TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.6, June 2014, pp. 4802 ~ 4807
DOI: 10.11591/telkomnika.v12i6.5525

Deployment of TinyOS for Online Water Sensing

Xin Wang*, Pan Xu
Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education),
Jiangnan University, Wuxi 214122, PR China
*Corresponding author, e-mail: wangxin@jiangnan.edu.cn

Abstract
Current quality assessment methods of water parameters are mainly laboratory based, require fresh supplies of chemicals, trained staff and are time consuming. Sensor networks are great alternatives for such requirements. We present a practical application of wireless networks: a remote water monitoring system running TinyOS. The contents of several chemicals in the water are sensed and transmitted. The sensor data are collected and transmitted via ZigBee and GPRS. Instead of focusing on theoretic issues such as routing algorithms, network lifetime and so on, we investigate special techniques involved in the implementation of the system while employing TinyOS and its special programming language.

Keywords*: TinyOS, hierarchical network, embedded operating system, water sensing*

1. Introduction

TinyOS is an open source, BSD-licensed operating system designed for low-power wireless devices, such as those used in sensor networks, ubiquitous computing, personal area networks, smart buildings, and smart meters. To confront the water pollution, various water monitoring systems based on cellular mobile network have been developed [1, 2]. These systems may assist environmental protection agencies in providing continuous water monitoring with minimum interaction of man interference. But, with such systems, the rare channel resources and hardware are greatly wasted when the monitoring nodes are distributed in higher density. The hierarchical organization [3], grouping of the monitoring nodes before transferring the sensor data to higher levels, is one of the mechanisms proposed to deal with that extravagance and is commonly referred to as clustering [4]. This paper is to show a hierarchical architecture, which is low in cost, easy to construct, less dependent upon network infrastructure, is implemented by employing ZigBee and GPRS devices, and especially with TinyOS as the sensor networks' operating system.

2. Research Method Structure and Composition of the System

TinyOS is designed to run on small, wireless sensors. Networks of these sensors have the potential to revolutionize a wide range of disciplines, fields, and technologies. Recent example uses of these devices include Golden Gate Bridge safety, volcanic monitoring and data center provisioning.

The hierarchical network is made up of one Base Station and some monitoring nodes, as is shown in Figure 1. These monitoring nodes are responsible for sampling the water, single-hopping the data to the base station by ZigBee channel, while the Base Station is set for coordination between the nodes, sending the sensor data which is collected from the nodes to the remote management platform.

The Base Station is composed of online monitoring device, GPRS (General packet radio service) DTU and ZigBee Module (worked as a coordinator, FFD), as is shown in Figure 2. The online monitoring device which uses CP1H PLC made by OMRON as the controller is responsible for testing the concentration of NO3-, PO43- and PH value. GPRS DTU implements the transmission of the remote signals by GPRS network.

GPRS is a packet oriented mobile data service on the 2G and 3G cellular communication system's global system for mobile communications (GSM) while ZigBee is new specification for a suite of high level communication protocols used to create personal area networks built from small, low-power digital radios.

Received December 30, 2013; Revised March 12, 2014; Accepted March 28, 2014

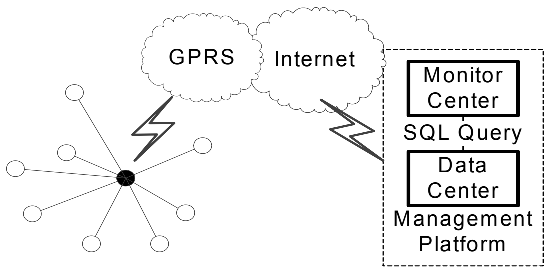

Figure 1. Topology of the Network and Remote Management Platform. The Disc Stands for Base Station, Circle for Monitoring Nodes

ZigBee module is made up of MSP430 controller and CC2420 RF chip. It is used to read the data of the water in PLC, collects the data of water in the other monitoring nodes and exchanges the data with the remote management platform. Different from the Base Station, the monitoring node, on the other hand, does not include GPRS DTU, and its ZigBee module is a Reduced Function Device (RFD).

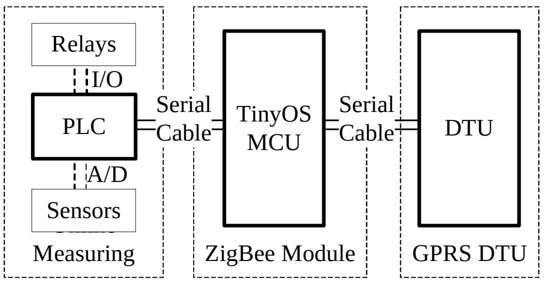

Figure 2. Components of the Hardware of the Base Station

3. Zigbee Module

Both FFD and RFD are built with TinyOS operating system. TinyOS is an embedded system for wireless network, with a set of components that are included as-needed in applications [5].

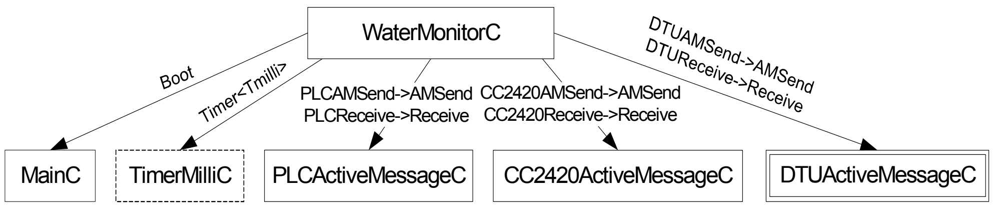

Figure 3. Components used for FFD and RFD. A solid box is for a singleton component, a dashed box for a generic component, while double-line box is only used in FFD. An arrow is an interface

At a high level, TinyOS provides three things to make writing systems and applications easier: (1) A component model, which defines how you write small, reusable pieces of code and compose them into larger abstractions; (2) A concurrent execution model, which defines how components interleave their computations as well as how interrupt and non-interrupt code interact; (3) Application programming interfaces (APIs), services, component libraries and an overall component structure that simplify writing new applications and services.

As is illustrated in Figure. 3, WaterMonitorC is composed of five components of which, MainC implements the boot sequence of a node and provides the Boot interface so that WaterMonitorC can be notified when a node has fully booted. In our application, WaterMonitorC needs to sample periodically, so we can use startPeriodic(60000) command, which will signal a fired event once a minute. It's worth mentioning that there are three ActiveMessageC components which look similar but have entirely different responsibilities. For instance, PLCActiveMessageC is used for learning the states of auxiliary relays in PLC and the concentration of NO3-, PO43- and PH value. Communication between FFD and RFDs is the duty of CC2420ActiveMessageC. DTUActiveMessageC, used only in FFD, sends data packets to remote management platform by GRPS channel.

3.1. RFD

RFD has two main functions: 1. Read the concentration of NO3-, PO43- and PH value (as Sensor Data) in PLC; 2. Send Sensor Data to the FFD. The Sensor Data is read through HOSTLINK instruction. HOSTLINK is Omron's proprietary protocol, by which the external device can communicate with Omron PLC through serial port. By HOSTLINK protocol, RFD can visit DM (Data Memory) and other registers in PLC [6]. In our system, the Sensor Data, starting from 0012 unit, is stored in the DM area of PLC, each being represented by an 8-byte ASCII. In order to read the Sensor Data, TinyOS sends the HOSTLINK instruction "@00RD0012000653*↙", and then PLC sends back a HOSTLINK instruction as response which contains Sensor Data. It is to be noted that PLC stores the Sensor Data as a double-byte ASCII, of the 8 bytes only the late half are valid.

Figure 4(a) is a subset of the RFD state diagram, the advanced operations being elided for simplicity. WaterMonitorC initiates periodic sampling in its booted event by Timer.startPeriodic command. Timer.fired requests a new PLC sampling by sending HOSTLINK instructions using the PLCAMSend interface, and a PLCAM packet stored in a plc_sensor_data_t packet buffer. The plc_sensor_data_t holds the current Sensor Data after HOSTLINK response arrives. Then we restore the Sensor Data into node_sensor_packet_t (Figure. 5), and send it to FFD by CC2420AMSend interface. Thus, the packet is smaller and easier to transmit; besides, it identifies which monitoring node it was born of.

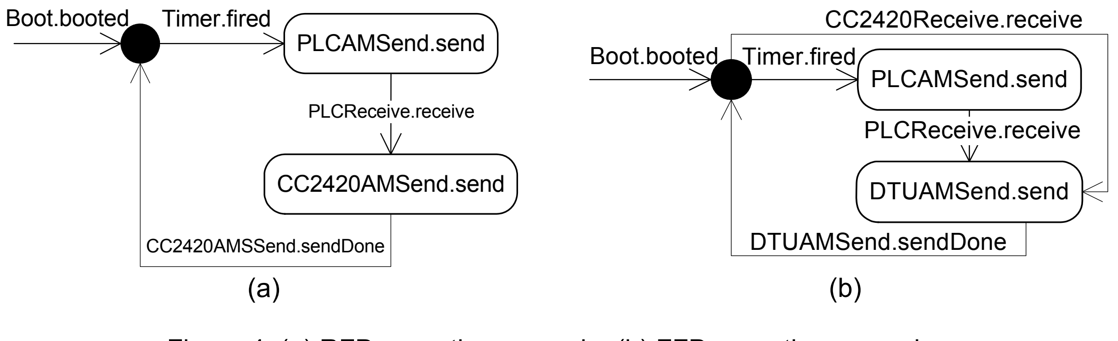

Figure 4. (a) RFD operating scenario; (b) FFD operating scenario.

3.2. FFD

FFD has functions as follows: 1. Receive node_sensor_packet_t variable sent from a RFD; 2. Read the Sensor Data in PLC; 3. Send every node_sensor_packet_t variable to the remote management platform. The remote transmission function is implemented by GRPS DTU. GPRS DTU is a device for conversion between serial data and IP packet. It has the PPP dial-up and TCP/IP protocol encapsulated in GPRS DTU, enabling transparent transmission between serial devices and remote computers [7].

Different from RFD, PLCReceive.receive does not retransmit HOSTLINK response, but sends the restored node_sensor_packet_t variable to the remote management platform by DTUAMSend interface. CC2420Receive forwards the node_sensor_packet_t variable by DTUAMSend interface after receiving it from RFD. When this is done, the TinyOS system returns to the original state waiting for next Timer.fired event. The FFD operating scenario is shown in Figure 4(b).

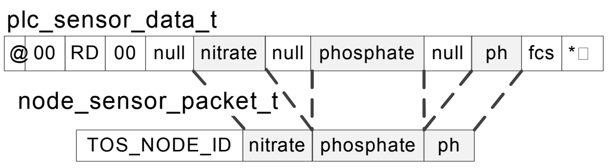

Figure 5. Relationship between plc_sensor_data_t and node_sensor_packet_t

4. Remote Management Platform

The architecture of remote management platform is shown in Figure 6.

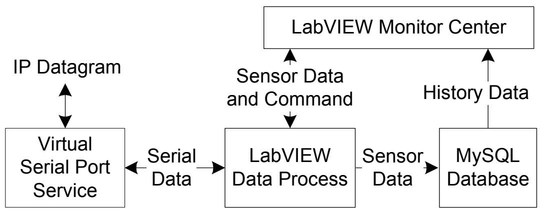

Figure 6. Architecture of Remote Management Platform

4.1. Virtual Serial Port Service

The remote management computer receives the IP packet which contains Sensor Data. For easy operation, the IP packet is converted into serial data by Virtual Serial Port Service. Then the serial data is used to communicate with upper procedures through the virtualized serial port.

4.2. Data Process Procedure

The Data Process Procedure, implemented by LabVIEW software, is responsible for binding the virtual serial port and receives Sensor Data from GPRS DTU. As float in CP1H PLC is stored as two 16-bit words in ascending order, LabVIEW must convert it before being transformed into float. The obtained sensor data is stored into the database and displayed in the monitoring center. This process is implemented by a producer-consumer pattern. In this way, the Sensor Data can be stored into MySQL database immediately in producer cycle, and is displayed in a constant speed in consumer cycle.

4.3. Data Process Procedure

MySQL, an open sourced relational database with high performance, can run in most operating systems. LabVIEW connects MySQL via Database Connectivity Toolkit and ODBC interface. The Monitoring Center is not only responsible for displaying the Sensor Data of monitoring nodes in real time but also achieves the functions of historical inquiry and abnormal alarm.

5. Other operating systems for WSN

Besides the TinyOS, there are some other embedded operating systems designed for the sensor networks.

Contiki is an open source operating system for networked, memory-constrained systems with a particular focus on low-power wireless Internet of Things devices. Examples of where Contiki is used include street lighting systems, sound monitoring for smart cities, radiation monitoring systems, and alarm systems. Contiki was created by Adam Dunkels in 2002 and has been further developed by a world-wide team of developers from Atmel, Cisco, Enea, ETH Zurich, Redwire, RWTH Aachen University, Oxford University, SAP, Sensinode, SICS, ST Microelectronics, Zolertia, and many others. The name Contiki comes from Thor Heyerdahl's famous Kon-Tiki raft.

ERIKA Enterprise is an open-source and royalty-free OSEK/VDX Kernel. This RTOS includes also RT-Druid, which is a development environment distributed as a set of Eclipse plugins. ERIKA Enterprise implements various conformance classes, including the standard OSEK/VDXconformance classes BCC1, BCC2, ECC1, ECC2, CCCA, CCCB. Moreover, ERIKA provides other custom conformance classes named FP (Fixed priority), EDF (Earliest deadline first scheduling), and FRSH (an implementation of resource reservation protocols).

Nano-RK is a real-time operating system from Carnegie Mellon University designed to run on micro-controllers for use in sensor networks. Nano-RK supports a fixed-priority fully preemptive scheduler with fine-grained timing primitives to support real-time task sets. "Nano" implies that the RTOS is small, consuming 2KB of RAM and using 18KB of flash, while "RK" is short for resource kernel. A resource kernel provides reservations on how often system resources can be consumed. For example, a task might only be allowed to execute 10ms every 150ms (CPU reservation), or a node might only be allowed to transmit 10 network packets per minute (network reservation). These reservations form a virtual energy budget to ensure a node meets its designed battery lifetime as well as protecting a failed node from generating excessive network traffic. Nano-RK is open source, is written in C and runs on the Atmel-based FireFly sensor networking platform, the MicaZ motes as well as the MSP430 processor.

LiteOS is a real-time operating system from University of Illinois for use in sensor networks. LiteOS is a UNIX-like operating system that fits on memory-constrained sensor nodes. This operating system allows users to operate wireless sensor networks like operating Unix, which is easier for people with adequate Unix background. LiteOS provides a familiar programming environment based on UNIX, threads, and C. It follows a hybrid programming model that allows both event-driven and thread-driven programming. LiteOS is open source, written in C and runs on the Atmel based MicaZ and IRIS sensor networking platform.

OpenTag is a DASH7 protocol stack and minimal Real-Time Operating System, written in the C programming language. It is designed to run on microcontrollers or radio Systems on a Chip (SoC). OpenTag was engineered to be a very compact software package. However, with proper configuration, it can also run in any POSIX environment. OpenTag can also provide all functionality required for any type of DASH7 Mode 2 device, rather than just the eponymous “tag”-type endpoint device

6. Conclusion

As a real time operating system optimized for WSN, TinyOS is first employed in the field of water quality monitoring in our system to achieve the goals of water quality data collection and its remote transmission. The system suggests the component-based programming method and the event-driven operating mechanism, which is independent of network infrastructure, flexible and affordable to be implemented and has a promising prospect. However, the system is far from being perfect. More research is needed concerning hidden terminal, out-of-order packet and other issues in the future. Besides, the balance between system lifetime and transmission rate is underd consideration.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 61273070), the Programme of Introducing Talents of Discipline to Universities (B12018), PAPD of Jiangsu Higher Education Institutions and the Fundamental Research Funds for the Central Universities (JUSRP11132).

References

[1] Zhao X. *Research on environment system for water monitor*. International Conference on Intelligent System Design and Engineering Application. Changsha. 2010; 2: 197-199.

[2] Alex A, Jenny DR. *Designing an automated water quality monitoring system for West and Rhode Rivers*. Proceedings of the 2009 IEEE Systems and Information Engineering Design Symposium. Charlottesville. 2009; 131-136.
[3] Joa-Ng Mario, Lu I-Tai. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*. 1999; 17(9): 1415-1425.
[4] Azzedine B. Algorithms and protocols for wireless sensor networks. Hoboken: John Wiley & Sons. 2009.
[5] Gay D, Levis P, Culler D. Software design patterns for TinyOS. *ACM Transactions on Embedded Computing Systems*. 2005; 40(7): 40-49.
[6] Omron Corporation. Communications commands reference manual. Kyoto. 2010.
[7] Metz C. A pointed look at the point-to-point protocol. *IEEE Internet Computing*. 1999; 3(4): 85-88.