International Journal of Computer and Communication Technology

Volume 2 | Issue 2 Article 3

April 2011

Wireless Sensor Networks for Indoor Environment Monitoring Using LabVIEW

Mrutyunjay Rout
Electrical Engineering Department IIT Roorkee Roorkee, India, mrutyunjay.rout@gmail.com

Dr. Harish Kumar Verma
IIT Roorkee, hkvfee@iitr.ernet.in

Subhashree Das
Electronics & Comm. Engg. Dept. NIT, Rourkela Rourkela, India, subhashreedas18@gmail.com

Recommended Citation
Rout, Mrutyunjay; Verma, Dr. Harish Kumar; and Das, Subhashree (2011) "Wireless Sensor Networks for Indoor Environment Monitoring Using LabVIEW," *International Journal of Computer and Communication Technology*: Vol. 2 : Iss. 2 , Article 3.
DOI: 10.47893/IJCCT.2011.1076
Available at: https://www.interscience.in/ijcct/vol2/iss2/3

Wireless Sensor Networks for Indoor Environment Monitoring Using LabVIEW

Mrutyunjay Rout
Electrical Engineering Department
IIT Roorkee
Roorkee, India
mrutyunjay.rout@gmail.com

H.K. Verma
Electrical Engineering Department
IIT Roorkee
Roorkee, India
hkvfee@iitr.ernet.in

Subhashree Das
Electronics & Comm. Engg. Dept.
NIT, Rourkela
Rourkela, India
subhashreedas18@gmail.com

Abstract*— Wireless sensor networks (WSNs) have gained worldwide attention in recent years, particularly with the rapid progress in Micro-Electro-Mechanical Systems (MEMS) technology which has facilitated the development of smart sensors. These sensors are small, with limited processing and computing resources, and they are inexpensive compared to traditional sensors. These sensor nodes can sense, measure, and gather information from the environment and, based on some local decision process, they can transmit the sensed data to the user. WSNs are large networks made of a numerous number of sensor nodes with sensing, computation, and wireless communication capabilities. In present work we provide a brief summary of the state-of-the-art in wireless sensor networks, investigate the feasibility of indoor environment monitoring using crossbow wireless sensor nodes. Here we used nesC programming language and TinyOS operating system for programming Crossbow sensor nodes and LabVIEW GUI is used for displaying different indoor environmental parameters such as temperature, humidity and light acquired from different Wireless sensor nodes. These sensor readings can help building administrators to monitor the physical conditions of the environment in a building for creating optimized energy usage.

Keywords*— Wireless Sensor Network (WSN), Wireless Sensor Node, TinyOS, LabVIEW.

I. INTRODUCTION

Wireless sensor networks (WSNs) have recently come into prominence because of their ad-hoc and easy deployment than that of the traditional wired sensing systems. Wireless sensor networks are usually composed of hundreds or thousands of inexpensive, low-powered sensing devices with limited memory, computational, and communication resources. These networks offer potentially low-cost solutions to many segments of human life and economy, from environmental monitoring and conservation, to manufacturing and business asset management, to automation in the transportation and health care [1].

WSNs can also be deployed the measurement of environment parameters which are dangerous or difficult to access. For example applications such as sensing a building integrity or structural vibrations during an earthquake, the stress of an airplane's wings, forest fire detection etc [2].

A WSN is a special kind of an ad-hoc network composed by hundreds, even thousands, of wireless sensor nodes. Each wireless sensor node is equipped with a microprocessor for data processing, radio chip for wireless communication and sensor board for sensing some physical phenomena, such as pressure, temperature, light, humidity, acceleration, etc. Depending on the specific application, sensor nodes are deployed into some sensing field so that they can collaborate with each other and form a wireless network and this sensor network is connected to the outside network through a base station.

Generally a WSN consists of a base station (or "gateway") that can communicate with a number of sensor nodes via a radio network as shown in figure 1. Data is collected at the wireless sensor nodes or motes (comprising sensors, data acquisition circuit, processor and RF transceiver), compressed, and transmitted to the gateway directly or, if required, uses other wireless sensor nodes to forward data to the gateway. The transmitted data is then presented to the system by the gateway connection [3].

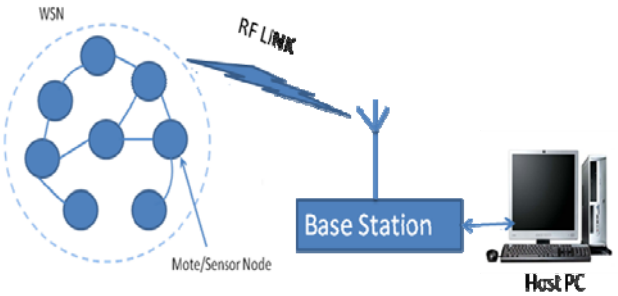

Figure 1: Basic WSN Architecture

The present paper gives the idea about the monitoring of different indoor environment parameters such as temperature, relative humidity, and ambient light by using crossbow wireless sensor nodes. This present work also gives the warning indications of different indoor

environment parameters which can help to improve the environmental settings for human comfort levels. These sensor readings can also help building administrators to monitor the physical conditions of the environment in a building for creating optimized energy usage.

II. LOW-POWER OPERATING SYSTEM – TINYOS

TinyOS is an event-driven operating system designed for sensor network nodes that have very limited resources, such as the mote platform. It was developed by UC Berkley, and is written in a component-based architecture called nesC (network embedded systems C). nesC is an extension to the C programming language designed specifically for wireless embedded sensor networks. Applications for the mote platform programmed in nesC are built out of components. Each component performs a specific small function. These components are "wired" together to form a larger application. nesC allows the programmer to deal with the radio and power management systems of the mote. TinyOS allows the programmer to control the power conditions of each mote. With the ability to put the mote to sleep and wake it up to take data readings the battery life of the mote can be prolonged. To that end mote can be programmed to take these sensor readings when the mote is in an awake mode [4].

TinyOS provides a proper networking stack for wireless communication that abstracts away the underlying problems and complexity of message transfer from the application developer. A major problem in wireless communication is message collisions due to the shared nature of the communication medium. When multiple nodes in vicinity start transmitting messages simultaneously, their transmissions overlap and corrupt each other. TinyOS provides a media access control (MAC) layer that arbitrates access to the channel using Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) technique and alleviates the changes of collisions significantly. MAC layer also detects bit-level corruptions in received messages using cyclic redundancy check (CRC). All of these are done by the networking component transparent to the application layer. The application developer does not have to know or worry about the details of the underlying message transfer process [4].

III. LabVIEW

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a graphical programming language developed by National Instruments [5]. It uses icons instead of lines of text to create applications. In contrast to text-based programming languages, where instructions determine the order of program execution, LabVIEW uses dataflow programming, where the flow of data through the nodes on the block diagram determines the execution order of the VIs and functions. The programming language used in LabVIEW, also referred to as G, is a dataflow programming. Execution is determined by the structure of a graphical block diagram on which the programmer connects different function-nodes by drawing wires. These wires propagate variables and any node can execute as soon as all its input data become available. Since this might be the case for multiple nodes simultaneously, G is inherently capable of parallel execution. Multi-processing and multi-threading hardware is automatically exploited by the built-in scheduler, which multiplexes multiple OS threads over the nodes ready for execution. The VIs, or virtual instruments, is LabVIEW programs that imitate physical instruments.

A VI consists of a front panel, block diagram, and an icon that represents the program. The front panel is used to display controls and indicators for the user, and the block diagram contains the code for the VI. The icon, which is a visual representation of the VI, has connectors for program inputs and outputs. For developing different applications in LabVIEW there are different palettes such as tools palette, controls palette, and functions palette are used.

LabVIEW is fully integrated for communication with hardware such as GPIB, RS-232, RS-485, and plug-in DAQ devices. LabVIEW also has built-in features for connecting application to the Web using the LabVIEW Web Server and software standards such as TCP/IP networking and ActiveX. Using LabVIEW, we can create test and measurement, data acquisition, instrument control, data logging, measurement analysis, and report generation applications. We also can create stand-alone executables and shared libraries, like DLLs, because LabVIEW is a true 32-bit compiler. The version used in this work is version 2009.

IV. HARDWARE PLATFORM

The most widely accepted hardware platform is the UC Berkeley mote platform. The mote platform uses an 8-bit microprocessor, such as ATMEGA 128, ATMEL 8535, or Motorola HCS08, with a 4MHz CPU. Typically the motes have 4kB RAM, which is used for holding run-time state of the program (values of the variables) 128 kB programmable Flash memory where the application program is downloaded (either via a programmer-board or wirelessly), and an additional flash memory storage space up to 512 kB.

The work represented in this paper required four Crossbow Sensor boards (MTS 400), five Mote Processor Radio boards (MPR2400) and a Mote Interface Board (MIB 520).

The MPR 2400 (MICAz) has been used as the primary workhorse [6]. The MICAz is latest generation of Motes developed by Crossbow Technology. The MPR2400 (2400 MHz to 2483.5 MHz band) uses the Chipcon CC2420, IEEE 802.15.4 compliant, ZigBee ready radio frequency

transceiver integrated with an Atmega128L micro-controller, 51-pin I/O connector, and serial flash memory. Due to fully compliance with the IEEE 802.15.4 standard, the MICAz is capable of establishing and maintaining a multi-hop mesh network. The MICAz radio processor MPR2400 is used for sensor-to-sensor and sensor-to-gateway communication.

The Crossbow USB-interface board MIB520 provides USB connectivity to the IRIS and MICA family of Motes for communication and in-system programming [6]. It supplies power to the devices through USB bus. The MIB520 is connected to the MICAz via the Hirose 51-pin connector. The MIB520CB offers two separate ports: one dedicated to in-system Mote programming and the other for data communication over USB. It has an on-board processor that programs Mote Processor Radio Boards. USB-power eliminates the need for an external power source.

The Crossbow MTS 400CA sensor boards utilize the latest generation of IC-based surface mount sensors [7]. These energy efficient digital devices in turn provide extended battery life and performance wherever low maintenance field-deployed sensor nodes are required. These boards offer five basic environmental sensing parameters such as temperature, humidity, barometric pressure, ambient light and accelerometer. These sensor boards are also having 2K EEPROM for user configuration data. The MTS400CA sensor boards are attached with Radio Processor (MPR2400) for sensing five environmental parameters such as temperature, humidity, barometric pressure, ambient light and acceleration and sending these parameters to the base station.

V. SYSTEM IMPLEMENTATION

Temperature, humidity, true-light sensors, infrared-based presence sensors (PIR sensor) and chemical sensors are useful for in-door environmental monitoring systems. The applications foreseen for indoor monitoring are controlling of heating, AC, or light intensity guided by the communication from these sensors.

The work represented in this paper, we have used crossbow sensor nodes (combination of Mote Radio Processor MPR2400 and sensor board MTS400CA) which are placed at four different rooms inside the building. These sensor nodes are sense environmental parameters such as temperature, ambient light and relative humidity from their surrounding and send these parameters to the base station (combination of Crossbow USB interface board MIB520 and Mote Radio Processor MPR2400). The base station connected to the host PC through which user can see all the environment parameters and the warning indications of different indoor environment parameters from LabVIEW GUI which can help to improve the environmental settings for human comfort levels. These sensor readings can also help building administrators to monitor the physical conditions of the environment in a building for creating optimized energy usage. A schematic of the hardware connections is shown in figure 2.

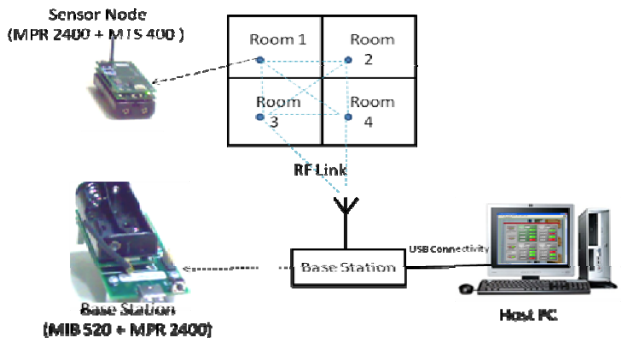

Figure 2: Schematic diagram of hardware connections

In this paper we have used Motework[8] platform for programming all the sensor nodes. All the programs are written in nesC language.

A. Compilation and Installation of the Code: To compile nesC application code from Programmer's Notepad: Select **Tools > make micaz.** The output section of the programmer's Notepad will print the compiling results to the screen. If there is no error then "writing TOS image" appears as the last line in the output window.

After compiling nesC application code successfully, the application code is installed to the mote by using following steps: (1) Plug the Mote (MPR2400) into the programming board MIB520, (2) Select **Tools > shell** and type *make micaz reinstall,<node Id> mib520,com3*. The node ID is selected between 1 and 65534 for the mote that function as the sensor nodes and node ID 0 for the mote that functions as the base station.

The steps required to compile and install the XMeshBase application onto the base station node are: (1) Select the ***XMeshBase.nc*** file in Programmer's Notepad, (2) Select **Tools > shell** and type *make micaz install,0 mib520,com3.*

After the completion of programming of all MTS400 sensor motes, the base station mote is plugged into the programming board and the sensor node motes are plugged into the MTS 400 sensor boards as shown in figure 2.

B. Results: The data acquired from all sensor nodes placed in four different rooms are displayed on host PC by running LabVIEW GUI.

The user can also acquire the data from the sensor nodes individually by operating the switches that are provided on the front panel of the LabVIEW GUI. A typical display is shown in figure 3.

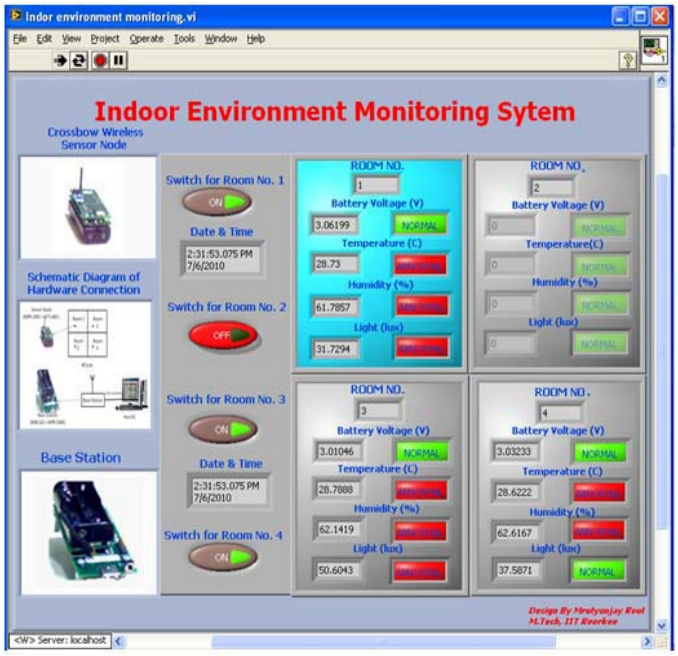

Figure 3: Experimental Results displayed on LabVIEW GUI

VI. CONCLUSION

Wireless Sensor Networks have proven themselves to be a reliable solution in providing remote sensing for indoor environmental monitoring systems. The wireless sensor nodes were adapted according to application-specific requirements to sense, collect and transmit relevant parameters. The information collected from the network was stored in a local database and made available for displaying in LabVIEW GUI displays.

The measurement of different environmental parameters such as temperature, relative humidity, and ambient light through Crossbow wireless sensor motes using MoteWork environment has been done and all the results has been shown on LabVIEW GUI. One of the major obstacles to the progress of the project was the understanding of tinyOS programming approach. When high-level programming languages become available to program motes the task will be simpler and the range of applications will broaden further.

REFERENCES

[1] Kazem Sohraby, Daniel Minoli, Taieb F. Znati, Wireless Sensor Networks: Technology, Protocol and Applications, Wiley-Interscience, 2007, PP. 17-64.

[2] Mohhamad Ilyas, Imad Mahgoub, Smart Dust: Sensor Network Applications, Architectures and Design,CRC Press, 2006, PP. 7-48.

[3] Jennifer Yick, Biswanath Mukherjee, Dipak Ghose, "Wireless Sensor Network Survey" Computer Networks: International Journal of Computer and Telecommunications Networking, vol. 52, no. 12, August 2008, PP. 2292-2330.

[4] Philip Levis, " TinyOS/nesC Programming Reference Manual", January 2006.

[5] "LabVIEW Basics I Course Manual" National Instrument, Document Number 320628G-01, September 2000.

[6] "MPR-MIB User's Manual." Crossbow Technology, Document Number 7430- 0021-08, Rev. A, June 2007.

[7] "MTS-MDA Sensor and Data Acquisition Boards User's Manual." Crossbow Technology, Document Number 7430-0020-05, Rev. A, June 2007.

[8] "MoteWork Getting Started Guide." Crossbow Technology, Document Number 7430-0102-05, Rev. D, March 2007.