microstrip patch antenna vs node using microstrip patch antenna and outside the enclosure.

Signal strength decreases quite clearly versus distance and clearly the enclosure is decreasing the performance of the microstrip patch antenna. Longer distances are achieved without using it like can been seen in the previous graph.

Results are logged into text files and then using Matlab (Mathworks, USA) to plot the graphs. The inclusion of just this graph in the report is because the maximum range is achieved in this scenario. The rest of the graphs suggest that the maximum distance in where nodes of a Wireless network might be placed from each other should not be greater than 80 metres.

4 DISCUSSION

In chapter 3 has been presented the practical work and experiments carried out in order to evaluate the performance of the chosen hardware platform and the software. In this section it will be commented conclusions about some important issues surrounding this technology.

4.1 HARDWARE

From the evaluation of the hardware platform acquired for this project there are some aspects to put forward. The hardware architecture is out of discussion since a microcontroller connected to an ADC to get the sensor readings and to a transceiver to transmit them over the air is just the only alternative. The price of this platform versus competitors is also an advantage. Receiving the PCB board plain without any sensors attached is also an advantage that allows users to try custom made sensor boards and experimenting with different sensors.

It has been observed through this project using this platform that power consumption is relatively good. According to Moteiv, the company responsible of this hardware platform, when running Delta Application if the number of nodes in network is five with one gateway transmitting a message every 10 seconds the lifetime of the network might be approximately of 16.92 months. The size of the actual platform could be reduced if in the future. In future applications of this technology it is very recommended to study the environment the network is going to be deployed in order to establish if energy scavenging techniques are suitable. Energy sources could include, sun light, vibration and even temperature, there are companies manufacturing devices that could be attached to these platforms and exploit these sources of Energy. One of these companies is called 'EnOcean' (http://www.enocean.com/en/).

As shown during the practical experiments particularly the Signal strength tests, there is a significant improvement in transmission parameter LQI when connecting an external antenna to the platform. The microstrip patch antenna printed in the circuit has a very poor performance. A way of increasing the transmission range would be to set the external antenna using a cable lead as high as possible and further away from the ground.

There is a lack of information from the manufacturer about which could be the best enclosure for the board to protect it from external environmental conditions. Because of the low power used to transmit of these nodes, it will be necessary to avoid any structure made of any material that absorbs energy or introduces any losses due to reflections. Some authors believe that simple plastic boxes would be suitable, like the one purchased for this project which is made of polystyrene. During the signal strength experiments the enclosure demonstrated to decrease significantly the transmission range. Taking this into account it seems not possible to deploy an outdoors network without protecting the nodes against rain, humidity or high temperatures. It might be necessary more tests using different types of plastic boxes. The effect of the enclosure would be insignificant with the use of an external antenna connected to the platform using a cable.

Some other features of the hardware have not been tested, specifically the JTAG connections, the user button, and the flash memory of 1024Kbytes. JTAG could be very useful for this project as it allows debugging of programs, and direct access to the Texas Instrument microcontroller.

The study of the structure of TinyOS after installation has been important to establish the location of useful components and applications. Possibly the most important part of this structure is the one containing the programs acting as interfaces in the system. The new TinyOS release from Moteiv has been brought to help developers with commercial applications.

The description of some of the most important data packets transmitted by these nodes will help in the future development of applications running in Tmote hardware platform. The main advantage of that message structure is the possibility of changing it and tailor it for optimum performance of the network in future applications. In comparison with Zigbee products and software TinyOS presents many advantages.

This is an open source operative system, where the developer can modify every single aspect of the communication protocol. In the other hand using Zigbee standard makes protocol stacks inaccessible for modifications or further research. Development kits from manufacturers are quite expensive. The software running on Zigbee platforms is written mainly in embedded C language, developers normally write in the top application layer. Bottom communication software layers are provided and not admit changes. Zigbee Alliance made public the standard but did not release any software code.

Zigbee Alliance has achieved more popularity and adoption from industry because is providing standard solutions to ensure compatibility between devices and software applications manufactured by the members of the organization. At the same time these companies are designing platforms for specific applications, ready to use by the industry.

Progressing with TinyOS and NesC is strongly linked to the collaboration between research organizations. There is only one forum to post messages looking for help that is not always attended by main developers of the kernel of the Operative System. Berkley University is the main source of this forum as there was the place where everything started. It is the institution where more projects and applications come from and also the people who have created Moteiv Company. The most important company collaborating in the TinyOS research community is Intel, which is focusing mainly in health care and remote medical applications.

The architecture of NesC applications is based as described in previous sections in components, modules, configurations and interfaces. This makes very easy to understand and learn about how to produce new applications. Reusability of existing software is a powerful key of TinyOS. The problem of not having the implication from industry is the lack of development software tools. Debugging process of software code becomes a difficult task, there are several free tools. The last trial has been with promising environment called Eclipse, but the installation made previous TinyOS unavailable. With no access to the previous distribution is not possible develop anything new.

Another important issue to mention about the practical work with Java is about the development environment. It has been very difficult to start using and modifying these classes and programs. There are several software tools to do this such as NetBeans and JavaBuilder from Borland. The difficulties are in configuring the environment of these tools to make them possible to link and compile the new applications. So far the production of new tools like Tools.java has been possible after configuring carefully the path of these programs. TinyOS java tools have been compiled for the Java 2 Platform, Standard Edition v 1.4.2 (J2SE). Therefore this is very important when setting any programming environment otherwise it will not be possible to access the current classes, methods, and interfaces. Once a developer has full control of existing Java programs it is possible to start developing new ones.

4.2 EXPERIMENT RESULTS

From the network set up in the glass house some important aspects to comment from observation and evaluation of this network. Using Java tool called Trawler.java that came with last distribution of TinyOS from Moteiv; it has been observed that transmission is fine for all nodes except node2. The location of this node has proven to be inadequate as performance is very bad. This proves that is necessary a proper survey of the area the network is going to be deploy, regardless the fact that nodes might be close one to each other. Avoid any proximity if possible of metallic structures, which it might be difficult in a glasshouse environment. It is recommended then to boost up the signal with the use of an antenna with some gain. Multihop communications is very possible as it has been proved along all experiments. No experiments have been done in regards with node failures and mechanism used by these networks to cope with them.

Different network topology configurations were achieved although after a short period of time network fixed it because there are not significant variations in LQI parameter. In some graphs is observed that direct communication is established between the nodes and the Base Station as there is enough transmission quality for it. To experiment how these networks cope with node failure and re establishing routes between them and the base station it is recommended to deploy them considering greater distances from one to each other.

The graphs displaying the sensor readings when oscilloscope application is programmed show how the light detector sensors are very sensitive to light variations. The transmission rates are very fast and after five days of continuously monitoring some nodes of the network are not working and the batteries are flat. Node1 running powered with a solar cell panel has demonstrated that is possible to take advantage of this alternative energy. It works only if there is full sun light available, second day of the trial was cloudy and node1 was transmitting only when sun light was clear of clouds. It is recommended to use scheduled transmission protocols techniques, threshold values to avoid redundant information and alternative power sources. It is also recommended not to use as it was done in this experiment rechargeable batteries instead normal or alkaline batteries will have better performance and longer life of operation. The rechargeable batteries used were normal nickel-metal hydride cell **(**NiMH) type, which suffers from high self-discharge rates. They were heavily used before setting up this network, which possibly increased the discharge rate significantly, and that explains why the network lasted only for five days.

It is possible to change transmission rate introducing some timers in the software but has not been proven that this reflects into a lower power consumption, and therefore longer life expectancy for the node’s batteries. More effective seem to be the NesC components capable of turning on and off the Microcontroller, the ADC and the Chipcon transceiver. This could be done by module in folder “cygwin\opt\tinyos-1.x\tos\lib\OnOff”, although it only controls the radio.

The process of capturing data has confirmed the lack of appropriate software tools mentioned earlier in this report. It has been needed like in the case of the RSSI experiments to design a new java tool. The graphs in this report have been made using the print screen key of a pc keyboard, and using photo editor programs.

Continuous monitoring has been used in all experiments but in a real system it would make more sense to do it on demand as parameters observed normally do not change constantly. The same philosophy should be applied to the units, just transmit when a new significant value is ready. It is recommended to put in practice a scheme of how often a sensor measuring a certain parameter must be read. The progress in this side means also further development of NesC components and more use of the existing ones located under a folder called interfaces. This folder is the one containing components to control every single Hardware element of the board. Custom made components will be necessary to make possible the connection of new sensors and the development of new routing techniques and algorithms.

There is not a generic protocol of communications that can be used for all scenarios and situations. It is the application requirements which are going to determine all the design aspects of the system. Time critical applications where alarms have to be transmitted fast are of special interest for future lines of research. There are several questions to answer around this, like the difficulties of achieve low latency and fast response versus saving energy. Work in this direction is very important to determine if these networks can be a solution for such applications.

5 CONCLUSION

This thesis summarises the research carried out in the wireless sensor networks technology area. It started with an extensive revision of the field which made possible to decide which hardware platform and software tools were the most suitable to use for the practical work of this research. Two case studies were selected to apply this new concept of monitoring systems and processes. The findings and observations can serve as a guide for future research projects using this technology.

The wireless sensor networks is an emerging technology, awaking the interest from the scientific community because of the challenges it presents and the opportunities that come out from its initial state. Developers from the electronic and software sector are working in the improvement of such networks with great enthusiasm, as it presents a field where new techniques are needed. There are different challenges that this technology will need to overcome to represent a real alternative to current or future systems in monitoring, control and surveying applications.

As it has been exposed in this project the option of designing a new hardware platform proved not to be reasonable due to the variety of existing platforms and the amount of work spent already by other research groups and companies. Another issue would have been to make any design compatible with TinyOS or Zigbee. It has not been possible to cover here the research and practical work with new power schemes to obtain energy from the environment or from the scenario where the network has been set up. A Solar Cell panel connected to a node proved not to be enough to maintain a network running much longer that just using small batteries. The experiment on this line that did not take place would have been the addition of some extra circuitry to recharge batteries when there is no other activity taking place. This addition will compromise the advantage of this technology of being small in size and in number of electronic components used. As it have been mentioned today there are more devices in the market to scavenge energy from the environment, this is a line future applications might need to take into account and to exploit further.

From experimental work carried out to determine the range of the hardware platform chosen, it is recommended to deploy the network locating the nodes within a maximum distance of 80 metres. It is assumed that the nodes need an enclosure of some type, preferably non metallic; therefore this is the maximum distance to guarantee transmission of packets. From the glass house experiment has been observed that obstacles might have a big impact on transmission, therefore a survey of the environment is crucial to optimize the location of nodes.

The software side of this technology is an area where more research and experimental work is also needed. TinyOS is a very complete and powerful tool to develop and research new applications. It needs to be improved as currently does not make use of all capabilities and features of the powerful tiny microcontrollers used in the hardware platforms like the one presented along this research.

Along the practical work carried out in this thesis a maximum of five nodes have been connected to form a network when the expectations for this technology is to have a great number of units. Future researchers will have to investigate methods of handling the information gathered by the nodes of a huge network and how to make this information available to the end user of this technology.

When dealing with alarm systems as for example one to avoid oil leakages and therefore serious contamination is when this technology as it stands now is possibly not the best answer. The requirements of such a system pose a great challenge and contradict some of the main features around wireless sensor networks. Current saving power schemes where nodes of a network go to sleep mode most of the time will not meet the requirement of constantly monitoring possible leaks. At the same time when talking about networks made up by hundreds of nodes is not practical having to replace batteries very often. The challenge deserves the efforts of the research community as success could benefit many other similar operational critical systems. The original aim and description of these networks has been altered by industry and research communities as platforms and software is becoming quite complicated. Basically the hardware platforms have been over designed with the use of very powerful and complex microcontrollers. The software has suffered from this also, and it is becoming too abstract and difficult to work with for the end user. That is the reason to shy there are so many lines opened at the moment instead of having adopted a common framework. There no present indications as to whether the programming language for these demanded end users applications is going to be Java, C++, C# or any other.

Market trends for the future are promising for this technology as it is forecasted a steady increase in the adoption of these networks by the industry. The challenges that need to be attended and solved for these prediction to become truth are, power consumption, congestion of WIFI bands and impact in the 2.4 GHz band as they will coexist, and finally further reduction of manufacturing cost.

6 REFERENCES

1. F.L.Lewis (2004). Wireless Sensor Networks. Smart Environments: Protocols and Applications ed. D.J.Cook and S.K.Das, John Willey, New York. 1-18.

2. K. Römer, F. Mattern: The Design Space of Wireless Sensor Networks, IEEE Wireless Communications, Vol. 11 No. 6, pp. 54-61, December 2004.

3. Theodore S. Rappaort, “Wireless Communications: Principles and Practice”, Prentice Hall, 1996.

4. 802.15.4-2003 IEEE Standard for Information technology— Telecommunications and information exchange between systems— Local and metropolitan area networks— Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE computer Society. December 2003.

5. Al-Karaki, J.N.; Kamal, Routing techniques in wireless sensor networks: a survey. A.E. Wireless Communications, IEEE [see also IEEE Personal Communications], Vol.11, Iss.6, Dec. 2004 Pages: 6- 28.

6. J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks," Wireless Networks, Vol. 8, 2002, pp. 169-185.

7. Ye F.; Chen, A.; Liu, S.; Zhang L. A scalable solution to minimum cost forwarding in large sensor networks. Proceedings of Tenth International Conference on Computer Communications and Networks, Pages 304 -309, 2001.

8. M. Handy, M. Haase and D. Timmermann. Low-energy adaptive clustering hierarchy with deterministic clusterhead selection, Proc. of IEEE Int. Conference on Mobile and Wireless Communications Networks, Stockholm, September 2002.

9. Lindsey, S.; Raghavendra, C.S. PEGASIS: Power-efficient gathering in sensor information systems; Aerospace Conference Proceedings, 2002. IEEE Volume 3, 2002 Page(s):3-1125 - 3-1130 vol.3.

10. C. S. Raghavendra, Krishna Sivalingam, Taieb Znati, Eds., Wireless Sensor Networks, Kluwer Academic Press, April 2004

11. Mark Hempstead, Nikhil Tripathi, Patrick Mauro, Gu-Yeon Wei, and David Brooks. An Ultra Low Power System Architecture for Wireless Sensor Network Applications. Accepted for Publication, 32nd International Symposium on Computer Architecture (ISCA'05), Madison, WI, June 2005.

12. Roundy, S., Wright, P. K., and Rabaey, J., 2003. "A Study of Low Level Vibrations as a Power Source for Wireless Sensor Nodes", Computer Communications, vol. 26, no. 11, pp 1131 - 1144.

13. Edgar H. Callaway, Jr. and Edgar H. Callaway, "Wireless Sensor Networks: Architectures and Protocols," CRC Press, August 2003.

14. Soo Young Shin, Sunghyun Choi, Hong Seong Park, and Wook Hyun Kwon, "Packet Error Rate Analysis of IEEE 802.15.4 under IEEE 802.11b Interference," in Proc. Conference on Wired/Wireless Internet Communications (WWIC'2005) (Springer LNCS, vol. 3510), Xanthi, Greece, May 11-13, 2005.

15. ZigBee Specification. ZigBee Document 053474r05, Version 1.0. Sponsored by: ZigBee Alliance. June 20, 2005. http://www.zigbee.org/en/.

16. Hill, J. L. System architecture for wireless sensor networks. Doctoral Thesis. University of California, Berkeley, 2003.

17. Philip Levis, Sam Madden, David Gay, Joe Polastre, Robert Szewczyk, Alec Woo, Eric Brewer and David Culler "The Emergence of Networking Abstractions and Techniques in TinyOS." In Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004).

18. D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In Proc. of Programming Language Design and Implementation (PLDI), pages 1--11, San Diego, CA, June 2003.

19. McCauley, I.; Matthews, B.; Nugent, L.; Mather, A.; Simons, J., Wired Pigs: Ad-Hoc Wireless Sensor Networks in Studies of Animal Welfare. Embedded Networked Sensors, 2005. EmNetS-II. The Second IEEE Workshop on 30-31 May 2005 Page(s):29 - 36.

20. Victor Shnayder, Borrong Chen, Konrad Lorincz, Thaddeus R. F. FulfordJones, and Matt Welsh. Sensor Networks for Medical Care. Division of Engineering and Applied Sciences Harvard University. Technical Report TR-08-05, Division of Engineering and Applied Sciences, Harvard University, 2005. http://www.eecs.harvard.edu/~mdw/proj/codeblue/.

21. Joseph Polastre, Jonathan Hui, & co. A Unifying Link Abstraction for Wireless Sensor Networks. In Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (SenSys), November 2-4, 2005)

22. Ritchie, L; Ferguson, C; Saini, S. A novel sensor for monitoring leakage of petroleum and other liquid hydrocarbons into soil environments. Journal of Environmental Monitoring. April 2000, Vol.2, IS 2. Pages (193-196).

23. Smart Energy Profile Specification. ZigBee Document 075356r14, Sponsored by: ZigBee Alliance. May 28, 2008. http://www.zigbee.org/en/.

24. Colin Forste, Steve Methley. Wireless Sensor Networks Final Report. Document Name 0MR003, Version 2. 23 May 2008 http://www.ofcom.org.uk/research/technology/research/emer_tech/sensors/