LAPPEENRANTA UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

**MASTER'S THESIS**

# IMPLEMENTING GATEWAY MONITORING SERVICE FOR INFRASTRUCTURE SENSOR NETWORKS

The topic of Master's Thesis was approved by the council of the Department of Information Technology on 09.12.2009

Supervisors: Professor Jari Porras
D.Sc (Tech) Pekka Jäppinen

Lappeenranta, September 29, 2010

M.Mubeen Khan
Leirikatu 2 A 2
53600 Lappeenranta

Mobile: +358 468949501
mubeen.khan@lut.fi

## ABSTRACT

Lappeenranta University of Technology
Department of Information Technology
Khan, Muhammad Mubeen

**Implementing Gateway Monitoring Service for Infrastructure Sensor Networks**

Thesis for the Degree of Master of Science in Technology
2010
66 pages, 30 figures, 3 tables and 2 appendices.

Examiners: Professor Jari Porras
D.Sc. (Tech) Pekka Jäppinen

Keywords: Web Services, Wireless Sensor Networks, Sensors Monitoring, Gateway Application, Wireless Sensor Node, Mobile Client.

Wireless sensor networks and its applications have been widely researched and implemented in both commercial and non commercial areas. The usage of wireless sensor network has developed its market from military usage to daily use of human livings. Wireless sensor network applications from monitoring prospect are used in home monitoring, farm fields and habitant monitoring to buildings structural monitoring. As the usage boundaries of wireless sensor networks and its applications are emerging there are definite ongoing research, such as lifetime for wireless sensor network, security of sensor nodes and expanding the applications with modern day scenarios of applications as web services. The main focus in this thesis work is to study and implement monitoring application for infrastructure based sensor network and expand its usability as web service to facilitate mobile clients. The developed application is implemented for wireless sensor nodes information collection and monitoring purpose enabling home or office environment remote monitoring for a user.

**TIIVISTELMÄ**

Lappeenrannan Teknillinen Yliopisto
Tietotekniikan osasto
Khan, Muhammad Mubeen

**Yhdyskäytävällisen valvonta palvelun toteutus infranstruktuurisessa sensoriverkossa**

Diplomityö
2010
66 sivua, 30 kuvaa, 3 taulukkoa ja 2 liitettä.

Tarkastajat: Professori Jari Porras
TkT Pekka Jäppinen

Hakusanat: Web-palvelu, langaton sensoriverkko, valvonta-sensori, yhdyskäytävä sovellus, langaton sensorisolmu, mobiililaite.

Langattomat sensoriverkot ja niiden sovellukset ovat laajasti tutkittu aihe, joka on otettu käyttöön niin kaupallisilla kuin ei-kaupallisilla alueilla. Langattomien sensoriverkkojen käyttö on lisääntynyt ja markkina-alue on kasvanut armeijan käytöstä jokapäiväiseen käyttöön auttaen ihmisten arkea. Valvonnan näkökulmasta langattomia sensoriverkkoja ja sen sovelluksia hyödynnetään muunmuassa kodin ja asukas-seurannassa, viljelys-pelloilla sekä rakennusvalvonnassa. Samaan aikaan kun langattomien sensoriverkkojen ja sovelluksien käyttörajat ovat laajentuneet, on tutkimus lisääntynyt. Meneillään olevat tutkimukset koskevat muunmuassa sensoriverkkojen elinkaarta, sensorisolmujen tietoturvallisuutta ja sitä kuinka laajentaa sensoriverkkojen sovelluksia web-palveluiksi. Diplomityön tarkoituksena on ollut toteuttaa infrastuktuuriperusteisen sensoriverkon sovellus ja laajentaa sovelluksen käytettävyyttä web-palveluna, jolloin sovellusta voidaan hyödyntää myös mobiililaitteella. Toteutettu sovellus on tarkoitettu sensorisolmujen keräämän informaation kokoamiseen ja valvontaan sallien täten käyttäjän kodin tai toimistoympäristön valvonnan.

**ACKNOWLEDGEMENTS**

This thesis is the result of my studies in the department of Information Technology at Lappeenranta University of Technology. The thesis work was carried out in department of Communication Software Laboratory (ComLab) as partial fulfillment for the requirement of the degree of Masters of Science in Technology.

I would like to give my special thanks to LUT international services and people involved in admission services for selecting me for the Master's Degree program. It has been truly learning journey this far and it will lead me positively towards better aspect of life. I feel myself blessed to have my supportive parents and same way the teachers at university. I will always remember my mother words "Teachers are next to your parents: talk to them humble, listen to them carefully and respect them as they are working for greater good of spreading the knowledge".

My sincere gratitude I give to Jari Porras and Pekka Jäppinen who have been great teachers and supervisors being role models throughout my studies to the completion of this thesis work. Their calm support and amicable demeanor is excellent encouragement for students. I also want to thank Susanna Koponen who has been great help assisting my study plan.

My appreciation I give to my brothers Ali, Afaraz, sister Abeer for their encouragement, my uncle Ishtiyaq Khan for financial support at time of needs and all the friends whom I have met in LUT. Last, but not least Minna Kunttu, Your morale support has been important, especially being supportive all the time, without you I don't think I would have completed the degree on time.

*Mubeen Khan*

**TABLE OF CONTENTS**

**ABBREVIATIONS**

| | |
|---|---|
| AES | Advanced Encryption Standard |
| AODV | Ad hoc On-Demand Distance Vector |
| API | Application Programming Interface |
| ARM | Advanced RISC Machine |
| BT | Bluetooth |
| CLDC | Connected Limited Device Configuration |
| CORBA | Common Object Request Broker Architecture |
| DL | Description Logics |
| DSDVR | Destination-Sequenced Distance-Vector Routing |
| DSSS | Direct sequence spread spectrum |
| ECC | Elliptic Curve Cryptography |
| EEPROM | Electrically Erasable Programmable Read-Only Memory |
| GEAR | Geographic and energy aware routing |
| GPRS | General Packet radio service |
| I/O | Input/Output |
| IC | Integrated Circuit |
| IEEE | Institute of Electrical and Electronics Engineers |
| IP | Internet Protocol |
| ISM | Industrial Scientific and Medical |
| J2SE | Java 2 Platform Standard Edition |
| JVM | Java Virtual Machine |
| LEACH | Low-energy adaptive clustering hierarchy |
| LIME | Linda in a Mobile Environment |
| MAC | Medium Access Control |
| MANETs | Mobile Ad Hoc Networks |
| MIDP | Mobile Information Device Profile |
| MiLAN | Middleware Linking Applications and Networks |
| MIPS | Million Instructions Per Second |
| nesC | network embedded system C |

| | |
|---|---|
| OGC | OpenGIS Consortium |
| OIL | Ontology Interchange Language |
| OMG | Object Management Group |
| OOP | Object Oriented Programming |
| OQPSK | Offset Quadrate Phase-Shift keying |
| OS | Operating System |
| OSI | Open System Interconnection |
| PAN | Personal Area Network |
| PEGASIS | Power-Efficient Gathering in Sensor Information System |
| PHP | Hypertext Preprocessor |
| PRB | Processor Radio Board |
| QoS | Quality of Service |
| RAM | Random Access Memory |
| RDF | Resource Description Framework |
| RF | Radio Frequency |
| ROM | Read Only Memory |
| RSA | Rivest-Shamir-Adleman encryption algorithm |
| SDK | Software development kit |
| SKEW | Self key establishment protocol for wireless sensor |
| SLIM | Secured Lightweight Interactive Middleware |
| SML | Sensor Model Language |
| SOA | Service Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| SPIN | Sensor Protocol for Information via Negotiation |
| SunSPOT | Sun Small Programmable Object Technology |
| SWE | Sensor Web Enablement |
| TinyDB | Tiny Database |
| TinyOS | Tiny Operating System |
| TSP | Twisted Shielded Pair |
| UML | Unified Markup Language |
| USB | Universal Serial Bus |

| | |
|---|---|
| VM | Virtual Machine |
| WSDL | Web Service Description Language |
| WSN | Wireless Sensor Network |

## 1. INTRODUCTION

Wireless Sensor Network (WSN) is a network consisting of small, battery-powered wireless devices which have on-board processing, communication and sensing capabilities. WSN uses radio communication method and technologies for communication among sensor nodes (MOTES) for low power consumption [1]. Wireless sensor nodes are designed with concept of having small electronic device which can sense for example environment changes, compute and transmit that data to remote host.

In recent advancement with the WSN technology the size of the sensor nodes can be microscopically small for example in case of surveillance use, so that they can be hidden in surrounding environment and deployed for monitoring usage [2]. The usage of WSN and its application is widening in industrial and commercial purpose and can be seen in cases like remote healthcare, home surveillance or monitoring, industry equipment and process monitoring.

The wireless sensor node heavily relies on the battery power source for communication and co-operate data with other sensor nodes to compute and transmit data to root node. With the limitation of battery power it is unaffordable if senor node goes down especially in extensively sensitive monitoring environment. Different research works has been done to overcome this challenge by creating power aware routing protocol. Although this research area is interesting but it is beyond scope of this thesis work and mainly the work is focusing on monitoring application aspects of WSNs.

The use of WSN and its applications are increasing in general living needs. In an example case a user is requiring to know the temperature, humidity, light and position of the equipment connected with wireless sensor node at user's home or office. The challenge occurs when added for information monitoring the scenario where user has access only to mobile phone instead of desktop system. The challenge of technology concept evolves when different kind of wireless sensors using proprietary communication protocol are connected to a host computer which facilitates user sensor monitoring requirements. The challenge is resolved by using collected sensing data from different wireless sensor nodes and processed to gateway computer application, which act as intermediate between

sensor network and client by enabling web services. The gateway computer consists of basestation node and *gateway application* to integrate different sensors node data, providing *web service* to give sensor data access to user anywhere on any device. Figure 1 shows the architecture scenario of infrastructure wireless sensor network with gateway server and possible clients. In Figure 1 white color nodes are wireless sensor nodes and black nodes are base station device. The base station device is connected to application server which is running sensor information collection application and web server. The client can be either a desktop client or mobile client whose requirement is to monitor the collected sensor information from deployed wireless sensor networks.

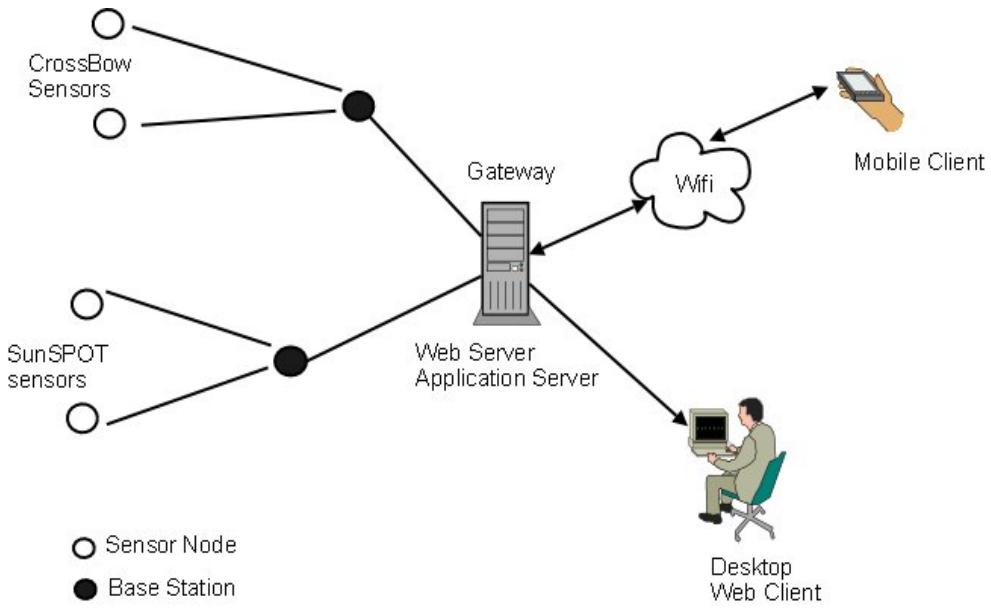


Figure 1. Network Architecture Diagram of Gateway Monitoring Application for Infrastructure Sensor Networks

### 1.1 Objective and Outline of Thesis

The main objective of this thesis is to explore the wireless sensor networks and their application usage in monitoring environment. On top of the work is the realization of web services benefits by integrating it with the standard desktop application to facilitate mobile client to view sensor node information from anywhere. The purpose of developed gateway application is that it serves as an intermediate between wireless sensor networks

and client. The mobile client will have same sensor data information as desktop client. Therefore gateway server is not only responsible for collecting sensor data but to also provide requested data to browser based and mobile based client. In addition to the WSN application development, mobile phone based application monitoring client development are taken into account. Programming language for application development for wireless sensor node is done with nesC language for *Crossbow Inc* wireless sensor nodes and Java for *SunSPOT* wireless sensor nodes. Different kind of middleware systems are reviewed and studied for the development of simple adoptable monitoring system for WSN and sensor node information collection. The final application developed is well suited for standard infrastructure WSN monitoring and it is possible to apply this system in a home or small scale environment monitoring.

The chapter 2 introduces to the wireless sensor networks, their network topologies and wireless sensor network communications with routing protocols. Chapter 3 is focusing on the wireless sensor hardware used in the project and the operating system in contrast to the programming language for wireless sensor boards. Chapter 4 discuss about the different middleware reviewed for WSN followed by Semantic Web, Web Services, Languages and approach to create monitoring service for WSN. Chapter 5 describes the technical project details about implementing the gateway monitoring for WSN and services. Finally the conclusion part summarizes the thesis and future work and possible enhancement to the work.

## 2. COMMUNICATIONS IN WIRELESS SENSOR NETWORKS

This chapter presents the WSN topologies and routing protocols following to communication methods for wireless sensor node. WSN is normally composed of number of sensor nodes scattered in physical space, which sync data to one or more base station or root node. The main function of base station in WSN is to query data from sensor nodes for example physical sensing information and process that data to required application [3].

### 2.1 Traditional Sensor Networks and Wireless Sensor Networks

Earlier in industrial and other special purpose, implementations of sensor networks were composed of using simple twisted shielded pair (TSP) implementation for every sensor to basestation device. Further these sensor network performances and processing were enhanced by Ethernet technology by implementing an industry adopted multi-drop buses to a central hub connecting to basestation [4]. This kind of infrastructure is like wired server based computing where mass collection of sensing data is aggregated to centralized database except on higher tier the connection is to Internet [4]. The cost of this kind of infrastructure to wired sensor is highly unfavorable not only in terms of cost, but also to power resources and placement of sensor devices and physical limitation of wires. Due to the physical limitation of wires with sensor nodes, wireless networking and communication module were introduced into the sensor network [5]. Researchers and companies have developed the sensor devices with on-board radio communication circuits. However better signal processing and marking distance limitation is still part of wireless sensor network research group [5].

Since the rapid development changes in technologies, the utilization of the true web based networks, for example smart home and smart networks are taking its boom in research and industry. Although WSN have similar components as traditional networks, they have to be designed and implemented differently. This is because WSN and sensor nodes have various constraints in their computation power, storage, memory, and bandwidth [4]. The major issue in WSN is often the energy resources as wireless sensor

nodes are normally deployed unattended, in a hazardous environment or a physically non-accessible location. Parameters like latency, bandwidth and accuracy are often trade offs with this major design consideration to extend the operational lifetime of the network [4]. The main difference with Traditional Sensor Networks (TSN) to WSN in terms of deployment is that the TSN were deployed in structured way either by hand or having limitation of wiring over head, whereas WSN due to its radio communication links can be deployed in unattended manner or randomly scatter on location of need [6].

Wireless sensor networks can be further differentiated from traditional ad hoc networks due to the following reasons. Numbers of sensor nodes are increased from smaller compositions to larger composition by connecting thousands of sensor nodes in the network to achieve finer granularity and increased robustness to the network [7]. Therefore sensor nodes are more densely deployed than in an ad hoc network. In general a wireless sensor node is sensitive to failure due to frequent changes of topology which are expected in a WSN. In WSN transmission method is used by broadcasting instead of point-to-point communication as in ad hoc network [7]. Because of transmission method constraints in power, processing power, bandwidth and device memory are different. In addition a wireless sensor node may not be uniquely identifiable due to a large number of sensor nodes in WSN [6].

### 2.2 WSN Network Topologies

Before deploying a wireless sensor network mainly two things are considered. These are the coverage and the connectivity of the whole network [4]. The coverage is related to application based information gathered from environment by the sensor node devices [8]. The connectivity is related to the network topology on which information routing will occur. Power consumption, energy limitation and robustness are depending on wireless sensor device selection [8]. The topologies for different kind of radio communication between wireless sensor networks are described below.

### *2.2.1 Star Network (Single Point-to-Multipoint)*

The star network topology is common network topology in networking. Basically star network topology has a single base-station which can send data or receive data from connected number of remote nodes. The remote connected nodes are only applicable to send data to other nodes if required via the base station [8]. Star network topology is easy to form and the advantage of having it in WSN environment is that the remote node power consumption can be reduced. Low level latency communication method can be used between basestation and remote sensor nodes [6]. The possible disadvantage of star network topology is that the basestation should be in radio communication range with the remote sensor nodes and failure of basestation will cut off the communication in the whole network [9]. The star network topology of WSN with single-hop central basestation is shown in Figure 2. Each remote sensor node in this topology communicates with its capacity in clear line of site to basestation. This topology formation is feasible approach and can radically simplify the design due to the networking concerns of minimal set of administration devices [8]. On other hand star network topology lacks in scalability and robustness due to its single hop transmission and routing technique. For example in larger and dense area the sensor nodes which are at distant from basestation have to compromise on poor wireless link. [9]

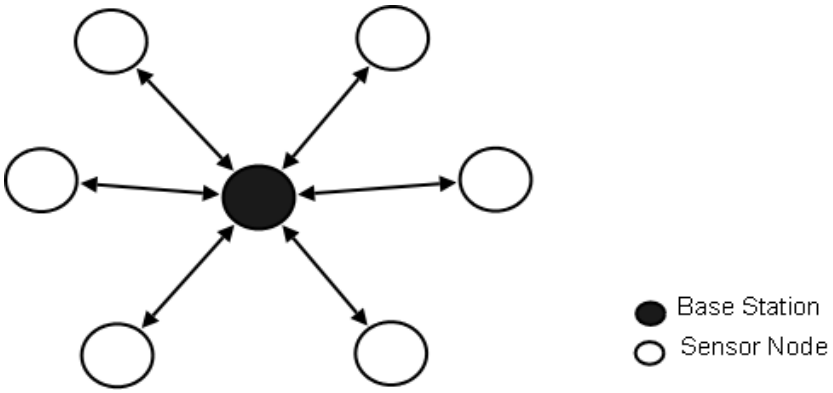


Figure 2. Star Network Topology [9]

### *2.2.2 Mesh Network*

The mesh network topology is one of the most common network topology in which devices or nodes are connected to many redundant interconnections. A mesh network lets any node in the network to transmit data to other node in the network, which is within its communication radio transmission range [3]. This technique is known as multihop communication. In multihop communication if a node needs to send a data to another node which can be out of its radio communication range, it can use another intermediate connected node to forward the data to the desired node. This message forwarding concept evolve from route technique, the internet is simple example of it as message is forward to desired node and can use alternative route in case of network or intermediate node problem. Mesh network topology is less redundant to network failure compared to star network and it is more scalable [8]. Figure 3 shows mesh network topology with the concept if an individual node 2 links fails with node 1. A node 1 can still communicate to 2 via node 3 which is in its communication range; in turn node 3 can forward the message to the desired node 2 or base station. In result the scalability of the network in mesh network is not compromise to limitation of range except between nodes, however the whole network is extendable by adding more nodes and creating multihop communication system between them [9].

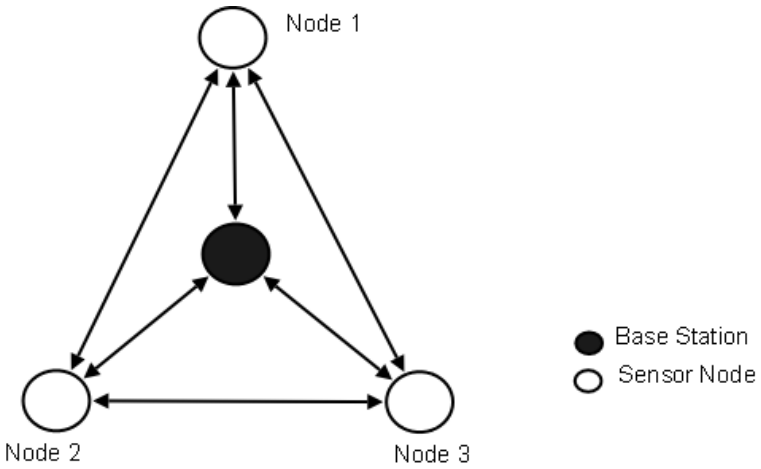


Figure 3. Mesh Network Topology [9]

The disadvantage in mesh network occurs due to power consumption of the nodes. Whereas nodes in star network topology tends to be in sleep mode after sending data to basestation, in mesh network topology the node has to be active in case of forwarding data to other nodes hence decreasing the life time of sensor node battery [8]. Also the communication from one node to another node and to desired destination can increase if the message has to pass from certain nodes, which will increase the message delivery time. Therefore mesh network is considerable choice when compromising of limited power resource and message delivery timing. [9]

### *2.2.3 Hybrid Star – Mesh Network*

The hybrid star network is a network between star and mesh network providing more robust and versatile communication network. The advent feature of hybrid network is to keep power consumption of the nodes to minimum [7]. The network topology formation is maintained in that manner that node with low power are not enabled to be in state to forward messages. This results less power consumption for overall network, but still keeping the nodes with the capability of multihop communication by forwarding the messages from low power nodes to other network nodes. [8]

Usually the nodes configured with the multihop radio communication capability have higher power consumption therefore they are connected with external power source. The hybrid network topology is usually implemented by mesh networking standard known as Zigbee [9]. Figure 4 shows the hybrid star network diagram where scalability of network is increased by having more than one basestation as compared to star or mesh topology which relies on single-basestation for the whole network.

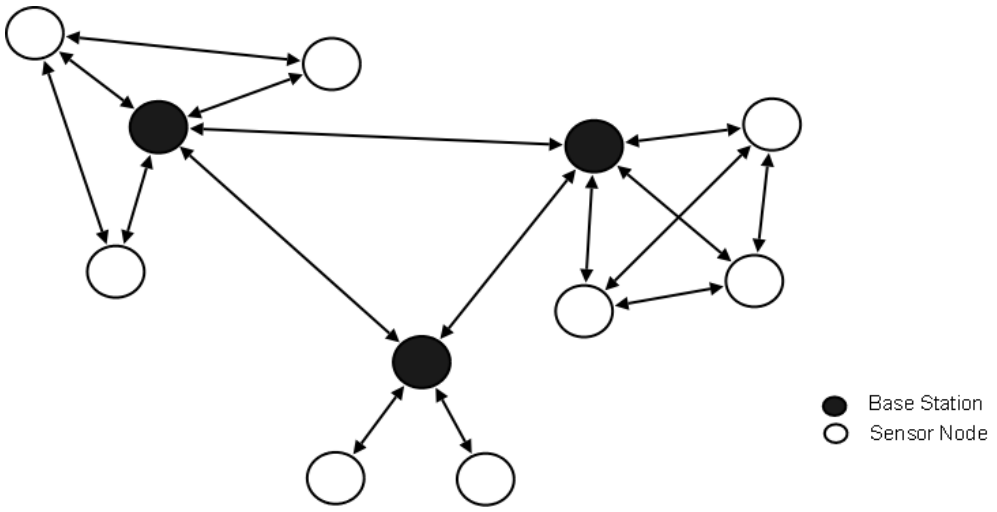


Figure 4. Hybrid Mesh Network Topology [9]

### 2.3 Analyses of WSN Routing Protocols

This section focuses on earlier proposed and researched routing protocol for wireless sensor network. The typical WSN network formations are flat network and hierarchy network [2]. A flat network is more like a star topology network where root node is the basestation device which is responsible for data gathering and every wireless sensor node in network is engage in the same role that is sensing data and sending information back to the basestation [3]. Hierarchy network have same network formation as star network but the difference is that the sensor nodes in the network is implemented on multihop radio data transmission [10]. This means that every sensor node can transfer data to another node in order to forward the data to basestation, which in result of using different routing protocol.

Routing protocol for WSNs are classified in terms of multipath-based, query-based, negotiation based and QoS based depending on flat, hierarchical and location based formation of wireless sensor network structure. [4] In flat network all nodes act as the same role, therefore any simple protocol or routing technique which is adequate in single hop communication is well suited. Hierarchical based network and its protocols aim at different routing techniques, for example clustering the nodes in which cluster heads can

reduce the overhead of extra data to save power consumption in WSN [10]. In contrast other routing technique as location based protocols relies on information data taken from position of specific regions rather then whole network [11].

WSN is closest to Mobile Ad Hoc Networks (MANETs) and therefore in most cases of wireless sensor network the topology is not fixed. In most cases star or mesh topology is commonly deployed, as wireless sensor node uses broadcast method rather then point-to-point communication as in ad hoc networks. [6]

*The Data Centric Protocol* [11] works in condition where large numbers of wireless sensor nodes are deployed and then assigned global single identifiers to each node, which can result in immense time taking task. The issue arising is that without unique identifier it is difficult to query data from wireless sensor node. In addition while transmitting the data from every node to redundant link it is in-efficient for energy consumption for WSN [2]. Therefore data-centric routing technique is considerable in those network scenarios where data is send from sink node to certain node in region. The data is requested in queries with name attribute to specific property of sensor node data [3]. *Sensor Protocol for Information via Negotiation (SPIN)* [12] is the data-centric protocol developed to eliminate redundant data and process less energy from wireless sensor network. Unlike SPIN, earlier protocols in WSN *Gossiping* and *Flooding* [11] use more energy resource by sending redundant data to whole network. The approach of this problem is resolved in SPIN by enabling data negotiation and resource aware and adaptive algorithm. Data on sensor nodes running SPIN protocol are assigned as meta-data which perform meta-data exchange negotiation between sensor nodes before transmitting, assuring this way that no similar data exists in wireless sensor nodes [12]. SPIN protocol deals with energy consumption by checking and adapting the remaining energy left in wireless sensor node.

*Low-energy adaptive clustering hierarchy* (LEACH) [13] is a cluster-based protocol that utilizes minimum energy dissipation in WSN by randomly selecting sensor nodes as cluster-heads by using hierarchy routing algorithm. The approach is apprehended by enabling clusters of wireless sensor nodes based on there signal strength and routing data to sink with local cluster heads [13], hence reducing the transmission energy by

transmitting only from cluster head nodes instead of all nodes in the wireless sensor network.

*Power- Efficient Gathering in Sensor Information System* (PEGASIS) [14] is a hierarchy based protocol. PEGASIS is slightly modified version of LEACH instead of forming multiple cluster head between sensor nodes in a network, it forms chains in WSN. The basic idea of this protocol is to maximize the network lifetime by allowing wireless sensor nodes to communicate absolutely with their closest neighbors forming a chain [14]. Therefore each sensor node in WSN can transmit and receive from neighbor sensor node. One sensor node from the formed chain is selected to communicate with the basestation, making it as turn based strategy to communicate with the basestation [15]. Figure 5 shows the aggregated data transmitting from node c0 to c4, where node c2 is selected node to communicate only with basestation in PEGASIS.

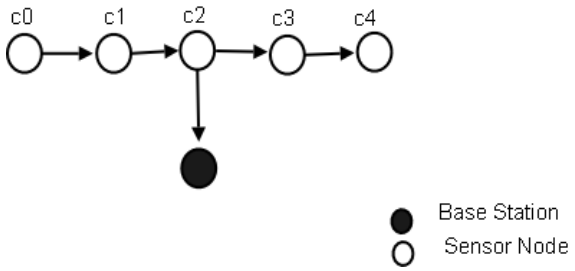


Figure 5. Chaining in PEGASIS [14]

*Geographic and energy aware routing* (GEAR) [16] is a location based protocol. Since there is no IP-address based identification for wireless sensor node, routing data based on location is quite near to energy efficient manner. Figure 6 shows the recursive geographic data forwarding in GEAR. The approach of forwarding data to wireless sensor nodes in GEAR works in two steps [16]. The first one include forwarding the data to target region shown as grey colored box in Figure 6, data forwarding is done by using geographic and energy aware neighbor selection based on heuristic routes . The next step is when the data arrives at target region it is distributed by recursive geographic forwarding algorithm. Every wireless sensor node in GEAR keeps learning record of destination and neighbor [17].

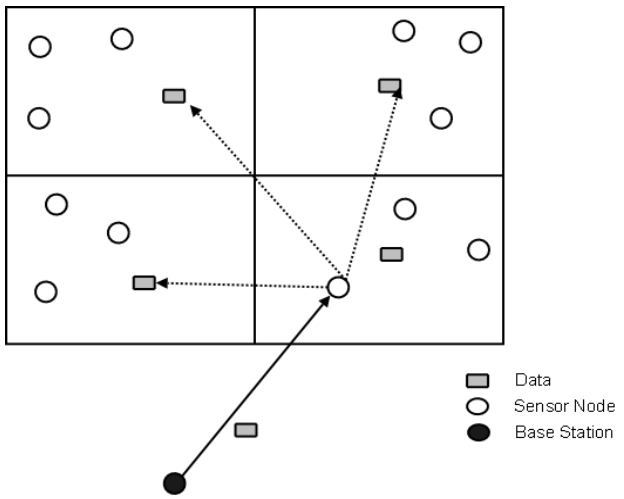


Figure 6. Recursive geographic forwarding in GEAR [16]

Table 1 shows the comparison of studied routing protocols for WSN that are SPIN, LEACH, Gossiping, PEGASIS and GEAR. Studied observation of these routing protocols is that they are appropriate with WSN performance and provide suitable results. These protocols have been mainly implemented and tested under network simulation environment. However in practical environment the wireless sensor manufacturing companies often tend to adopt different routing protocol and communication standards like Crossbow technology wireless sensor device uses *XMesh* routing protocol discussed in chapter 3.5 and IEEE 802.15.4 communication standard in case of SunSPOT. *XMesh* is a multihop routing protocol technique and are outcome of research by TinyOS community by characterizing different ad-hoc, multi-hop protocol and performance issues on Crossbow mote platform [42]. The XMesh protocol stack forms dynamically mesh network [42] between nodes. The key advent feature with XMesh is that it uses ad hoc routing methods like minimum transmission technology to reduce number of radio messages in network extending the lifetime for overall WSN [42].

Table 1. WSN Routing Protocol comparison

| | **Flooding** | **SPIN** | **LEACH** | **PEGASIS** | **GEAR** | **XMesh** |
|---|---|---|---|---|---|---|
| **Scalability** | Limited | Limited | Good | Good | No | Good |
| **Lifetime** | Short | Long | Long | Long | Long | Long |
| **Meta-Data** | No | Yes | No | No | No | Yes |
| **Data Diffusion** | No | No | Yes | Yes | No | Yes |
| **Location Awareness** | No | No | No | No | Yes | Yes |
| **Power Required** | High | Limited | High | High | Limited | Limited |
| **Classifi-cation** | Flat | Data-centric | Hierarchical | Hierarchical | Location based | Hierarchy & Location |
| **Optimal Route** | No | No | No | No | No | Yes |
| **Multi-Hop** | Yes | Yes | No | No | Yes | Yes |

The selected properties in the Table 1 for comparison between the studied WSN routing protocol can be described as; *Scalability* refer to extending the network formation between nodes and basestation, *Lifetime* refer to power consumption in WSN higher power consumption result in short lifetime of WSN. *Metadata* provides certain element resource of sensor information for example instead of broadcasting whole data, sensor node can exchange metadata between another sensor nodes. In result this will consume less energy for transmitting and receiving data on sensor nodes. *Data diffusion* is used to track route dynamically and compute data based on sensor energy in order to sink data to root node. *Location awareness* provides location of sensor node and region, only *GEAR* and *XMesh* protocol gives this facility. *Classification* is referring to network and routing formation of wireless sensor nodes. *Optimal route* technique selects the best route to destination only *XMesh* protocol follows this method. *Multi-hop* communication is used to transfer data from sensor node to another sensor node or to base station.

## 2.4 Zigbee and IEEE 802.15.4 in Wireless Sensor Networks

The radio communication for wireless sensor networks is defined in physical layer according with the Open System Interconnection (OSI) reference model [20]. The radio layer for WSN consists of operating frequency, modulation methods and interface radio scheme to sensor node radio hardware. Integrated Circuit (IC) manufacturing companies like Atmel, MicroChip and Chipcon are developing its own standard low power proprietary radio scheme for radio layer in WSN [9]. Most of the wireless sensor devices are designed with concept of integrating them with other networks and therefore a standard communication choice of IEEE 802.15.4 is used in most cases. However in some special cases, sensor devices are installed with Bluetooth (IEEE802.15.1 and .2) and external GPRS communication boards [9].

*Bluetooth* (BT) was developed by Ericsson in 1994 as an open wireless standard of exchanging data by using short distance radio link by creating personal area network (PAN) between communicating nodes [18]. The original implementation was made to transfer data between computers to peripheral devices. The network topology for BT is star network topology refer as Piconet with Master-Salve concept, the master device can communicate with seven remote nodes as a single basestation [18]. The operation radio frequency used by BT is 2.4GHz which is industrial scientific and medical use band(ISM). The frequency range for ISM band is from 2400 MHz to 2483.5 Mhz [18]. Although there is some research work and companies have implemented Bluetooth communication with sensor network, the reasons like over complex MAC layer, limited number of communicating nodes, time synchronization with network and more power consumption in returning from sleep mode makes BT protocol less attractive choice for wireless sensor applications.

The *IEEE 802.15.4* standard was particularly designed for having the requirements in mind of wireless sensing applications. The main emphasize was to create low-cost and low-speed communication between different devices [19]. The main features of IEEE802.15.4 standard are that it is flexible for multiples data and transmission frequency, with supporting topologies like mesh and star. Additionally it has features like

security with AES-128 for encryption while transmitting data with link quality indication and using direct sequence spread spectrum (DSSS) for communication [4]. The hardware for this standard is designed in this manner that it is able to be in sleep mode in terms of radio communication when not required to do any instruction sets, making it less power requirement standard and when nodes wake up from sleep mode can synchronize to the network in minimum time [5]. The specification allows for system low power supply to periodically turn off the radio. The frequencies ranges are 868 MHz, 902-928 MHz up to 2.48-2.5 GHz with supportive data rate of 20 Kbps on lower frequencies and 250 Kbps on higher frequencies [19].

The *ZigBee* standard is expansion to IEEE 802.15.4 developed by ZigBee Alliance companies to enhance network, security, cost-effective, low-power, wirelessly networked devices monitoring and controlling on an open global standard [20]. Figure 7 presents the IEEE 802.14.5 and ZigBee stack, in which ZigBee alliance specifies the application framework and security layer, build on top of physical and Medium Access Control (MAC) layer by IEEE standards. The ZigBee network specification supports star network and hybrid star mesh networks. Later IEEE 802.15.4 standard was named commercially ZigBee after forming alliance between IEEE 802.15.4 task group and Zigbee Alliance [20].

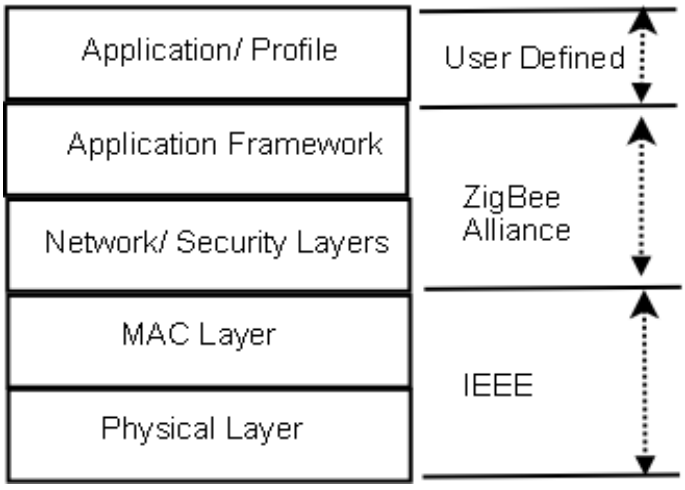


Figure 7. IEEE 802.14.5 and ZigBee stack [20]

The IEEE802.15.4 standard specifies appropriate communication architecture for wireless sensor network, although it lacks in specification for sensor interface. IEEE 1451.5 wireless sensor working group is another standardize to the specification for sensor interface on pervious IEEE1415 smart sensor working group standard [21].

### 2.5 Applications and Security Aspects of WSN

The applications of WSN are designed to serve and facilitate people different needs of daily routines. Environmental monitoring is one of the most popular choices in sensor networks, such as for monitoring water level, measuring soil quality, fire detection and flood warnings. Other well known applications with sensor networks are ‘Great Duck’ a bird observation on Great Duck Island [22], Glacier Detection [23], Disaster Operations and Monitoring [24], Medical and Monitoring [25] and Military Surveillance [26]. Since there is much more applications been developed with WSN, the security issues are increasing. The possibilities of security threats in application and wireless sensor networks like eavesdropping, forgery of sensor data, denial of service attacks or physical tampering with sensor nodes are vital issues. The easiest solution is to analyze the traffic and check the behavior of WSN on regular basis. Other possibilities are cryptographic algorithm like HIGHT [27] designed to run on 8-bit computing devices keeping the resource consumption to limited in WSNs. Hybrid Adaptive Security Framework [28] provides security suites on each packet transmission in wireless sensor network. Protocol like SKEW [29] works providing security key to wireless sensor network with focusing on less storage and computational overheads. Architecture like SLIM [30] shields the difference in sensor application layer by having the middleware on mobile as well as on wireless sensor nodes.

# 3. HARDWARE & SOFTWARE CONSTRAINTS IN SENSOR NETWORK

In this section, the various components of a wireless sensor node in the wireless sensor network are presented. In addition the details on the actual hardware used in the project work are discussed. Sensor devices are basically made up of a sensor board and a mote. Sensor board are integrated circuit designed to sense event changes, mote is main hardware which is composed of a processor, memory, radio transceiver and power supply. These components will be briefly discussed in the following paragraphs.

## 3.1 Component of Sensor Node

The basic building block hardware architecture for a sensor node is presented in Figure 8 where sensor is referred to the actual sensor circuit, which can have the capabilities of sensing light, temperature, accelerometer-degree, motion detection and others based on its hardware design. Power supply in common case is given through batteries or in advance cases can be provided via solar cells. Memory and processor are part of sensor node which gives capability to process information and run the desired application and operating system on the device [31]. The communication device is referring to the radio communication board by which sensor can communicate to host basestation device. In some of the experiments and works even BT and *Wireless Local Area Network* (WLAN) communication board is installed with sensor node [31].

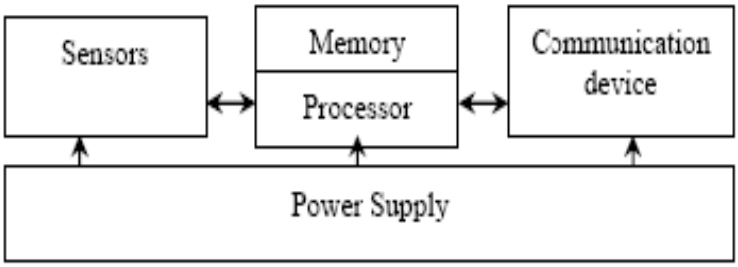


Figure 8. Block Diagram of Sensor Node Components [31]

The processor collects data from sensors and processes the data for further actions. It also gives capability to sensor device to decide when and where to send data from other sensor nodes, and decide on the connected actuator's alignments and actions. Other aspects of processor are to execute the programs, setting up communications protocols and signal processing to application and programs [22]. Normally a random access memory (RAM) and flash memory is used in sensor mote. Short term data like sensor reading and data packets from other motes are stored through RAM. Even though it is fast but disadvantage is lost of data in power interrupts. Flash memory tends to store program code and for data retained after power interrupts. The disadvantage is that it uses high energy and sometimes slowdown the access time to the mote [22]. The idea of deploying WSN is to be deployed it in unattended way, for example hazardous environment monitoring area physically beyond human reach. Therefore power supply on regular basis to sensor node is practically difficult to apply where it leads to solutions like using the device on short intervals. Other way to increase the overall lifetime of WSN is by providing external power supply like vibration energy, solar cells and temperature gradient [31]. In order to exchange data among sensor nodes or to communicate with basestation devices, radio frequency (RF) methods are applied in motes for wireless sensor networking. The advantages of RF method are that no line of sight needed and long distance operational range is achieved with high data transmission rate. The frequency ranges from 433 MHz to 2.4 GHz are commonly used in wireless sensor networks. The radio boards are built in for bidirectional but in half duplex mode, where multiple channels are available for every band and management software are used to control the band [31]. The sensors are categorized as active sensors and passive sensors. Passive sensor works on methodology by measuring changes in environment without probing energy into environment [22]. Examples of it are light, humid and vibration detection sensors. Active sensor instead provokes self generated energy to measure or find changes in respective usage environment [22], such as a seismic sensor system which measure earth quake or radar sensor system which generate energy into environment to detect changes. The sensors which are used in the project work are passive sensors.

### 3.2. Mote-Micaz and Gateway MIB-520

Micaz is mote developed by Crossbow Technology, presented in Figure 9. Micaz is compliant with IEEE 802.15.4 standard making it popular choice in research and development in wireless sensor networks. The microprocessor in Micaz is ATmega128L chip which operates at 8MHz being capable of a maximum throughput of 8 million instructions per second (MIPS), using AES-128 security method for encrypted data transmission [32]. In addition for radio communication Chipcon CC2420 is embed on Micaz. *Chipcon CC2420* implements the physical layer as defined by the IEEE 802.15.4 standard for transmitting data in standard specified 2.4 GHz radio frequency range a compatible ISM band for industrial, scientific and medical (ISM) use [33]. Chipcon radio transceivers are able to transmit up to a 250 kbps data rate. The flash memory of 128kB is reserved for as program memory with 4kB SRAM for variables and data. Micaz also implements Offset Quadrate Phase-Shift keying (OQPSK) modulation encoding, with direct sequence spread spectrum (DSSS) which gives resistant to RF interference and data security. Technical specification of Micaz brief that data can be transmitted up to 135 meters with line of sight on half-wave dipole antenna [32].

Figure 9. Actual MICAz Hardware [32]

Figure 10 shows MIB520 which acts as a gateway and also used for configuration and programming applications into MTS400 sensor motes. The MIB520 programming board is called gateway because it also serves as the basestation device to transmit and receive data to terminal device or host machine from motes. The MIB520 is connected via USB

to computer for communication and device interface with motes. The USB connection to host terminal also eliminates the need for power source for the gateway. The Micax-series connector is dedicated for mote programming and also for communication over USB to motes. To work as a basestation a mote is connected to micax-series connector and programmed to act as basestation. The on-board processor on MIB520 is that which programs MICA Processor Radio Boards (PRB) [32].

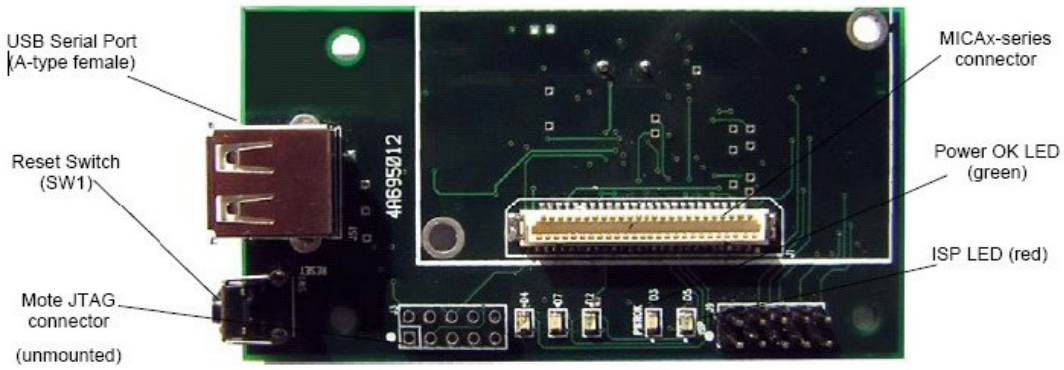


Figure 10. MIB520 USB Gateway [33]

### 3.3 SunSPOT

*Sun Small Programmable Object Technology* (SunSPOT) is developed by Sun Microsystems Laboratories (SunLabs) [34]. The basic Sun SPOT unit includes a basestation device and two sensor devices called emote. The platform includes an ARM-7 with 256Kb of RAM with 2Mb flash and 802.15.4 radio. Sensor board are loaded with 3D accelerometer, temperature, light sensor, 8 color LEDS and digital input / output pins for external device connections [34]. The point of having Sun SPOT basestation software is that it allows applications to run on the terminal host machine and to interact with applications running on end target SunSPOT sensor. Figure 11 shows the block layout of physical arrangement of SunSPOT devices connected to host via basestation to target device.


Figure 11. Layout of SunSPOT with base station connected to Host [34]

SunSPOT uses a 32 bit ARM-7 CPU with an 11 channel 2.4GHz radio. Sun Labs have developed several security technologies for wireless sensor and transducer such as public-key cryptography which is essential for boot strapping secure communication among nodes. Other security implementations on SunSPOT are Rivest-Shamir-Adleman encryption algorithm (RSA) for more optimized performance and Elliptic Curve Cryptography (ECC) for having efficiency in resources, as an alternative to RSA [34].

The host application is implemented with Java 2 Platform Standard Edition (J2SE) and target application runs in Squawk (Java Virtual Machine) program which simplifies the development of wireless sensor applications [35]. Development environment like Netbeans and Eclipse simplifies the task for developer to build wireless application using the sensor board for I/O, over radio communication of IEEE 802.15.4. The host terminal machine can be any *Windows* or *Linux* supported platform and operating system. SunSPOT SDK documentation defines that basestation can be run in either dedicated or shared mode [35]. The main difference with both modes is that dedicated mode runs in same Java Virtual Machine (JVM) as host application and only that application can use it, so therefore the host uses the same address as base station. Instead of single JVM in shared mode two java virtual machines are launched. In shared mode one JVM manages the basestation and another one runs the host application. In shared mode model the application running on host have its own address generated from system different from the base station device and more than one host application can interact and use base station concurrently [35]. The host application uses multiple processes to communicate by using the standard defined radio communication stack [34]. Possible disadvantage of having shared mode is lack of run-time management of basestation like controlling PAN ID, radio channel and output power cannot be implemented [35]. The default mode of

SunSPOT is dedicated mode and can be changed by implementing configuration changes in .sunspot.properties file in root directory [35].

### 3.4 Software Constraints

The operating system of WSN differs from traditional operating system which are more multi-threading and multi-process systems. Figure 12 shows the architecture layout for WSN where operating systems reside between the actual sensor hardware connecting it to the middleware and application. The wireless sensor nodes use less complex operating systems and event-driven programming models because of its design constraint and limited resources [37]. Therefore the operating systems of wireless sensor node are designed with even-driven technology. Also it is noticeable that, wireless sensor nodes have similar hardware to embedded devices. Therefore it is possible to use embedded operating systems such as eCos, uC/OS for sensor networks [36].

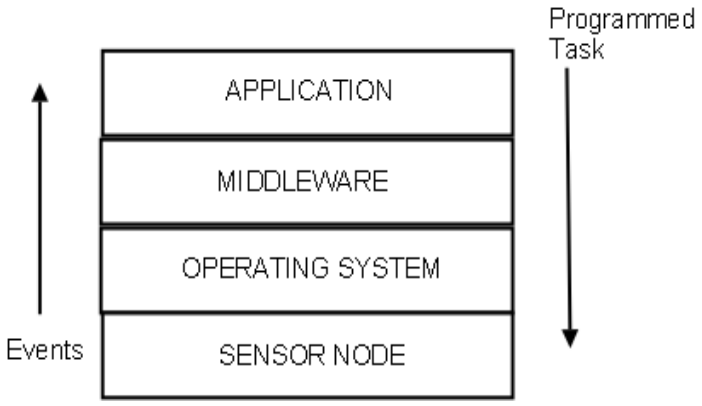


Figure 12. Architecture Layout for Middleware and Operating System [37]

Operating system is seen as software platform on which other application and program can run and interact with the hardware. In WSN an operating system hides the low level details of the sensor node by giving a virtual access to the device [36]. Operating system tasks of low-level service are processor management, memory management, device management, scheduling policies, multi-threading and multitasking [37]. The other features of an operating system in WSN are dynamic loading, unloading of modules and given application programming interface (API) for accessing the sensor hardware [37]. The key features an operating system must provide in WSN are power management,

memory management and bandwidth [38]. Further is discussed about TinyOS which is an event driven operating system used on Crossbow Micaz Motes.

### 3.5 TinyOS and nesC

*TinyOS* is an open-source operating system designed for wireless embedded sensor networks originally by the University of California Berkeley, featuring component-based architecture and enabling rapid development [39]. Figure 13 shows the TinyOS software architecture layout. The block referred as A*pplication* is component model in TinyOS which reacts on events and programmers can supply their commands, on top of it is the block referred as *Main* scheduler model which handles the constrained of given task and events [39]. The other blocks are more towards the actual sensor hardware calibration and inter-communication by using the properties for example sensing, actuating and handling radio communication defined from the component model. The wide popularity of TinyOS for WSN application is because of small memory footprint essentials. For low power devices TinyOS is perfect fit because of its event driven and object oriented operating system approach. The component library of TinyOS includes network protocols, sensor drivers, distributed services and data acquisition tools [39].

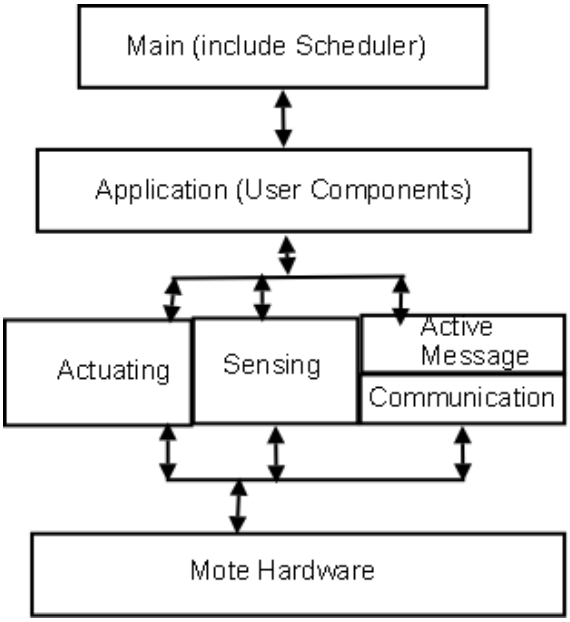


Figure 13. Simplified TinyOS Architecture Diagram [39]

The applications in TinyOS are written in *necC* (network embedded system C) language which is a small extension to C Language with consideration of power and resource limitation for wireless sensor networks [44]. TinyOS can support the microprocessors which can be as small as 8-bit architecture with 2KB RAM to more as 32-bit with 32 MB RAM [39]. The well defined sets of APIs reduce the application development from variety of system component to developer. The API also gives access to computing features of sensor nodes allowing developers to design more intelligent and specific goal oriented application to network and needs [40]. For example a node can process sensor data and undo unnecessary message before hand transmission to optimize network performance and power life time. TinyOS also supports the execution of multiple threads and provides a variety of additional extensions like the database *TinyDB* [41] which is for cooperative data acquisition.

*Xmesh* is a mesh networking protocol developed by CrossBow Inc, for developer access with wide sets of flexible networking features [42]. Figure 14 shows the relationship layout of TinyOS and XMesh networking protocol developed by Crossbow Technology. TinyOS is an open source operating system and therefore any of the OSI layer can be modified in TinyOS depending on the requirements of application. Protocol stack of

XMesh is an open-architecture which is flexible and powerful for embedded wireless networking and sensor nodes. The stack can be controlled from varied of software libraries in XMesh by using TinyOS. The network layer and data link layer block as shown in Figure 14 refers where XMesh is used to control time synchronization, sleep modes, low-power listening and node-node or basestation-node routing on sensor nodes. The rich control platform built of XMesh supports number of applications in TinyOS can extendedly give access to developers to write applications for real world. XMesh merge performance and interoperability with the support of IEEE 802.15.4 protocol in physical and MAC Layer [42].

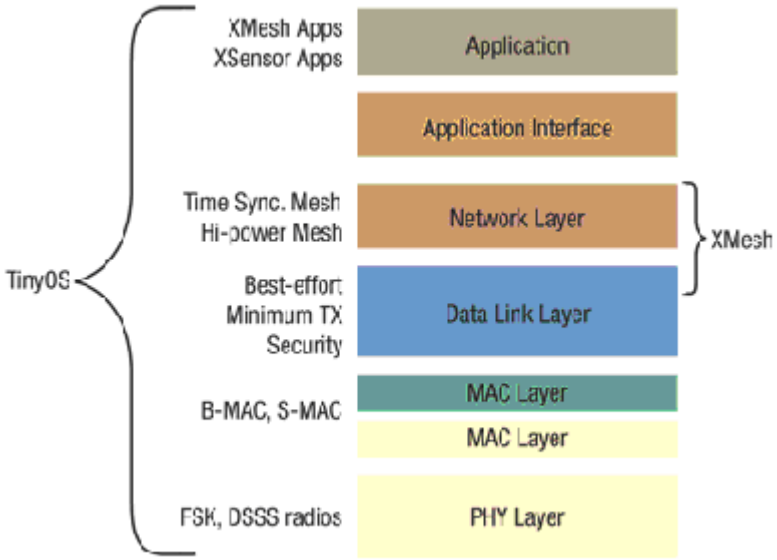


Figure 14. Relationship layout between TinyOS and XMesh Protocol [42]

XMesh's routing techniques are outcome of research by TinyOS community by characterizing different adhoc, multi-hop protocol and performance issues with Crossbow mote platform [42]. The XMesh stack forms dynamically mesh network between nodes with proven ad hoc routing methods like minimum transmission technology to reduce number of radio messages in network, vice versa extending the network life time and supporting high bandwidth. Low power mesh networking is primary feature of XMesh, advance feature of XMesh are implemented with QoS methods [43]. In default mode the XMesh performance has displayed better performance compared to other routing

schemes. Even without the use of any of its advanced QoS features, XMesh forms a reliable deterministic network and the performance is shown to be superior to other techniques including shortest-part, Destination-Sequenced Distance-Vector Routing (DSDVR) , Ad hoc On-Demand Distance Vector (AODV) and other proprietary routing schemes [43].

*nesC* is a programming language used to program Crossbow Micaz motes and it has syntax like C language, but the programming style differs in way as it is more event-driven programming language. It is therefore used to control sensor hardware and react on given events [44]. TinyOS merge an efficient execution model, component model and communication mechanism, therefore nesC is referred as modular language that is built on smaller component, which performs given functionality. The components are called ‘Modules’ and are joined together to larger application called ‘Linking’. Conceptually Modules are like objects and have encapsulated and couple state as functionality. The naming scope in nesC is different from Java and C++ object, which refer to function and variable in global namespace, but in nesC component are purely local namespace. This means that while declaring the functions, a component must also declare the functions that it calls and the name which a component employs to call these functions is purely local [44]. An example to understand this would be that a component ‘A’ declares that it calls a function ‘B’, it is basically initiating the name ‘A.B’ into a global namespace. As well as if a different component ‘C’ that calls a function ‘B’ introduces ‘C.B’ into the global namespace. Eventually both A and C refers to the function B, they might be still referring to completely different implementations. In summary for this, every component has a specification in nesC where a code block declare the functions for which it provides the implementation and function that is uses to call.

### 3.6 Squawk (JVM) on SunSPOT

The Squawk [45] is a virtual machine (VM) written in Java which aim is to run small devices without operating system. Most of the VM are written in C or assembler instead Squawk is written in higher language and uses the same Java language to implement on top of VM. The mechanism squawk uses is isolate mechanism, the goals is to refine

applications. The idea is to run multiple isolates in single VM and those isolates can be migrated to different instances of VM [45]. The main benefit of squawk is that virtual machines is written in java and are easily portable, maintainable and easy to debug. Advent feature is that it is compliant with Connected Limited Device Configuration (CLDC 1.1) which is meant to be used in devices with limited resources such as mobile phones and personal digital assistants. CLDC defines a set of programming interfaces and when coupled with Mobile Information Device Profile (MIDP) it provides a Java platform for developers to write application for devices with limited memory and processing power capacity [46].

At minimum squawk system requires 8K bytes RAM with 32K bytes of EEPROM, also with 160 Kbytes of ROM to have optimized running with 32-bit processor [47]. Figure 15 shows the architecture diagram of Squawk extended from Squeak and KelinVM architecture.

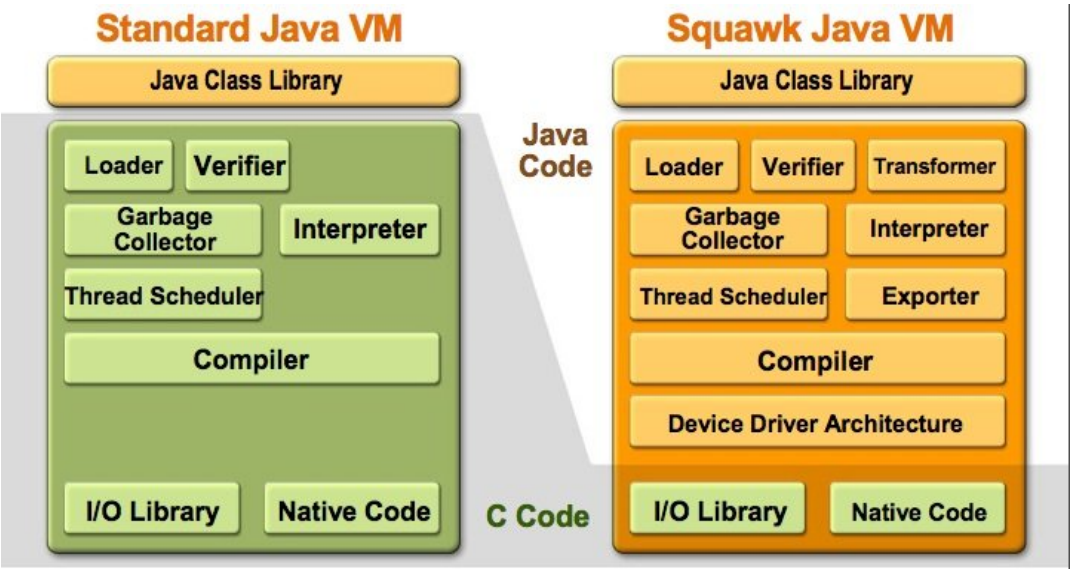


Figure 15. Extended from Squeak and KelinVM architecture [47]

The squawk architecture is a split of two VM which have class-file processor called *translator* as one end and *execution engine* on another end [47]. The translator generates a compact version of input Java byte code, generating properties like symbolic reference to other classes resolving fields and methods. All local variables are re-allocated so that slots can be partitioned to hold pointer or non-pointer values and finally operand stack is

assured to be empty for the instructions which are memory allocated. The final two transformations immense ease garbage collector as method and only require a single pointer map hence resulting in unnecessary scan of operand stack [45]. Table 2 presents the constraints list for Crossbow Micaz sensor and SunSPOT which are used in project.

Table 2. Constraints list of sensor platform used in project work.

| **Sensor Platform** | CrossBow Micaz | SunSPOT |
|---|---|---|
| **Processor** | ATmega-128L | ARM-7 |
| **Data-Security** | AES-128 | Public key, RSA, ECC |
| **Communication-Method** | IEEE 802.15.4 | IEEE 802.15.4 |
| **Operating Frequency** | 900 MHz- 2.4 GHz | 2.4 GHz |
| **Distance Range (Line of Site)** | 135 Meters | 100 Meters |
| **Battery** | 1.5 AA * 2 | 3.6v lithium-ion |
| **Operating System** | TinyOS | No (Squawk JVM) |
| **Programming-Lang** | nesC | Java |
| **External board Connectivity** | Yes | Yes |

# 4. MIDDLEWARE APPROACH TOWARDS SENSOR MONITORING SERVICE

Middleware is used to reduce gap between application and operating system, creating an inner boundary to bridge the complexity and for enhancing the development of distributed applications for any system [48]. WSN have same boundary properties and share many inheritance from traditional distributed system. Even though distributed computing middleware seems suitable for wireless sensor networks. Due to the facts of device limitation and energy constraints in the sensor node, middleware for WSN is approached in a different manner. In this chapter different middleware systems are reviewed and approach to create gateway monitoring service for infrastructure sensor network is taken into account.

## 4.1 WSN and Middleware's

Middleware resides between the operating system and the application Figure 12 previously gives example of it in case of a sensor node. The challenge of WSN middleware is not limited to network, but also to the sensor devices connected to the network [48]. WSN applications are more concerned on real-world data, location and physical environment. Considering a scenario where a large number of different sensor nodes with different sensing capabilities, power source and computing are scatter in heterogeneity.

The question to arise here is "*What if every wireless sensor node has to be operated unattended*"

Therefore middleware designing is the important factor in WSN system. A middleware should provide a mechanism to suppress application knowledge into the WSN infrastructure [49]. Hence the middleware will give the support to the development, deployment and maintenance of WSN and its application, coordinating and splitting the task into sensor nodes and merging data for high level abstraction [48]. The next sections present the middleware approaches for distributed computing systems; *Jini* provides interaction between hardware and software, *Lime* is a middleware system with primary function to provide communication between agents, CORBA is one of the most common

middleware system and *Milan* works on application to indicate policies for managing the network and the sensors.

*Jini* provides a high level of interaction support to both hardware and software services, in a distributed computing environment which can offer network plug and play [50]. Jini's service discovery protocol and leasing method make use of client applications to discover services and handle connections to client-server as set of available services. Service discovery is useful in cases of dynamic sensor networks to know what sensors services are available. Jini specification consist a set of middleware components with application programming interface (API) for creating services, component and a pure Java middleware implementation as package [50]. Hence by including API into classpath as packages the client or service invokes method for Jini middleware protocol for joining Jini services and client.

The *Lime* (Linda in a Mobile Environment) [48] is a middleware system which primary function is to provide communication between agents. Agents are run on host with active tuple space managers. The concept is adopted from Linda model where computation is represented as globally accessible, namely a shared memory scheme for mobile ad hoc components persistent tuple space [48]. The tuple spaces are extended with by notion of location and react to states on given program. Neither Jini nor Lime is overlooking the limited energy constraints of sensor networks and their supporting protocols are heavyweight when compared to protocols tailored to sensor networks [48].

*CORBA* (Common Object Request Broker Architecture) is one of the most common middleware system [51]. The main feature of CORBA is that software components written in different computer language or even running on different platform are integrated together. These integrated standards are given by Object Management Group (OMG) [51]. Further features of CORBA can be classified as it hides the location of remote objects by simplifying the application's interactions with these remote objects. By allowing all operations to appear as they are local, this approach is applicable to sensor networks to provide access to the sensor data as it hides the location of the sensor. On other hand the context information of the sensor is lost. Moreover by giving individual

sensor access with object method the energy saving potential with aggregation is mislaid [51].

Middleware which have been developed for WSN attends to change the properties of network with their own criteria to match the conditions detected within the network. For example middleware like *Limbo* and *FarGo* relocate components by reordering data exchanges to respond to changing network conditions such as bandwidth availability or link reliability [30]. Lower level middleware like *Mobiware* [49] enables support to various levels of QoS by enabling streams within the network with active filters deployed with the routers. Other middleware systems provide hooks to allow the applications to adapt from the network. Other examples like *Odyssey* are platform application which can register for alteration of changes in the core network data rate [48]. These approaches are feasible to wireless sensor networks, but the drawback is that they does not integrate data aggregation protocol of sensor node and sensor network or either take into consideration of low-level wireless protocol.

*Milan* (Middleware Linking Applications and Networks) works on application to indicate policies for managing the network and the sensors [48]. The key feature of Milan is that it adapts network configuration by stating to sensors to either route data, send data or have special requirement on network. *SLIM* (Secured Lightweight Interactive Middleware) hides the complexity of sensor technology with the application layer [30]. It inherits data acquisition and plug-play capability of middleware to further functionality like secure data to unauthorized devices by running middleware on mobile device as a gateway. Other approaches as *Senceive* [52], *TinyDB* [41], *Agilla* [53] and *Cougar* are well fitted on there scenarios as well as global approaches like *Senseweb* [54] and *Open Sensor Web Architecture and Sensor Web* [55].

### 4.2 Semantic Web and Web Services

The World Wide Web is seen as a repository of information containing documents and multimedia resources concerning every possible subject [58]. All this data is spontaneously reachable to everyone with an internet connection. The major success of web based application is due to its decentralized design method where web pages are

hosted by numerous computers and each document can link to other documents, either on the same or different domain computers [52]. Initially web pages were taken in account as simple display of information and later revolutionary search mechanism has given access to user to search information on its need. Search engines are one of the many features from Semantic Web. *Swoogle* is a semantic web search engine that uses ontologies to refine search by using existing ontologies and RDF data from the web. It provides services to user via browser interface and software agents via Restful web services [58]. Another example of semantic web is internet agents acting as autonomous programs to request and perceive web pages and execute web services. For example a user request for flight booking to some destination, then internet agents perform action and provide for user car rental and hotel information to the same destination. These agents rely on webpage information and perform its predefined task making them robust to semantic of webpage. To be able to make a webpage intelligent, computer must not only understand the text but also have ability to understand natural language and its process [52]. Researchers and web developers have proposed and given solutions to enhance the Web with languages that make the meaning of web pages precise [56]. Tim Berners-Lee, the inventor of the Web, has coined the term Semantic Web to describe this approach. Berners-Lee, Hendler and Lassila [57] give the following definition:

*"The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation."*

Figure 16 shows the Semantic Web layer tower which is composed of metadata, ontologies, logic and rules. Metadata is referred to data, a part which gives meaning to all data. Ontology defines the concept and meaning of that data with co-relation to other terms. Rules are associated with ontology and to obtain stated information. Logic provides basis for expressing knowledge and driving new knowledge. Languages to represent ontologies such as RDF, OIL are discussed in the next section.

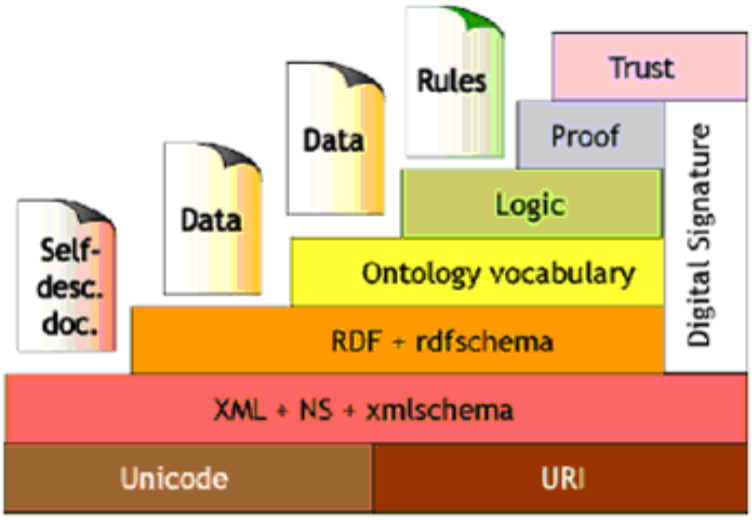


Figure 16. The Semantic Web Layer Tower by Tim Berners-Lee [57]

The aim of *Semantic Web* is to bring machine understandable information on the web and change the way how users browse web and also organize its resources connected with different data [59].

*Web Service* is an application logic that exceeds network, communication protocols, programming languages, operating systems and data representation for the Web. Web Service provides an infrastructure for deploying and developing distributed applications for the web [52]. Web Services are used to expose applications consumption for users with contemporary Web applications [59]. The industry standard for developing and deploying Web Services are eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description Discovery and Integration (UDDI) [51].

Semantic Web and Web Services convergence provides a powerful *Semantic Web Services* concept [56]. Semantic Web Service gives the prospective to access enhanced value added services by autonomously discovering and assembling web services to accomplish a domain task [54]. The framework for semantic web services is known as Service Oriented Computing (SOC) [56].

### 4.3 Semantic Web Languages

The Semantic Web combines all the data from web as RDF schema and its inference language as data repository. The Semantic Web uses distributed data objects framework and therefore validly fits as an Object Oriented Framework [58]. Both the Semantic Web and Object Oriented Programming (OOP) have classes, attributes and instances [59]. The Semantic Web is build on with data and languages which are RDF, XML, OIL, DAML, WSML and WSDL [57].

*Resource Description Framework* (RDF) [59] is used to describe information and resources on the web. The W3C have defined standards and recommendation for XML serialization of RDF called syntax in RDF model. The common interchange format in Semantic web is RDF/ XML. RDF is generic format and the information maps directly and unambiguously to a model [59].

*Ontology Inference Layer / Ontology Interchange Language* (OIL) was developed by Dieter Fensel, Frank van Harmelen and Ian Horrocks [61]. It can be regarded as an ontology infrastructure for the Semantic Web. Ontologies share common understanding of domain and can communicate in different applications and exchange process in different domains [60]. OIL is based on concepts developed in Description Logic (DL) and frame-based systems and is compatible with RDFS [61]. Figure 17 shows the three roots of OIL. Description Logics (DL) describes the knowledge referring as concepts and role limitations used to automatically knowledge representation in expressing structured knowledge with principled way. The frame-based system refers as central model of primitives logics which are classes (frames) with attributes. These frames have attributes only valid with defined classes and different values representation when used with different classes [61].

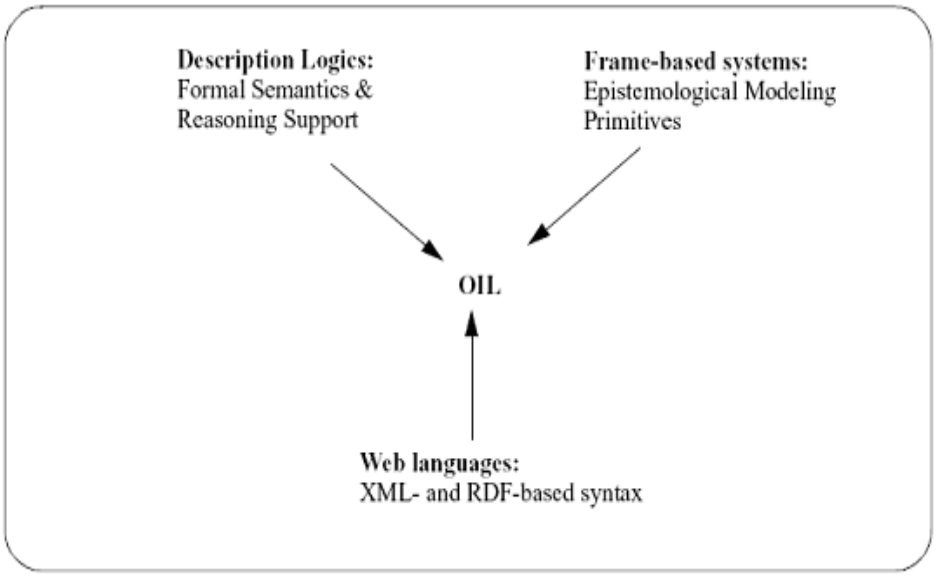


Figure 17. Three roots of OIL [61]

*Web Service Description Language* (WSDL) is a way to define XML description with programmatic access to Web Services with port and messages. A port is network address with defined binding and ports gives the service to requested clients. A message is seen as way to exchange data referred as abstract description of document with port types and supported operations. In general cases WSDL is used with SOAP by which client can request the operation from WSDL operations list [59].

### 4.4 Service Oriented Architecture for Sensor Networks

Service Oriented Architecture (SOA) defines a distributed software architecture which depends on web services for putting together a system [62]. SOA provides a mechanism to describe, discover and invoke services from various systems. The common overview of SOA is a client application or system lookup for the services which are registered to the service directory [62]. The services are referred to not only as software but any hardware devices that can enhance the work formation for users. XML format is standard for passing data whereas the web services can be based on protocols such as SOAP, WSDL and UDDI [62].

The actual SOA for sensor network is *OpenGIS Consortium* (OGC) [63]. OGC defines a Sensor Web Enablement (SWE) which is composed set of observation and measurement. These observation and measurements are useful for sensor collection service, sensor planning service and web notification service [62]. *Sensor Model Language Standard* (SensorML) specification is used to write models in XML encoding to provide a geometric, dynamic and observational characteristics of sensor systems framework [63].

The key point is that the different sensors are supported by atomic process model and process chain. In addition in SensorML all processes and components are programmed as application schema of the model in Geographic Model Language which is a part of OGC-SWE suite standards [64].

The prompt feature of OGC's SWE is that it allows developers to work with all kind of sensors and their data repositories making them discoverable, manageable and useable through the *Web*. OGC's SWE not just allows developers to create discovery of sensors, processes and observations, also it helps developers to stir up tasking to sensors models. These sensor models are useful to create controlled access to observations and observation streams, by putting up publish-subscribe capabilities for robust sensor system and process descriptions [65].

### 4.5 WSN Application Approach to Sensor Monitoring Service

The approach to build a monitoring service for infrastructure sensor network is taken in account through SOA. The development starts from programming the sensor board continuing to the development of a desktop host application. The host application named *gateway* collects sensor data and store sensor data in database, gateway is also a web server and provides web services for sensor monitoring to the browser based client and mobile phone client.

Figure 18 presents the software architecture diagram of the complete system. The ovals refer to sensor nodes which sense light, temperature and axis degree accelerometer, sensors are connected to the basestation in next level which is running on TinyOS and Squawk. The nesC and Java presents programming languages used to control and

program basestation and sensor nodes. J2SE application stores the generated sensor information to PostgreSQL database. Lastly the web services are connected to PostgreSQL which provides requested sensor information to mobile client.

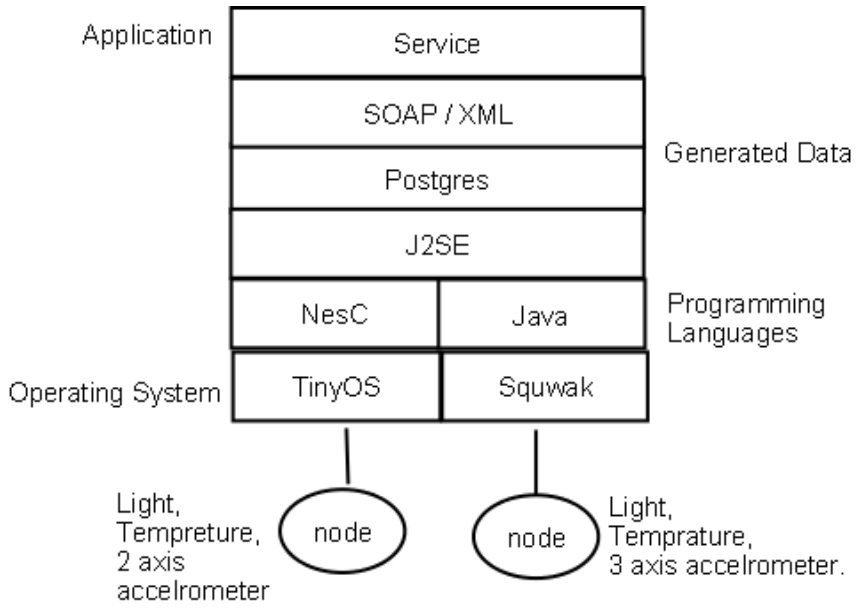


Figure 18. Software Architecture Diagram for Gateway Monitor Service for Sensor Network

SOA based approach is well suited for the project as it supports distributed software architecture and service delivery to client relies on by web services. Figure 19 presents the SOA based architecture for sensor monitoring service. The general idea is that client request for sensor data which is registered in web service *Sensor Directory* and requests the information of selected sensor via service. The service is also used as to describe and publish the sensor information to sensor directory. For the communication between client and web services is used SOAP protocol and XML as data format. The web service architecture is composed of SOAP and WSDL protocol.

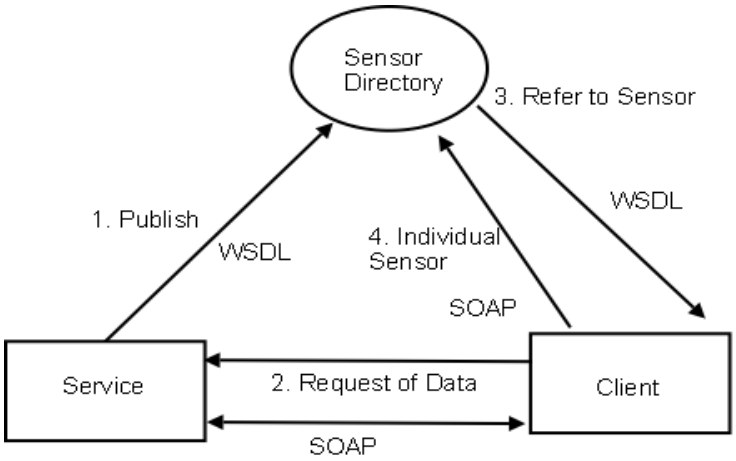


Figure 19. Web Service Architecture of Sensor Monitoring Service

Table 3 shows the result of studied and tested web services framework. The web services framework was tested on Dell Optilex Desktop system with 2.3 GHz processor, 2GB RAM and 80 GB hard-disk running on Windows XP operating system. The numerical values in Table 3 represent number of hours spend for testing and studies. Property *Environment Setting* represents installing and configuring framework, *Uptime* writing sample web service and testing. The *Resource* is referring to the computer memory utilization. The reason to select *NuSOAP* for project work was solely due to less computer memory utilization and ease of development with PHP programming language.

Table 3. Tested and Studied Web Service Framework

| **Web-Service Framework** | **Environment Setting** | **Studied** | **Uptime** | **Programming Language** | **Resource** |
|---|---|---|---|---|---|
| Apache Axis 2 | 1 | 2 | 1 | Java | High |
| .Net Framework | 2 | Less then 1 | 1 | C# | High |
| NuSOAP | 2 | 2 | 1.3 | PHP | Less |

## 5. SYSTEM IMPLEMENTATION

This chapter gives details for developing the application for monitoring service of infrastructure wireless sensor networks. Figure 20 shows the finalized solution developed for infrastructure sensor network monitoring with used technologies. Initially sensor boards are programmed to communicate and send sensing data to basestation device over defined default radio link. The database application is directly connected to basestation device which is considered as information and data collector hub gateway. Gateway application is J2SE application which reads information from COM-7 & 9 port and store the information to PostgresSQL database. After collecting the data from sensor and storing to database, the *Apache Web Server* is setup to provide the web client application and requests. Web service is written in programming language PHP with extension of NuSOAP, in order to facilitate the mobile client request via created WSDL. The client application request data from the web service which is returned to client via SOAP message exchange. More details about the web service to facilitate mobile client and web client are explained later in this chapter and part of programming code is shown to ease the understanding of system application. The source code for gateway application and web service is given in the appendix 1 and 2.

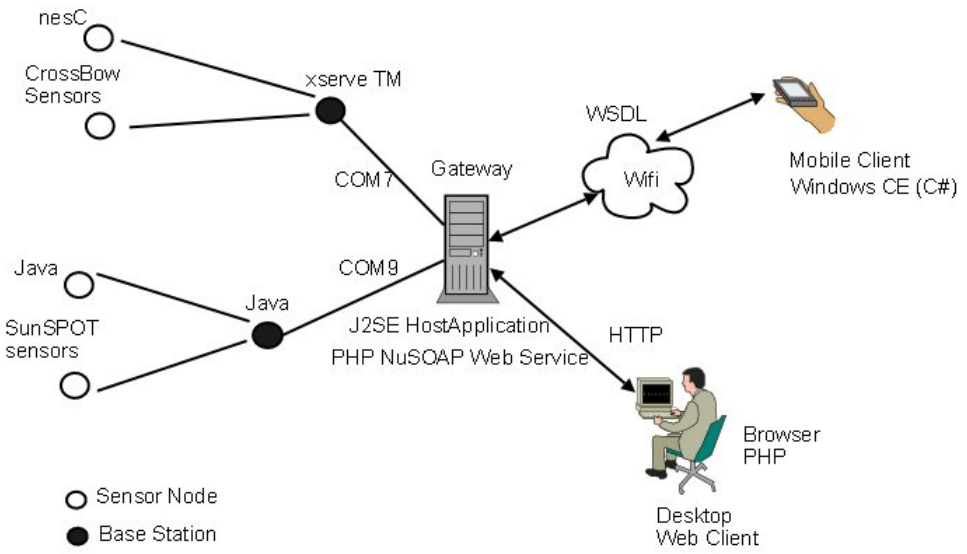


Figure 20. Details of technologies used in infrastructure wireless sensor network monitoring service.

### 5.1 Programming Sensor Boards

Two of the CrossBow Inc sensor devices are used with the basestation. The sensor board is XMTS400 shown in Figure 9, which is connected to MIB520 programming board. The default sample application with slight modification for XMTS400 is used to send data to basestation with XMesh routing protocol. The application to collect data from XMTS400 sensor device is *xserve* a CrossBow Technology propriety application. In second phase the configuration parameters of xserve were changed to store sensor data in PostgreSQL 8.4 because by default it works only with PostgreSQL 8.0. Finally the application is compiled for TinyOS under *cygwin* and loaded to XMTS400. Cygwin is Linux like environment for Windows. The application for SunSPOT sensor board is written from scratch under Netbeans 6.0 editor. Netbeans 6.0 editor for SunSPOT comes with required libraries for developing application for SunSPOT. For the SunSPOT sensor in the *StartApplication* only public class defines the sensor leds, light, temperature instances. Figure 21 shows code snippet for StartApplication class for sensor board.

```
public class StartApplication extends MIDlet {

    private ITriColorLED [] leds = EDemoBoard.getInstance().getLEDs();
    private ISwitch sw1, sw2;
    private IAccelerometer3D accel =
EDemoBoard.getInstance().getAccelerometer();
    private ITemperatureInput tempSensor =
EDemoBoard.getInstance().getADCTemperature();
    private ILightSensor lightSensor =
EDemoBoard.getInstance().getLightSensor();
    EDemoBoard demoboard = EDemoBoard.getInstance();
    IAccelerometer3D acc = demoboard.getAccelerometer();

    protected void startApp()
            throws MIDletStateChangeException
    {
new BootloaderListener().start(); // monitor the USB (if connected) and
recognize commands from host
```

Figure 21. Code snippet of class StartApplication.

If the condition is true the sensor broad is set to connect with SunSPOT host application on radiogram port 99. Figure 22 shows the code snippet of connection to base station hardware address, where hexadecimal address is of base station and port number is 99.

```
RadiogramConnection conn = (RadiogramConnection)
Connector.open("radiogram://0014.4F01.0000.4CB9:99");
```

Figure 22. Code snippet for Radiogram connection for SunSPOT.

SunSPOT basestation device is more like transparent device to sensors as the main application for collecting data is running on desktop computer. Only the base station device is allowing radio port communication from sensor and forwarding data to desktop computer at COM9 port. Figure 23 shows the class diagram of SunSPOT host application for sensor data collection and generating XML files for web client usage and information.

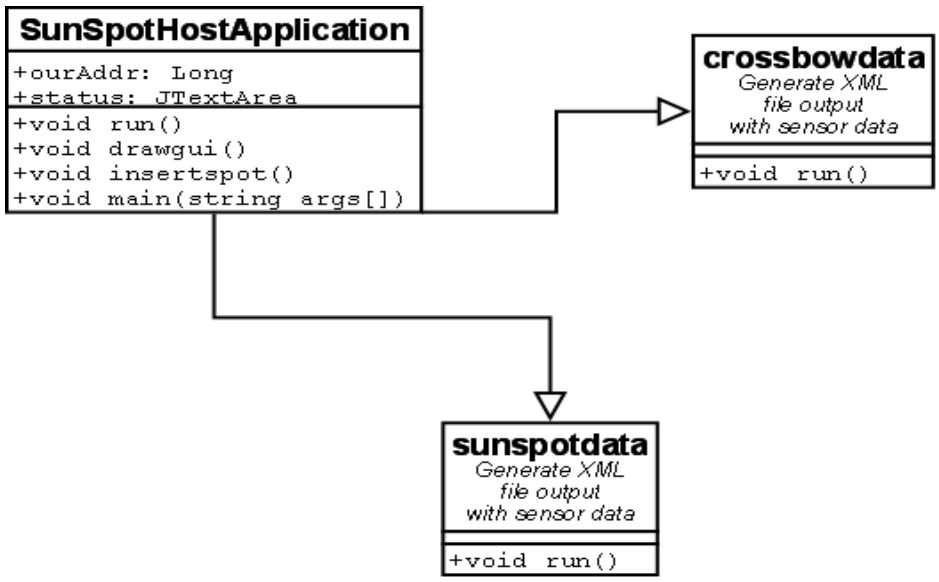


Figure 23. Class diagram of SunSPOT main application.

### 5.2 Setting Host Machine & Collecting Data

For sensor data collection and storing PostgreSQL 8.4 was database of choice, since it is supported by Crossbow Technology default application xserve. For SunSPOT sensor data collection the Netbeans editor is configured with Java Database Connectivity (JDBC) with PostgresSQL plugin. The database is created with name ‘task’ with two tables. One table for the Crossbow sensor named mts400_results and another table spot_results for SunSPOT sensor. Both tables have respective fields for sensor data for example nodeid, temp, light and degree (xaxis, yaxis and zaxis). Figure 24 shows the code snippet for calling the JDBC in SunSPOT host application.

```
Class.forName("org.postgresql.Driver");
Connection c= null;
    c =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/task",
                                "postgres", "b0331969");
    Statement select = c.createStatement();
```

Figure 24. Code snippet for JDBC in HostApplication.

Apache 2.2 web server [66] for Windows, is configured for running web service and web application. In addition PHP 5.0 is configured to develop web application. Jpgraph library [67] in PHP is configured to generate usage and timing graph of sensor node information from database tables.

### 5.3 Implementing Web Services for Sensor Information

The Web Services are programmed under PHP using NuSOAP [68] which is set of classes to develop and consume Web Service in SOAP and WSDL. The defined web services performs parameter value requested from mobile client, each parameter value runs select query on database table according to requested sensor from client mobile interface. Figure 25 shows the message sequence chart (MSC) of mobile client and web services. The client initiate connection to the web server as shown in Figure 25, after successful connection screen two on mobile interface allows user to select the sensor nodes to view its information by sending parameter value for example ‘a’ for sensor 1. The web service connects to database and runs the query for specific sensor, and return sensor data result to the web service. Final stage of messaging is the web service returns the sensor information to mobile client in XML/SOAP format. Figure 29 and 30 gives the overview how the result display on mobile client interfaces.

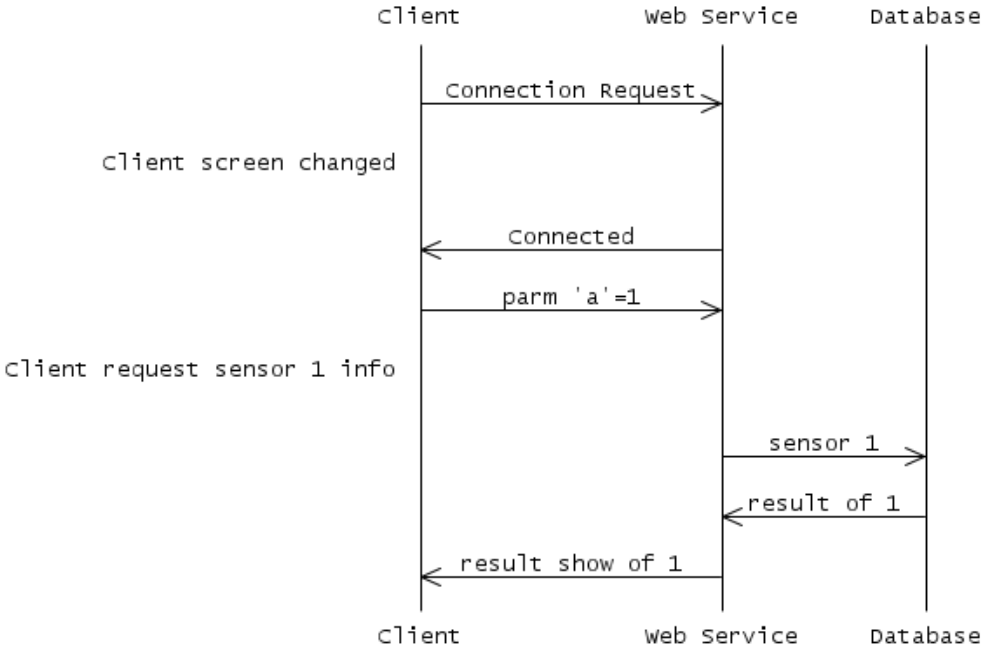


Figure 25. MSC between Mobile client and web services

### 5.4 Client Interfaces Web based & Mobile based

The interfaces for client to view sensor information are browser based and mobile phone based. The web based interface is where client can view all the information from sensor database through web browser. Browser based client can also view the generated graph of sensor information showing for example temperature changes in week or month on specific sensor. The web interface is developed with PHP are login screen page shown in Figure 26, view data page shown in Figure 27 and view graph page shown in Figure 28.

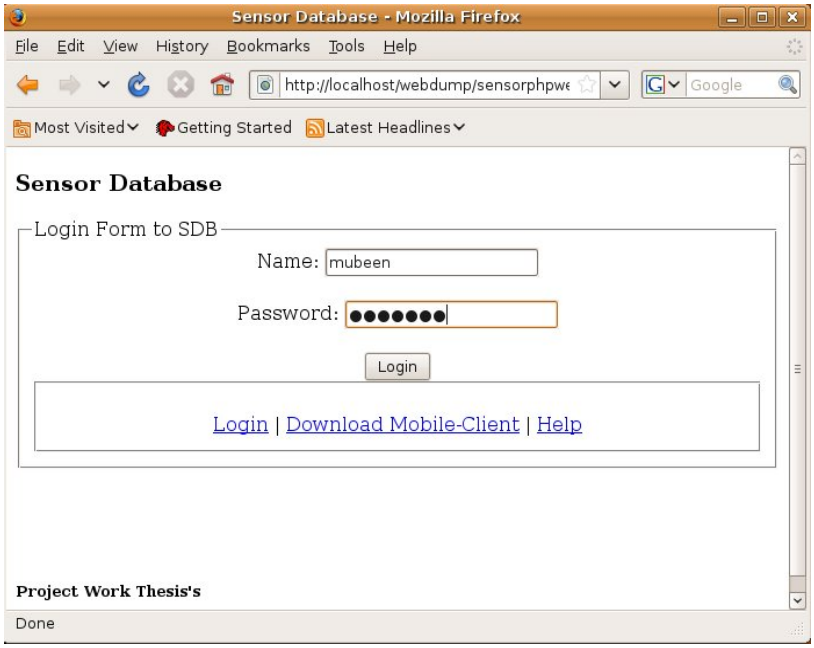


Figure 26. Login screen for sensor view

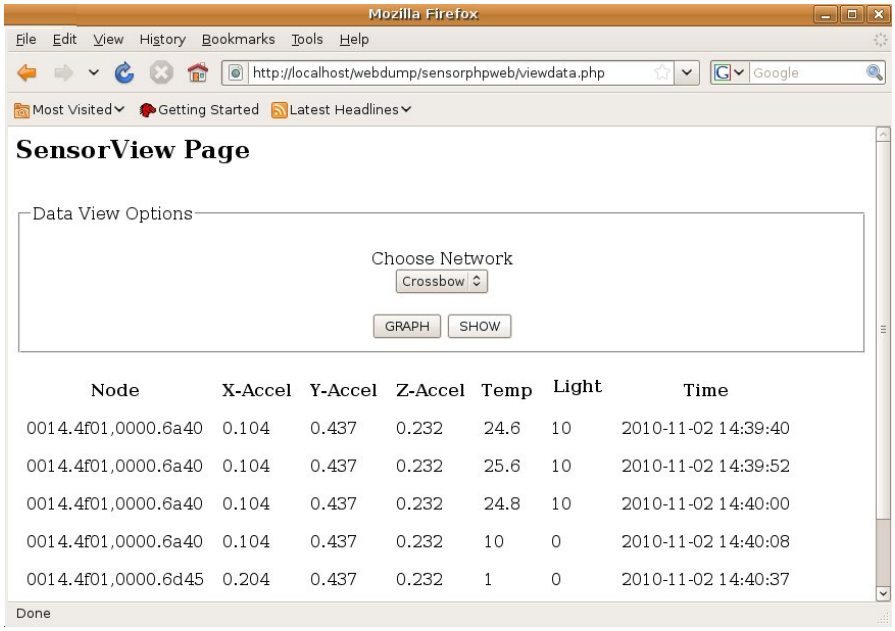


Figure 27. Sensor View Page

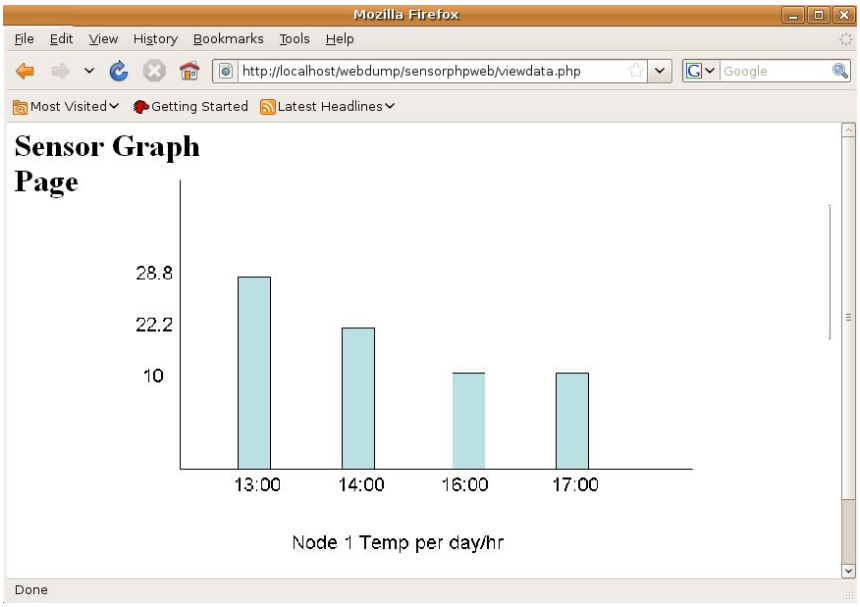


Figure 28. Sensor Graph View Page

Second interface is Mobile client in Windows Mobile (WM) which facilitates mobile client users. Mobile client can check only current status of provided sensor through his mobile phone as shown in Figure 29 and Figure 30. The programming language used for mobile interface and event actions is done in C#.


Figure 29. Mobile Initial Screen


Figure 30. Sensor Info Screen

## 6. CONCLUSIONS

The primary objective of the thesis was to develop gateway monitoring application for infrastructure sensor networks. Real time monitoring system based on wireless sensor networks and its application has been studied. A feasible application has been developed which can fit from normal to average use of home or small scale area monitoring.

The overall topic of wireless sensor network has different level of research areas such as of MAC layer, nodes and protocol security, routing enhancement protocol to energy efficiency. The main focus in this thesis work was to develop monitoring application for wireless sensor network. Therefore sensor middlewares are reviewed based on different approaches such as Microsoft research project *SenseWeb* which acts like a common sensor data repository and application such as *SensorMap* that is build on top of it using the sensor data. In addition Open GIC Consortium (OGC) has been studied, which introduce Sensor Web Enablement (SWE) concept providing actually a set of specifications. These specifications include Sensor Markup Language, Observation & Measurement, Sensor Collection Service, Sensor Planning Service and Web Notification Service to implement the Sensor Web. The Open Sensor Web Architecture (OSWA) extends the SWE and integrates the Sensor Web and grid computing by providing middleware support for Sensor Web. Also concept of Service Oriented Architecture approach is studied and used as key development method in thesis project work, which is useful method to describe and invoke services on heterogeneous platform using SOAP and XML standard.

The initial challenge in the project work was faced with setting up development environment for SunSPOT, although the technical documentation refer that Netbeans 6.0 can be configured in Linux for SunSPOT. The basestation doesn’t respond properly under Linux although it works better in Windows OS. Another challenge was faced in configuring the Crossbow Micaz sensor to store data in PostgreSQL 8.4. This problem was tackled by changing the configuration script of XMTS400 sensor board. Finally for selecting the web service framework after testing Apache Axis 2 and .Net Framework,

NuSOAP was chosen because it provides yet powerful and simple web service framework with easily deployed and development methods.

The developed application suits well for standard infrastructure sensor monitoring and applicable in the home or small scale of environment monitoring. In the work is used environmental sensors to measure temperature, light, humidity, accelerometer-degree and the developed system is excellent example of real time system and its monitoring. This thesis work can be further enhanced by creating common protocol for both Micaz and SunSPOT sensor networks, making possible to apply single basestation node for both sensor networks. Another enhancement could be a middleware test bed to integrate different or heterogeneous sensor data and provide common API for all sensors nodes.

**REFERENCES**

[1] Jeremy Elson and Deborah Estrin, *Sensor networks: a bridge to the physical world*, Center for Embedded Networked Sensing, University of California, Wireless Sensor Networks, p.p. 3-20, 2004 ACM, ISBN:1-4020-7883-8.

[2] Shijin Dia, Xiaorong Jing, Lemin Li, *Research and analysis on Routing Protocols for Wireless Sensor Networks*, Proceedings of International Conference on Communications, Circuits and Systems, p.p.407-411, University of Electronics Science and Technology of China, May 2005 IEEE, ISBN: 0-7803-9025-6.

[3] Chonggang Wang; Sohraby, K.; Daneshmand, M.; Bo Li, Yueming Hu, Arkansas Univ., AR, USA, *A Survey Of Transport Protocols For Wireless Sensor Networks*, IEEE Network, p.p. 34-40, June 2006 IEEE, ISSN: 0890-8044.

[4] Holger Karl and Andreas Willig , *Protocols and Architecture for Wireless Sensor Networks*,. John Wiley & Sons Ltd 2005. ISBN: 0-470-09510-5.

[5] Chee-Yee Chong, Kumar, S.P. Booz Allen Hamilton, *Sensor networks: evolution, opportunities, and challenges*, Proceeding of IEEE, p.p. 1247-1256, August 2003 IEEE, ISSN: 0018-9219.

[6] Stefano B, Marco C, Silvia G, Ivan S, *Mobile Ad Hoc Networking, 'Mobile ad-Hoc networking with a View of 4G Wireless'*, John Wiley & Sons Ltd 2004 , ISBN: 0-0471-37313-3

[7] Seapahn Meguerdichian, Farinaz Koushanfar, Miodrag Potkonjak, Mani B. Srivastava, *Coverage Problems in Wireless Ad-hoc Sensor Networks*, Poceedings of IEEE. INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. p.p. 1380-1387 vol.3,University of California LA, IEEE 2001, ISBN: 0-7803-7016-3.

[8] Kyildiz, I.F, Weilian Su, Sankarasubramaniam, Y Cayirci, *A Survey On Sensor Networks*, IEEE Communications Magazine, p.p. 102-114 vol.40, Atlanta, GA, August 2002 IEEE, ISSN: 0163-6804.

[9] Chris Townsend, Steven Arms, *Sensor Technology Handbook, 'Principles and Applications'*. MicroStrain, Inc. Book 2005, ISBN: 0-7506-7729-5.

[10] José Carlos , Teresa Olivares and Luis Orozco-Barbosa, *Routing protocols for wireless sensor networks-based network, Technical Report,* Albacete Research Institute of Informatics University of Castilla-La Mancha [pdf: accessed 08.07.2010] (http://www.dsi.uclm.es/descargas/thecnicalreports/DIAB-07-06-1/tehcnicalreport.pdf)

[11] Kemal Akkaya and Mohamed Younis, *A Survey On Routing Protocols For Wireless Sensor Networks*, Ad Hoc Networks, vol 3, Issue 3, May 2005, p.p. 325-349 , Elsevier, ISSN: 1570-8705

[12] W. Heinzelman, J. Kulik and H. Balakrishnan, *Adaptive Protocols for Information Dissemination in Wireless Sensor Networks*, Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, p.p 174-185, Seattle, Washington, USA 1999,ACM, ISBN: 1-58113-142-9.

[13] Md. Habibe Azam, Abdullah-Al-Nahid, Md. Abdul Alim, Md. Ziaul Amin, *A Survey and Comparison of Various Routing Protocols of Wireless Sensor Network (WSN) and a Proposed New TTDD Protocol Based on LEACH*, (IJCNS) International Journal of Computer and Network Security, Vol. 2, No. 8, August 2010.

[14] Lindsey, S. Raghavendra, *PEGASIS: Power-efficient gathering in sensor information systems*, Proceedings Aerospace Conference, 2002. IEEE, p.p 3-1125 - 3-1130 vol.3 , Los Angeles, CA, USA , IEEE 2002, ISBN: 0-7803-7231-X.

[15] Laiali Almazaydeh, Eman Abdelfattah, Manal Al- Bzoor, and Amer Al- Rahayfeh, *Performance Evaluation of Routing Protocol in Wireless Sensor Networks*, International Journal of Computer Science and Information Technology, Volume 2, Number 2, April 2010.

[16] Yu Y, Govindan R, Estrin D, *Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks*, UCLA Computer Science Department Technical Report UCLA/CSDTR-01-0023, Citeseer May 2001.

[17] Yongling Guo; Qianping Wang; Hai Huang; Wei Tan; Guoxia Zhang, *The Research and Design of Routing Protocols of Wireless Sensor Network in Coal Mine Data Acquisition*, International Conference on Information Acquisition, 2007. ICIA '07. July 2007,p.p. 25 - 28, China ,Xuzhou, July 2007 IEEE, ISBN: 1-4244-1220-X.

[18] McDermott-Wells, P., *"What is Bluetooth?,"* Potentials, IEEE , vol.23, no.5, pp. 33-35, Dec. 2004-Jan. 2005, ISSN: 0278-6648.

[19] Timmons, N.F.; Scanlon, W.G.;Fac. of Eng., Letterkenny , *Analysis of The Performance of IEEE 802.15.4 For Medical Sensor Body Area Networking*, First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. p.p. 16-24, Ireland, IEEE 2004, ISBN: 0-7803-8796-1.

[20] Bob Heile, *"ZigBee Alliance Overview"* ZigBee Tutorials, [http://www.zigbee.org/LearnMore/Tutorials.aspx]. Accessed 22 August 2010

[21] Nemeth-Johannes, J.; Sweetser, V.; Sweetser, D.; *Implementation of an ieee-1451.0/1451.5 compliant wireless sensor module*, IEEE Autotestcon 2007,p.p.364 - 371,Baltimore, MD , IEEE 2007. ISSN: 1088-7725.

[22] Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, *"Wireless Sensor Networks for Habitat Monitoring"*, Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications 2002,Atlanta, Georgia, USA, p.p. 88 - 97 , ACM, ISBN:1-58113-589-0.

[23] K. Martinez, R. Ong, J. K. Hart, and J. Stefanov, *"Glacsweb: a sensor network for hostile environments"*, First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004, p.p. 81-87, IEEE, ISBN: 0-7803-8796-1

[24] F. Michahelles, P. Matter, A. Schmidt, and B. Schiele, *"Applying Wearable Sensors to Avalanche Rescue"*, Computers and Graphics, Volume 27, Number 6, pp. 839-847, December 2003.

[25] H. Baldus, K. Klabunde, and G. Muesch, *"Reliable Set-Up of Medical Body-Sensor Networks"*, Proc. EWSN 2004, Berlin, Germany, pp. 353-363, January 19-21, 2004.

[26] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, *"A two-tier data dissemination model for large-scale wireless sensor networks"*, Proceedings of the 8th Annual International Conference on Mobile Computing and Networking, September 23-28, 2002, Atlanta, Georgia, USA, pp. 148–159, ACM Press, 2002.

[27] Woo Kwon Koo, Hwaseong Lee, Yong Ho Kim, Dong Hoon Lee, *Implementation and Analysis of New Lightweight Cryptographic Algorithm Suitable for Wireless Sensor Networks*, Proceedings of the 2008 International Conference on Information Security and Assurance, p.p. 73-76, Busan 2008, IEEE Computer Society. ISBN:978-0-7695-3126-7.

[28] Taeshik Shon Bonhyun Koo Hyohyun Choi ,Yongsuk Park, U-Convergence Lab., Samsung Electron., *Security Architecture for IEEE 802.15.4-based Wireless Sensor Network*, Wireless Pervasive Computing, 2009. ISWPC 2009. 4th International Symposium on Suwon, p.p. 1-5,Melbourne 2009, IEEE 2009, ISBN: 978-1-4244-2965-3.

[29] Sharifi, M.; Ardakani, S.P.; Kashi, S.S., *SKEW: An efficient Self Key Establishment protocol for Wireless sensor networks*, International Symposium on Collaborative Technologies and Systems, 2009. CTS '09, Baltimore, MD, 18-22 May 2009, p.p. 250-270, IEEE 2009, ISBN: 978-1-4244-4584-4.

[30] Agustinus Borgy Waluyo, Isaac Pek, Xiang Chen, Wee-Soon Yeoh *SLIM: A Secured Lightweight Interactive Middleware for Wireless Body Area Network*, 30th Annual International IEEE EMBS Conference Vancouver, British Columbia, Canada, August 20-24, 2008, p.p. 1821-1824, IEEE 2008, ISBN: 978-1-4244-1814-5.

[31] Holger Karl and Andreas Willig *Protocols and Architecture for Wireless Sensor Networks, 'Single-Node Architecture'*, John Wiley & Sons Ltd 2005. ISBN: 0-470-09510-5.

[32] *Corssbow 2007 Wireless Product Catlog*, PDF (accessed 11.March.2010) [http://www.investigacion.frc.utn.edu.ar/sensores/Equipamiento/Wireless/Crossbow_Wireless_2007_Catalog.pdf]

[33] *XServe Users Manual* (Crossbow Inc), PDF (accessed 11.March.2010) [http://www.xbow.com.cn/LinkClick.aspx?fileticket=UjCWY6KXa78%3D&tabid=121]

[34] Sun™ Small Programmable Object Technology (Sun SPOT) *Developer's Guide*, PDF (accessed 05.August.2010), [http://www.sunspotworld.com/docs/Purple/spot-developers-guide.pdf]

[35] Sun™ Small Programmable Object Technology (Sun SPOT) *Theory of Operation*, PDF (accessed 07.August.2010) [http://sunspotworld.com/docs/Purple/SunSPOT-TheoryOfOperation.pdf]

[36] Vlado Handziski, Joseph Polastrey, Jan Hinrich Hauer, Cory Sharpy, Adam Wolisz and David Cullery, *Flexible Hardware Abstraction for Wireless Sensor Networks,* Proceedings of the Second European Workshop on Wireless Sensor Networks, 2005, p.p. 145-157, IEEE 2005, ISBN: 0-7803-8801-1.

[37] Adi Mallikarjuna Reddy V AVU Phani Kumar, D Janakiram, and G Ashok Kumar, *Operating Systems for Wireless Sensor Networks: A Survey Technical Report* , International Journal of Sensor Networks (IJSNet), Vol. 5, No. 4 2009, pp. 236 – 255. ACM, ISSN: 1748-1279

[38] Mauri Kuorilehto, Timo Alho , Marko Hännikäinen, and Timo D. Hämäläinen, *SensorOS: A New Operating System for Time Critical WSN Applications,* Proceedings of the 7th international conference on Embedded computer systems: architectures, modeling, and simulation, p.p. 431-442, Greece 2007, ACM, ISBN:0302-9743.

[39] P. Levis,S. Madden,J. Polastre,R. Szewczyk,K. Whitehouse,A. Woo,D. Gay,J. Hill,M. Welsh,E. Brewer,D. Culler.,*TinyOS: An Operating System for Sensor Networks,* Book: Ambient Intelligence, Springer 2005, ISBN: 978-3-540-27139-0

[40] *TinyOS Programming Manual,* Philip Levis, PDF accessed (27.07.2010) [http://csl.stanford.edu/~pal/pubs/tinyos-programming.pdf ]

[41] Samuel R. Madden,Michael J. Franklin, Joseph M. Hellerstein,Wei Hong, *TinyDB: an acquisitional query processing system for sensor networks,* ACM Transactions on Database Systems Volume 30, p.p. 122-173, ACM March 2005, ISSN: 0362-5915.

[42] Teo A,Singh G.,McEachenJ.C., *Evaluation of the XMesh Routing Protocol in Wireless Sensor Networks,* 49th IEEE International Midwest Symposium on Circuits and Systems, 2006. MWSCAS '06. San Juan 6-9 Aug 2006, p.p. 1548-3746, IEEE 2007, ISBN: 1-4244-0172-0.

[43] *XMesh User's Manual,* Revision D, April 2007, Crossbow Inc, PDF accessed (22.07.2010) [http://www.memsic.com/support/documentation/wireless-sensor-networks/category/6-user-manuals.html?download=95%3Axmesh-user-s-manual]

[44] David Gay,Philip Levis,Robert von Behren,Matt Welsh,Eric Brewer,David Culler , *The nesC language: A holistic approach to networked embedded systems,* Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation, San Diego USA 2003, p.p. 1-11, ACM 2003, ISBN:1-58113-662-5.

[45] Doug Simon, Cristina Cifuentes, *The Squawk Virtual Machine: Java(TM) on the Bare Metal,* Conference on Object Oriented Programming Systems Languages and Applications: Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications,San Diego, CA, USA p.p. 150 - 151, ACM 2005, ISBN:1-59593-332-6.

[46] *Connected Limited Device Configuration* (CLDC); JSR 139, web page (accessed 19.08.2010) [http://java.sun.com/products/cldc/]

[47] *The Squawk Project,* web page (accessed 19.08.2010) [http://labs.oracle.com/projects/squawk/]

[48] Heinzelman, W.B, Murphy A.L.,Carvalho H.S.,Perillo M.A., *Middleware to support sensor network applications,* IEEE Network, p.p. 6-14, Jan/Feb 2004, IEEE Feb 2004, ISSN: 0890-8044

[49] Kay Römer,Oliver Kasten,Friedemann Mattern, *Middleware challenges for wireless sensor networks,* ACM SIGMOBILE Mobile Computing and Communications, Volume 6 October 2002, p.p. 59-61, ACM 2002, ISSN:1559-1662

[50] Jan New March, *Jini 2 Programming,* Apress Inc 2006, '*Overview of Jini*', ISBN-10: 1-59059-716.8

[51] Yang Yu,Bhaskar Krishnamachari,Prasanna V.K., *Issues in designing middleware for wireless sensor networks,* IEEE Network, Jan/Feb 2004,Vol 18 ,p.p. 15 - 21,Los Angeles, USA, IEEE Feb 2004, ISSN: 0890-8044

[52] Hermann, C. Dargie, W. Tech. Univ. of Dresden, *Senceive: A Middleware for a Wireless Sensor Network,* Dresden, 22nd International Conference on Advanced Information Networking and Applications AINA 2008, 25-28 March 2008, Okinawa, p.p. 612-619 IEEE 2008, ISBN: 978-0-7695-3095-6.

[53] Chien-Liang Fok,Gruia-Catalin Roman,Chenyang Lu, *Agilla: A mobile agent middleware for self-adaptive wireless sensor networks,* ACM Transactions on Autonomous and Adaptive Systems (TAAS),Volume 4 2009,Article No:16, ACM 2009, ISSN:1556-4665.

[54] Aman Kansal,Suman Nath,Jie Liu,Feng Zhao, Microsoft Research, *SenseWeb: An Infrastructure for Shared Sensing,* IEEE Multimedia, Oct.-Dec. 2007,Vol 14, Issue:4 p.p. 8 - 13, IEEE 2007, ISSN:1070-986X .

[55] Mike Botts, George Percivall, Carl Reed and John Davidson, *OGC® Sensor Web Enablement: Overview and High Level Architecture,* GeoSensor Networks Second International Conference, GSN 2006, Boston, MA, USA, October 1-3, 2006 p.p. 72-86, Springer 2008, ISBN: 978-3-540-79995-5.

[56] McIlraith, S.A.,Son T.C.,Honglei Zeng, Knowledge Syst. Lab., Stanford Univ., CA, USA, *Semantic Web services,* IEEE Intelligent Systems, Mar-Apr 2001,Vol 16 ,p.p. 46 - 53, IEEE 2001, ISSN: 1541-1672.

[57] Nigel Shadbolt,Tim Berners-Lee,Wendy Hall, *The Semantic Web Revisited,* IEEE Intelligent Systems archive, Vol 21 May 2006, p.p. 96 - 101, ACM 2006, ISSN: 1541-1672.

[58] *Swoolge Project* HomePage, Web Page (accessed 19.08.2010), [http://pear.cs.umbc.edu/swoogle]

[59] *Semantic Web Tools*: An Overview, PDF (accessed 07.07.2010) [http://iam.inflibnet.ac.in:8080/dxml/handle/1944/1030]

[60] Li Ding, Pranam Kolari, Zhongli Ding, Sasikanth Avancha, Tim Finin, Anupam Joshi,University of Maryland Baltimore County Baltimore, *Using Ontologies in the Semantic Web: A Survey, Ontologies A Handbook of Principles, Concepts and Applications in Information Systems,* Springer 2007, ISBN: 978-0-387-37022-4.

[61] Fensel D.,van Harmelen, F.,Horrocks I.,McGuinness D.L.,Patel-Schneider P.F.,Vrije Univ.Amsterdam, Netherlands, *OIL: an ontology infrastructure for the Semantic Web,* IEEE Intelligent Systems, Mar-Apr 2001,Vol 16, p. 38 - 45 , IEEE 2001 , ISSN: 1541-1672.

[62] Xingchen Chu and Rajkumar Buyya, *Service Oriented Sensor Web,* Sensor Networks and Configuration 2007, Springer June 2007, ISBN: 978-3-540-37366-7.

[63] OGC, *Sensor Model Language*, web page (accessed 06.08.2010) [http://www.opengeospatial.org/standards/sensorml#overview]

[64] OGC, *Introduction of SensorML*, web page (accessed 06.08.2010)
[ http://www.ogcnetwork.net/SensorML_Intro ].

[65] OGC, *Sensor Web Enablement*, web page (accessed 06.08.2010)
[http://www.ogcnetwork.net/SWE ].

[66] *Apache HTTP Server Project*, web page (accessed 01.12.2009)
[http://httpd.apache.org/download.cgi]

[67] *JpGraph* , Graph Library for PHP, web page (accessed 25.12.2009)
[http://jpgraph.net/]

[68] *NuSOAP*, PHP classes for Web Services, (accessed 04.01.2010)
[http://www.scottnichol.com/nusoapintro.htm]

**APPENDIX 1.**

The code below shows the Gateway Application developed in by using J2SE for collecting sensor data and generating XML data.

```
/*
 * SunSpotHostApplication.java
 *
 * Created on 12.10.2009 16:03:27;
 */

package org.sunspotworld;

import com.sun.spot.peripheral.radio.RadioFactory;
import com.sun.spot.peripheral.radio.IRadioPolicyManager;
import com.sun.spot.io.j2me.radiostream.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.IEEEAddress;
import com.sun.spot.peripheral.TimeoutException;
import com.sun.spot.util.IEEEAddress;
import javax.microedition.io.*;
import com.sun.spot.*;

import java.io.*;
import java.io.IOException;
import java.io.EOFException;
import javax.microedition.io.*;
import com.sun.spot.io.j2me.radiogram.*;


import javax.swing.*;
import java.lang.*;


import java.sql.*;
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.SQLException;
import org.postgresql.Driver;

/**
 * Sample Sun SPOT host application
 */
public class SunSpotHostApplication {

private JTextArea status;
long ourAddr;


public void run() throws Exception {

          long ourAddr =
RadioFactory.getRadioPolicyManager().getIEEEAddress();
         System.out.println("Our radio address = " +
IEEEAddress.toDottedHex(ourAddr));
```

```
         //drawgui();
        /*  insert the sensor data from sunspot class     */
        insertspot();
        //startxserve();

}

public void drawgui () {

 //creating GUI
             JFrame.setDefaultLookAndFeelDecorated(true);
        JFrame fr= new JFrame("Communication Spot Free Range Sensor ");
        status = new JTextArea();
        //stop = new JButton("STOP");
        JScrollPane sp = new JScrollPane(status);
        fr.add(sp);

        fr.setSize(360, 200);
        fr.validate();
        fr.setVisible(true);
        //fr.add(stop);
}

public void insertspot() throws Exception {


         /* Broadcast port on which sensor connect */
RadiogramConnection conn=(RadiogramConnection)
Connector.open("radiogram://:99");
Datagram dg = conn.newDatagram(conn.getMaximumLength());
Datagram dgreply= conn.newDatagram(conn.getMaximumLength());

while (true)
{
try {

   Class.forName("org.postgresql.Driver");
   conn.receive(dg);
    String RawData = dg.readUTF();


/*Print    */
System.out.println(RawData);

/*
status.append("\n"+"Sesnor
BaseStation:"+"\n"+IEEEAddress.toDottedHex(ourAddr)+"\n"
        +"Remote Sensor Reading"+"\n"+RawData);
 * */

//changing the spot hardware address to store in database
String nodea = "0014.4F01.0000.6D45";
String nodeb = "0014.4F01.0000.6A40";


/* splitted String RawData */
String readata = RawData;
```

```
String[] temp1;
String delimeter = ";";
temp1 = readata.split(delimeter);

for(int i =0; i < temp1.length ; i++)

System.out.println(temp1[i]);

 // System.out.println(temp1[5]+"VALUE IS THIS"); //debugging

/* DataBase Connection and  Insert  */
    System.out.println("Inserting values to Database");
    Connection c= null;
    c =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/task",
                                    "postgres", "b0331969");
    Statement select = c.createStatement();

    int update = select.executeUpdate("INSERT INTO spot_result" +
          "(nodeid,xaxis,yaxis,zaxis,temp,light)" +
                        " VALUES(\'" + temp1[0] + "\',\'" + temp1[1] +
"\',\'" +
                        temp1[2] +"\',\'" +temp1[3]+ "\',\'" +temp1[4]+
"\'," +temp1[5]+ ")");

  // System.out.println("Querying insert Error"); //debugging

/* Reset the stream */
dgreply.reset();
dgreply.setAddress(dg);
dgreply.writeUTF("Hello from Base");

conn.send(dgreply);

} catch (IOException e) {
System.out.println ("No route to " + dgreply.getAddress());
}

finally {

  org.sunspotworld.spotdata testa = new org.sunspotworld.spotdata();
   org.sunspotworld.crossbowdata testb = new
org.sunspotworld.crossbowdata();
        testa.run();
        testb.run();

  } //finally

}    }

//Class to run the Crossbow Mote Xserve Application
//Or either run that application manually from cygwin
/**
public static void startxserve() throws Exception{

    String path = ("C://Crossbow/cygwin/cygwin.bat");
    try {
```

```
        Runtime rt = Runtime.getRuntime();
        Process pr = rt.exec(path);

    }
    catch (Exception e){
        System.out.println(e.toString());
         e.printStackTrace();}
} **/



    /**
     * Start up the host application.
     *
     * @param args any command line arguments
     */
    public static void main(String[] args) throws IOException,
ClassNotFoundException, SQLException, Exception {
        SunSpotHostApplication app = new SunSpotHostApplication();
                app.run();


    }
}


//END//
```

// CLASS spotdata to generate XML result from database.

```
package org.sunspotworld;

import com.sun.spot.peripheral.radio.RadioFactory;
import com.sun.spot.peripheral.radio.*;
import com.sun.spot.io.j2me.radiostream.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.IEEEAddress;
import com.sun.spot.peripheral.TimeoutException;
import com.sun.spot.util.IEEEAddress;
import javax.microedition.io.*;

import com.sun.spot.*;

import java.io.*; import java.lang.*; import java.net.*;
import java.util.*;
import java.sql.*;
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.SQLException;
import org.postgresql.Driver;


public class spotdata {

public void run() throws Exception{
try {
    Class.forName("org.postgresql.Driver");
    System.out.println("Generating SunSpot XML-DATA of Sensors...");

    Connection c= null;


    c =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/task",
                                    "postgres", "b0331969");
    Statement select = c.createStatement();

   String sqlxmlquery = "select xmlelement(name Sensor,xmlelement(name
NodeId, c.nodeid),"+
"xmlelement(name Xaxis, c.xaxis),xmlelement(name Yaxis, c.yaxis),"+
"xmlelement(name Zaxis, c.zaxis),xmlelement(name Temp, c.temp),"+
"xmlelement(name Light, c.light),"+
"xmlelement(name Time, c.modtime)) from spot_result c where
nodeid='0014.4F01.0000.6A40';";

    ResultSet result1 = select.executeQuery(sqlxmlquery);

 //Debugging  System.out.println(result1);
    while (result1.next()) {
    String outputsunspot= result1.getString(1);
    System.out.println(outputsunspot);

    File spot1 = new File("C:/Documents and
Settings/Admin/Desktop/webdump/xml_sensor_result/spotresult1.xml");
```

```
    Writer writer = new BufferedWriter(new FileWriter(spot1));
    writer.write(outputsunspot);
 writer.close();

       } //while

    String spotquery2 = "select xmlelement(name Sensor,xmlelement(name
NodeId, c.nodeid),"+
"xmlelement(name Xaxis, c.xaxis),xmlelement(name Yaxis, c.yaxis),"+
"xmlelement(name Zaxis, c.zaxis),xmlelement(name Temp, c.temp),"+
"xmlelement(name Light, c.light),"+
"xmlelement(name Time, c.modtime)) from spot_result c where
nodeid='0014.4F01.0000.6D45';";

    ResultSet result2 = select.executeQuery(spotquery2);

 //Debugging  System.out.println(result1);
    while (result2.next()) {
    String outputsunspot2= result2.getString(1);
    System.out.println(outputsunspot2);

    File spot2 = new File("C:/Documents and
Settings/Admin/Desktop/webdump/xml_sensor_result/spotresult2.xml");
    Writer writer = new BufferedWriter(new FileWriter(spot2));
    writer.write(outputsunspot2);
 writer.close();

       } //while



}
finally {

} //finally
}

}
```

//CLASS crossbowdata to generate XML data from database.

```
package org.sunspotworld;



import java.io.*;
import java.io.*;
import java.util.*;
import javax.swing.*;
import java.lang.*;
import java.lang.String;
import java.net.*;
import java.sql.*;
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.SQLException;
import org.postgresql.Driver;

/**
 *
 * @author Mubeen
 */
public class crossbowdata {

    public void run() throws Exception{
try {
    Class.forName("org.postgresql.Driver");
    System.out.println("Generating CrossBowXML-DATA of Sensors...");
    Connection c= null;
    c =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/task",
                                    "postgres", "b0331969");
    Statement select = c.createStatement();

   String crossbowquery1 = "select xmlelement(name
Sensor,xmlelement(name NodeId, c.nodeid),"+
"xmlelement(name Xaxis, c.accel_x),xmlelement(name Yaxis, c.accel_y),"+
"xmlelement(name Humid, c.humid),"+
"xmlelement(name Temp, c.humtemp),"+
"xmlelement(name Time, c.result_time)) from mts400_results c where
nodeid='1';";

    ResultSet result1 = select.executeQuery(crossbowquery1);

 //Debugging  System.out.println(result1);
    while (result1.next()) {
    String outputcross1= result1.getString(1);
    System.out.println(outputcross1);

    File file1 = new File("C:/Documents and
Settings/Admin/Desktop/webdump/xml_sensor_result/crossbow_result1.xml")
;
    Writer writer = new BufferedWriter(new FileWriter(file1));
    writer.write(outputcross1);
```

```
writer.close();
       } //while


     String crossbowquery2 = "select xmlelement(name
Sensor,xmlelement(name NodeId, c.nodeid),"+
"xmlelement(name Xaxis, c.accel_x),xmlelement(name Yaxis, c.accel_y),"+
"xmlelement(name Humid, c.humid),"+
"xmlelement(name Temp, c.humtemp),"+
"xmlelement(name Time, c.result_time)) from mts400_results c where
nodeid='2';";

 ResultSet result2 = select.executeQuery(crossbowquery2);
 while (result2.next()) {
    String outputcross2= result2.getString(1);
    System.out.println(outputcross2);

    File file2 = new File("C:/Documents and
Settings/Admin/Desktop/webdump/xml_sensor_result/crossbow_result2.xml")
;
    Writer writer = new BufferedWriter(new FileWriter(file2));
    writer.write(outputcross2);
writer.close();

       } //while

}
finally {

} //finally
}



}
```

**APPENDIX. 2**

Below code shows the web service developed in NuSOAP by using PHP.

```
<?php
//call library
require ("C:/php/nusoap/lib/nusoap.php");

// Instantiate a new soap server object
$server = new soap_server();


//Initiate WSDL Configuration
$server->configureWSDL('Sensorinfo','http://localhost/webservices');

//Desigate the WSDL namespace
$server->wsdl->schemaTargerNamespace =  'urn:Sensorinfo';   //
'http://localhost/Webservices'


// Function: getNodedata()
// Inputs: None
// Outputs: A string containing information about a nodeid,
// its temperature, and date.

function getNodedata($parms) {

$username = "postgres";
$password = "b0331969";
$database = "task";

// Connect to the PGSQL server
$pg = pg_connect("host=localhost user=$username password=$password
dbname=$database");

if ($parms=='b')
{



// Create and execute the query
$query = "SELECT * FROM spot_result WHERE nodeid='0014.4F01.0000.6A40'
ORDER BY modtime DESC LIMIT 1";


$result = pg_query($query);

if (!$result) {
return new soap_fault('Server', '', 'Internal server error.');
}


$row = pg_fetch_array($result);
```

```
// Retrieve, assemble, and return the  data
$nodeid = $row["nodeid"];
$temp = $row["temp"];
$time = $row["modtime"];
$light = $row["light"];
$xaxis = $row["xaxis"];
$yaxis = $row["yaxis"];
$zaxis = $row["zaxis"];

return "$temp, $nodeid, $time, $light, $xaxis,$yaxis,$zaxis";

}

if ($parms=='b1')
{



// Create and execute the query
$query = "SELECT * FROM spot_result WHERE nodeid='0014.4F01.0000.6D45'
ORDER BY modtime DESC LIMIT 1";


$result = pg_query($query);

if (!$result) {
return new soap_fault('Server', '', 'Internal server error.');
}


$row = pg_fetch_array($result);


// Retrieve, assemble, and return the  data
$nodeid = $row["nodeid"];
$temp = $row["temp"];
$time = $row["modtime"];
$light = $row["light"];
$xaxis = $row["xaxis"];
$yaxis = $row["yaxis"];
$zaxis = $row["zaxis"];

return "$temp, $nodeid, $time, $light, $xaxis,$yaxis,$zaxis";

}


if ($parms=='a')
{

// Create and execute the query
$query = "SELECT * FROM mts400_results WHERE nodeid='1' ORDER BY
result_time DESC LIMIT 1";
```

```
$result = pg_query($query);

if (!$result) {
return new soap_fault('Server', '', 'Internal server error.');
}


$row = pg_fetch_array($result);


// Retrieve, assemble, and return the  data
$nodeid = $row["nodeid"];
$humtemp = $row["humtemp"];
$light = $row["taosch0"];
$humid = $row["humid"];
$xaxis = $row["accel_x"];
$yaxis = $row["accel_y"];
$time = $row["result_time"];

return "$humtemp, $nodeid, $time, $light,$xaxis,$yaxis,$humid";

}

if ($parms=='a1')
{



// Create and execute the query
$query = "SELECT * FROM mts400_results WHERE nodeid='2' ORDER BY
result_time DESC LIMIT 1";


$result = pg_query($query);

if (!$result) {
return new soap_fault('Server', '', 'Internal server error.');
}


$row = pg_fetch_array($result);


// Retrieve, assemble, and return the  data
$nodeid = $row["nodeid"];
$humtemp = $row["humtemp"];
$light = $row["taosch0"];
$humid = $row["humid"];
$xaxis = $row["accel_x"];
$yaxis = $row["accel_y"];
$time = $row["result_time"];

return "$humtemp, $nodeid, $time, $light,$xaxis,$yaxis,$humid";

}
```

```
}

// Register the getNodeData() method
$server->register("getNodedata",
array('input' => 'xsd:string'),
array('return' => 'xsd:string'),
'urn:Sensorinfo',
'urn:Sensorinfo#getNodedata'
//,'rpc',
//'encoded','Return data to mobile client'
);

// Automatically execute any incoming request
$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ? $HTTP_RAW_POST_DATA:
'';

$server->service($HTTP_RAW_POST_DATA);


?>
```