Design guidelines for building a wireless sensor network for environmental monitoring

Nikos Giannopoulos[2], Christos Goumopoulos[1,2], Achilles Kameas[1,2]
[1]Research Academic Computer Technology Institute,
N. Kazantzaki, 26500 Rio Patras, Hellas
[2]Hellenic Open University, 16, Sahtouri Str, Patras, Hellas
ngianop@gmail.com, goumop@cti.gr, kameas@eap.gr

Abstract

Environmental monitoring is a critical process that demands accuracy, reliability and stability at the operation level. Monitoring variables such as temperature, humidity, barometric pressure, soil moisture and ambient light facilitates research in fields such as precision agriculture, habitat monitoring, weather monitoring etc. The use of wireless sensor networks (WSNs) provides a technology solution for dynamic and unattended environmental monitoring, under the condition that requirements such as efficient power management and system robustness are satisfied. This paper presents the design and implementation of a WSN for monitoring environmental variables and evaluates its effectiveness. Based on the acquired experience we describe how we have confronted certain problems such as network synchronization and data consistency and we provide certain design guidelines for building such a system.

Keywords: Wireless Sensor Networks, Environmental Monitoring, Software Engineering.

1. Introduction

Recent advances in the technology of electronic circuits gave the opportunity for minimizing the size and reducing the cost of circuits' productions. This rapid development led to the implementation of autonomous compact nodes (motes) that are capable to run complicate operations consuming very little energy using plain batteries. These nodes have approximately the size of a box of matches. Such nodes communicate wirelessly and use sensors that are capable to measure physical variables such as temperature, moisture, light level etc. The most important thing is that they do not need the human presence in order to operate. This gives the advantage of using them in remote places that may be also hazardous for the human life as for example in volcanoes.

These nodes consist of a wireless communication unit, a microprocessor, a data acquisition unit and a memory unit. The existence of both microprocessor and memory unit give the ability the nodes to be programmed in order to perform specific measurements taken either at fixed time intervals [1] or based on an event driven model [2]. Also they can be programmed in such ways in order to follow specific routing protocols [3].

WSNs allow the coverage of wide geographical areas. The range of the area depends on several factors such as the number of nodes, the way that have been placed and the range of the wireless units. Researchers have proposed placements in a structure aiming for power efficiency and data reliability [4].

This paper presents the design and implementation of a WSN for monitoring environmental variables and evaluates its effectiveness using laboratory tests. In order to develop the monitoring applications we used on the hardware side the Mica2 motes by Crossbow [5], embedded and external sensors; on the software side we have used TinyOS [6], an open source operating system developed by the University of Berkeley and NesC [7], a component-based and event-driven programming language.

The main contribution of this paper is to provide a number of design guidelines for implementing a WSN for environmental monitoring. We also discuss how certain parameters have been selected for maximizing network reliability and lifetime and how certain issues have been confronted such as network synchronization and data consistency.

In Section 2 we present related work. Section 3 gives an overall description of the system developed and the tools that were used. In Section 4 we outline the constraints and the design goals that were established as well as the solutions that were provided.

978-0-7695-3788-7/09 $25.00 © 2009 IEEE
DOI 10.1109/PCI.2009.17

In the next section we discuss the lessons learnt from this effort and finally we give the conclusions.

2. Related Work

A large number of projects regarding environmental monitoring is running or have successfully completed in all over the world [8]. The encouraging results give the necessary credits for further improvements and development of the specific technology. One such effort applied WSN technology to monitor the extremely dangerous and hostile environment of the Tungarahua volcano [9]. Scientists placed a WSN into the volcano to monitor volcanic eruptions with low-frequency acoustic sensors and data were received for 54 hours.

The LOFAR [10] and PLANTS [11] projects applied WSNs in the precision agriculture domain. The aim of both projects is the monitoring of microclimate in agriculture. Both projects are based on custom micro-controller nodes that are similar to Mica2 motes equipped with special-purpose sensors. The idea is that a number of nodes are placed in a field. The network can use the MintRoute routing protocol that is available with TinyOS. These motes transmit data to each other which ultimately are gathered at a collection-point. In the case of PLANTS the data collected are analyzed in order to take proactive actions such as irrigation or fertilization by activating the corresponding actuators.

Finally, a WSN was placed in Pinjar north of Perth in Western Australia to monitor spatial variations in surface soil moisture over time [2]. The network consists of Mica2 motes on MDA platforms. The collected data are sent via a GSM gateway to a database viewable from an internet web page using the SOAP protocol and web services.

While most related work focus on certain characteristics of WSN development such as communication protocols, power management etc., the contribution of this paper is to present a methodological approach on developing such systems. We report our experience on tackling critical issues that may arise during the development process and provide certain guidelines to handle them.

3. System Description

In this section we shall briefly describe the WSN that we developed for environmental monitoring. Our WSN will be able to perform measurements for temperature, humidity, light level and soil moisture. The data will be gathered by the sensors and will be transmitted to the base station (one-hop network). The base station through appropriate interface will log the data in a database. Through a web based application the user will be able to monitor the collected data and request graphs on demand. Finally, the user will be able to change certain variables of the network mainly for debugging purposes. The architecture of the system is illustrated in a high-level view in Figure1.

In addition, the monitoring application logic can incorporate rules that change the sampling rate proactively. For instance, a rule can be defined that says when the difference of the temperature within an hour is above a certain limit then increase the sampling rate from hours to minutes (e.g., because a rainfall has been detected).

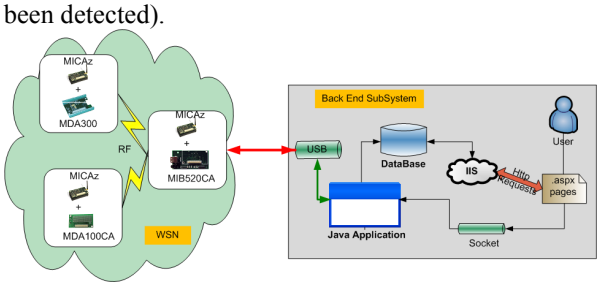

Figure 1 – High-level system architecture

3.1. Hardware Tools

To implement our WSN we used the following hardware [5]:

- Three MPR2400 MICAz modules.
- The MIB520CA base station module.
- One MDA100CA data acquisition board. It provides a precision thermistor, a light sensor/photocell and general prototyping area.
- One MDA300 data acquisition board which includes an onboard temperature and humidity sensor.

Finally, we used the moisture probe Echo-10 by Decagon [12], which was plugged on MDA300 acquisition board as we shall explain later.

3.2. Software Tools

For the needs of our project we used a variety of programming environments. For the implementation of the applications which run on the motes we used the nesC programming language and the MoteWorks environment [13]. We used Java to implement the application for the communication between the MIB520CA and the database. For the web based application we used the Microsoft Visual Studio .Net 2003 and the .aspx technology. For the graphs we used the Dundas Chart for asp .net 2003. Finally, we used

the Microsoft SQL Server 2000 to develop the database in which the data will be logged.

4. Design and Implementation

The nature of the hardware of WSNs imposes many constraints that must be considered when establishing the design goals and trade-offs of the applications. These constraints are mainly attributed to the limited resources of the motes: processing power, memory, communication bandwidth/range and power supply. Therefore, developers need to take into account energy requirements during the design phase.

Regarding the engineering approach we followed, given that no prior experience existed, we had to be ready to confront several new challenges and to overcome many difficulties. For that reason a risk management analysis had to be done before starting the implementation. During that phase, we identified the potential risks that would jeopardize the project. The risks were classified into the following categories: sensitive equipment usage, integration of heterogeneous systems and technologies, open source, insufficient tool documentation, limited number of nodes, measurement accuracy and network reliability. After creating the list of the risks, a risk analysis was performed evaluating the issues depending on the severity and the impact of each of them on the project. A major issue that we had to confront related to the combination of TinyOS with Crossbow software. Although the use of open source software has many advantages like no cost and customizability, it may also come with a few holes. The most important in our case were related with the inadequate documentation and the existence of not well tested code.

As a consequence of the constraints discussed above we have followed an incremental development model with risk analysis and assessment which can be seen as a light spiral model. The basic functional requirements of the project were specified in the previous section. Furthermore, two critical non functional requirements specified are data reliability and power efficiency.

4.1. Network Synchronization

Network synchronization is achieved by exchanging a sequence of messages when the network starts to operate. Upon starting a node sets a boolean variable *isSynch* to FALSE and sends a synchronization request message to the base station (BS). A FALSE value means that the node has not been synchronized.

After sending the request the node turns to stand-by mode. Upon receiving the synchronization request from all nodes the BS sends a synchronization signal which contains also the sampling rate. After receiving the synchronization signal, each node sets *isSynch* to TRUE and starts the timer that controls the sampling rate. When the user or the application logic changes the sampling rate in the network, the BS broadcasts a STOP message to all nodes. When the node receives the STOP message, immediately stops its timer and sends back to the BS an ACKN message that has stopped. Upon receiving the acknowledgment from all nodes the BS broadcasts a new synchronization message to the network. Each node that will receives the SYNCH message starts the timer with the new time interval and sends back to the base station an ACKN message. If the BS will not receive the ACKN from all the nodes, reports the error and repeats the procedure.

4.2. Data Consistency

Data reliability is a critical issue for WSNs. We have to ensure that the system will not lose packets. One way to achieve that is by numbering the packets with the data that each node sends. Each packet takes a number before being transmitted back to BS. After the packet's transmission, it is stored in the memory of the node. The basic message structure of a TinyOS packet is 36 bytes by default. This includes 7 bytes of generic Active Message fields and a maximum of 29 bytes for the payload. The payload is determined by the application. Part of the payload structure of the message is shown in Figure 2 (we omit the standard fields of TinyOS *message_t* structure).

1 byte	2 bytes	2bytes	2bytes	2bytes	2 Bytes
packID	temp	light	humidity	Echo10	batterry

Figure 2 – Part of the message structure

The important point is that the BS keeps the last packet id of each node. If the BS receives a packet from a node having packet id greater than the one from the last received packet, infers that it has lost a packet. Then the BS sends a request to the specific node, asking for the lost packet(s) by sending the specific id(s). To avoid network congestion the BS checks periodically (e.g., a few times a day depending on the sampling rate) that all packets have been received and broadcasts an acknowledgement signal so that the nodes can reuse their local memory.

An additional measure we have used to handle memory shortage is the node to explicitly ask for reconciliation with the BS. If the BS returns a message that doesn't need reconciliation the node will clear the

buffer. The reconciliation can be achieved by sending to the BS the last package id so that the BS can perform data consistency verification.

4.3. Data Aggregation

Data aggregation is a solution that helps the network to consume less energy considering that the packet transmission is a highly energy consuming operation. Aggregation requires to separate the transmission rate from the sampling rate (the former is larger than the latter). Each time a measurement is completed the result is stored in node's memory. When a transmission timer elapses, the node reads from memory the collected data, performs the aggregation and sends an aggregated packet to the BS.

We have tested the Delta compression algorithm as part of the aggregation process. The specific algorithm is used for files' compression and is based on the practice of sending characters and their frequency. According to the Delta algorithm and supposing we have five values to send to the BS (e.g., 10, 20, 10, 15, 20), the packet that will be sent will be the following 10x2 20x2 15x1 instead of 5 simple messages. Another approach of using Delta is not to send the frequency of values measured but the change in the value from some expected value. In many cases, with well chosen expected values, these changes tend to be small relative to the range of possible values and lead to a high frequency of values within a small range.

A consequence of using Delta is that we need to send extra information for each type of measurement. The problem that arises in this situation is that we lose the time that each value was taken. A solution to that problem is to use the time the packet arrived at the BS and the sampling rate. By knowing these two parameters we can easily calculate the timestamp of each measurement.

4.4. Power Management

From the datasheets of Mica2 motes [5] the energy consumption for each node operation can be taken. The transmission of a packet requires about five times more energy than reading a sensor and storing the value locally. Thus, minimizing the number of transmissions required, as discussed in the previous section, is a critical step for extending the lifetime of the WSN. A simple technique in order to reduce the level of power consumption is to turn-on the radio of a node only when it is needed. However, a node not only sends measurement packets but as well may receive packets such as a new sampling rate by the BS or in the case of a multi-hop network, packets that should be forwarded to the next level. In the latter case a scheme that alternates the node state through sleep/wake cycles is more appropriate in order to preserve energy. S-MAC [14] is a MAC protocol, with available source code, designed to address the issue of energy efficiency and coordinated sleeping and therefore is well suited for supporting applications with sleep/wake cycles.

5. Discussion

The testing of our system took place in the lab. We have tested that the embedded and the external sensors can make accurate measurements, the motes can store measured values, the transmission of values and associated ID data trough the wireless link and the transmission of data from the BS to the database. For the measurements of the soil moisture we used a plant. The WSN was able to operate without interruption for 15 days with a sampling rate of 3 seconds. Figure 3 shows soil moisture variation and reaction to watering in the period of the 7 first days.

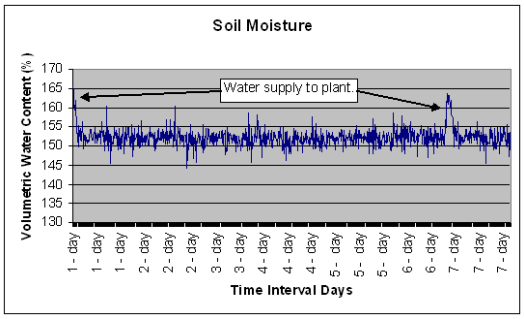

Figure 3 – Soil moisture measurements

From the lab tests, there is sufficient evidence that the system performs the basic functionality specified. The experiments serve as a feasibility study of our prototype and the design goals made. Using the TOSSIM simulator we tested our network in order to ensure that is functional for more than two nodes.

During this endeavor several lessons have been learnt as several problems had to be encountered. We discuss first the issue of the development process. Using an incremental development model has proved to be an effective way to handle the specification of the requirements and the gradual design of the system. With many requirements unidentified and lack of experience in the hardware and software tools in the beginning, the prioritization of requirements and the tackling of the high priority tasks provided a feasible path towards implementing the WSN.

Data aggregation and compression is vital due to the high cost of transmission and the limited energy constraints of motes. Whether data should be

aggregated at nodes or only significant changes in data in respect to expected values should be transmitted depends on the nature of the application and whether it is feasible to define successful thresholds.

A constraint of our experimental process was the limited number of available motes. The network topology in our case was a simple V. A layered architecture is important in large WSNs due to limited energy and computational power of motes. The lower layer includes the motes which are less powerful and are used to take measurements, whereas the upper layer includes more powerful motes which perform data aggregation and fusion on the values taken from lower layer nodes. The use of TOSSIM simulator can verify the design decisions and the algorithms used in such large-scale networks.

Finally, we report that we had to overcome a few obstacles with respect to the open source software that we have used for the development process. There was an unexpected and hard to spot incompatibility between different versions of the operating system regarding the Active Message structure. Finally, the undocumented code that is provided with the MoteWorks environment, led us to spend many hours of experimental programming. Conclusively, the use of open source software is a critical issue that requires gathering of as much information as possible about the hardware and software tools to be used. Backward compatibility is not always guaranteed.

In the following table we summarize our experience from building a WSN for environmental monitoring in the form of guidelines.

TABLE 1. DESIGN GUIDELINES FOR BUILDING A WSN FOR ENVIRONMENTAL MONITORING

Topic	Guidelines
Development Model	Incremental model. Perform risk analysis. Process can take the form of a light spiral model.
Implementation	Keep code size small. Do not use complicated and time consuming procedures. Implement reusable code modules.
Power Management	Using aggregation and compression techniques minimizes the number of transmitted messages. Multi-layered architecture for large-scale WSN.
Data Integrity	Implement a protocol that prevents data loss.
Testing	Testing is difficult due to non-determinism. Validate accuracy of sensors. External sensors need calibration. Radio module is the most sensitive factor. Use simulator (e.g., TOSSIM) in order to verify protocol designs.
Open Source	Be ready to confront issues regarding hardware and software compatibility. Study carefully the existent documentation and related work.
Hardware Safety	The nodes should be housed tightly in water-proof packaging (e.g., IP-67 rated) to withstand harsh conditions.

6. Conclusion

We presented the design and implementation of a WSN for monitoring environmental variables and evaluated its effectiveness. Based on the acquired experience we described how we have confronted certain problems and we provided certain design guidelines for building such a system.

Future work will focus on addressing the limitations of the current prototype and building a larger network with additional sensors such as a photosynthetic solar radiation meter. We plan also to deploy the network to an open field for agricultural monitoring.

9. Reference

[1] Beckwith, R., Teibel, D., and Bowen, P., Report from the Field: Results from an Agricultural Wireless Sensor Network, In 29th IEEE International Conference on Local Computer Networks (LCN'04), pp. 471–478, 2004.

[2] Cardell-Oliver, R., Kranz, M., Smettem, K., and Mayer, K., A Reactive Soil Moisture Sensor Network: Design and Field Evaluation. International Journal of Distributed Sensor Networks, 1:149-162, 2005.

[3] Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H., Energy-Efficient Communication Protocol for Wireless Microsensor Networks, In 33rd Conference on System Sciences, IEEE, CS, pp. 8020, 2000.

[4] Ganesan, D., Cristescu, R., Beferull-Lozano, B., Power-efficient sensor placement and transmission structure for data gathering under distortion constraints, In IPSN'04, ACM Press, pp. 142–150, 2004.

[5] Crossbow Technology Inc., Mica2 Motes Specifications, http://www.xbox.com/

[6] TinyOS, http://www.tinyos.net.

[7] Gay, D., et al, The nesC language: A holistic approach to networked embedded systems, In ACM SIGPLAN on PLDI, ACM, pp. 1-11, 2003.

[8] Hakala, I., Tikkakoski, M., Kivela, I., Wireless Sensor Network in Environmental Monitoring-Case Foxhouse, 2nd Inter. Conf. on Sensor Technologies and Applications, pp. 202-208, 2008.

[9] Werner-Allen, G., Johnson, J., Ruiz, M., Lees, J., Welsh, M., Monitoring volcanic eruptions with a wireless sensor network, In 2nd European Workshop on WSNs, pp. 108-120, 2005.

[10] Baggio, A., Wireless Sensor Networks in Precision Agriculture, In ACM Workshop Real-World WSNs, 2005.

[11] Goumopoulos, C., Kameas, A., and Oflynn B., Proactive Agriculture: An Integrated Framework for Developing Distributed Hybrid Systems, In Ubiquitous Intelligence and Computing (UIC-07), Springer-Verlag, pp. 214-224, 2007.

[12] Decagon, http://www.decagon.com/echo/

[13] Crossbow Technology Inc., MoteWorks Brochure, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MoteWorks_OEM_Edition.pdf

[14] Ye, W., Heidemann, J., Estrin D., Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks, IEEE/ACM Trans. on Netw., 12:493–506, 2004.