AC 2009-824: DESIGNING A CONTINUOUS MONITORING AND TRACKING SYSTEM BASED ON A WIRELESS SENSOR NETWORK

Jimmy DiTraglia, Middle Tennessee State University

Xiaojing Yuan, University of Houston

Mequanint Moges, University of Houston

Design of a Continuous Monitoring and Tracking System based on Wireless Sensor Network

Abstract

Today the way we work and live has been changed by the deployment of ubiquitous intelligent wireless sensor networks. The infusion of such emergent technology into the current under-graduate lab designs becomes a critical issue in order to prepare and engage our students for future engineering and technology development. Such new modules will allow students to have improved learning experience through more involvement in research and hands-on activities and better outcome. This paper presents the experience of undergraduate research during summer 2008 supported by NSF REU program at the University of Houston on "Sensor Networks and security Infrastructure". The project also serves to upgrade existing upper level design projects that aim at introducing research components into the curriculum of the computer engineering technology program.

Introduction

Wireless smart sensor networks have the ability to integrate sensing, communication and computation and are being implemented in a wide range of data gathering and decision making applications. Their low cost, compact size and low power consumption makes them ideal for wider deployment. This presents considerable difference from the traditional networks and as the result requires proper design in terms of the protocols used, energy use as well as mobility in smart wireless networks must be carefully studied. The integration of smart sensor networks with wireless transmission systems are expected to provide a wide range of services with various constraints including but not limited to variable quality of service and time-varying bandwidth requirements. Recently various applications of wireless smart sensor networks such as remote measurement systems[1, 2], remote lab simulation of actual experiments[3] have been realized with minimum bandwidth and cost.

A critical issue with wireless sensor nodes is effective power management to extend battery life. The reason this is such an important issue is because it is often difficult or infeasible to replenish power supplies of these wireless devices. The power supply in the sensors is used for several purposes: for sensing, for powering the memory and CPU, and for communication with other sensors in the network. Different applications and sensors draw different amounts of energy but most literatures indicate that communication is a major power drain[4, 5]. Some sensors try to be aware of this and will shut down certain components like radio when they are not needed. They may also scale voltage or frequency of the processor depending on the workload[6].

Well engineered sensors can result in power conservation but for a variety of applications, further gain can be observed if the network is designed properly and by the algorithms implemented in the program to be downloaded onto the microprocessor. In this paper we present the design and development of a continuous monitoring and tracking system using emergent smart wireless sensor network that can be widely used in applications such as security system, patient monitoring and tracking, etc.

The project involves the design and implementation of a method for power conservation by only allowing three closest wireless sensor nodes to send their data to the base station. This is necessary because as mentioned earlier, communication is the biggest power drain and these motes will be communicating continuously to determine which three are closest to the object. This paper presents the details of the design architecture, the hardware and software implementtation and a simple localization algorithm based on optical sensor to track an object moving around in the wireless sensor network.

The remainder of this paper is organized as follows. The next section will present the hardware and software platform used for the sensor nodes. The third section will present details of the of the algorithm development. The detailed project design including preliminary result is presented in the fourth section. Conclusion and future works is presented finally.

Sensor Network Testbed

Our sensor network consists of 17 MICAz motes manufactured by Crossbow Technology[7]. While the sixteen motes make up the sensor network, one mote is configured as a base station /gateway to receive data sent from the sixteen motes. We now briefly review the hardware and software structure of these motes.

Hardware

The MICA platform is based on a single central microcontroller that performs all the sensing, communication, and computation tasks. These motes use the ATmega 128L processor manufactured by Atmel[8] as shown in figure 1. The RF module is composed of an RF Monolothics 916.50 MHz transceiver (TR1000). It can be externally controlled by a potentiometer to have a communication radius from a few inches to several yards and can operate at data rates up to 115 kbps. The antenna for the radio is integrated onto the circuit board so connecting the visible external antenna is optional. The motes run off a pair of AA batteries for power supply.

The sensor board, MTS310, has several sensors that can be used to gather data. Theses sensors include: light sensor, sound sensor, magnetic field sensor, acceleration sensor, and a temperature sensor. These sensors are used to acquire data and then the data is converted through the 10-bit ADC.

Figure 1. MICAz Sensor Mote

Software

The MICAz motes run on event-based operating system called Tiny OS which fits into 178 bits of memory. It manipulates the hardware directly and there is no kernel layer. There is no dynamic memory allocation so the memory is allocated at compile time. The Tiny OS code and applications are complied together and run in a single linear address space; this reduces the memory management overhead. Within the Tiny OS there is a collection of software *components*. The complete system software consists of a scheduler and an interconnection of these components. There are three types of components: Hardware Abstraction Components, Synthetic Hardware Components, and High Level Components. The Hardware Abstraction Components directly map to a hardware device such as the LEDs, UART, or the ADC, and manipulate them. The Synthetic Hardware Components are used to simulate the behavior of a certain component if it is absent in the mote. The High Level Components perform various data manipulations and transformations such as a routing algorithm or any other application.

The Tiny OS system provides two levels of scheduling, events and tasks. Events are synchronous and are carried out until completion the moment they occur. They cannot be preempted. Tasks can be preempted by an event. They are asynchronous and involve time consuming computations. Tasks are scheduled when no events are to be processed.

We used the MoteWorks software to work with the MICAz motes. This software also is developed by Crossbow Technology. The programming for Tiny OS is done in a language called NesC which is derived from the C language. The programming and compiling is done with a program called Programmers Notepad which comes with the download of the MoteWorks software. The motes send data from the light sensor to a base mote referred to as the base station /gateway. This base station is connected to the PC via USB and the data is read with the XSniffer software. XSniffer is also included in the download of the MoteWorks package.

The Problem: Tracking Moving Objects

The sixteen motes used for tracking the object are set up in a 4x4 grid. In this case an incandescent light bulb is the source to be tracked. Light sensor data is sampled rapidly so the motes can quickly update as the light source moves within the network. Based on the data sent to the base station, we can estimate the distance of the object from the motes. Tracking the object involves a tracking algorithm in our program downloaded to the Atmel microprocessor and also a computation algorithm with MATLAB used to estimate the distance of the object from the three closest motes.

Implementation of the Tracking Algorithm

The tracking algorithm is easily implemented in both hardware and software. Every 0.4 seconds each node samples the light sensor and our algorithm compares the data of each node with the data from the other 15 nodes. First the closest node is determined, this will be known as the head node. After the head node is found, the second and third closest nodes will also be determined based on the data sampled by their light sensors.

The first thing that happens after the light sensor is sampled is each node broadcasts its data to all the other nodes in the network. This allows each node to receive the data from every other node in the network. The light data is an 8-bit number so the maximum reading that can be displayed on XSniffer for any mote will be 255.

Once the data has been sent from each mote to all the other motes, the algorithm begins its first step which is to find the mote closest to the object. The head data, data from the head node, is initialized with a value of zero and the algorithm compares that value with the value of the first data packet received. If the data packet sent is larger than the head data, which it will be because the head data is zero, then the data from that packet is the head data and the node ID of that node is now the ID of the head node. This process repeats and the values of head data and head node are only changed if the data compared with it is larger. After data from all sixteen motes have been compared, the head data and head ID have been successfully computed.

This algorithm must repeat once again to find the second and third closest motes to the object, only this time the head data and head ID from the previous cycle must be excluded. Of course to find the third closest mote, the head data and the data from the second closest mote is excluded. The LEDs on the sensor boards are used to indicate which motes are the closest, second closest, and third closest. The head node will turn on the yellow LED, second closest node will turn on the green LED, and the third closest will turn on the red LED.

We used XSniffer to display data readings sent from the motes. In XSniffer, all the data will be displayed that is broadcasted to the entire network. This is the data sent from each mote to all the other motes so the algorithm can compare and determine the three closest to the object. For power conservation purposes, we only allow the three closest motes to send their data to the base station. This can be easily observed in the XSniffer display because data sent to the base station address is highlighted in green. This data from the three closest nodes is used with the computation algorithm. Figure 2 shows some snapshots from XSniffer.

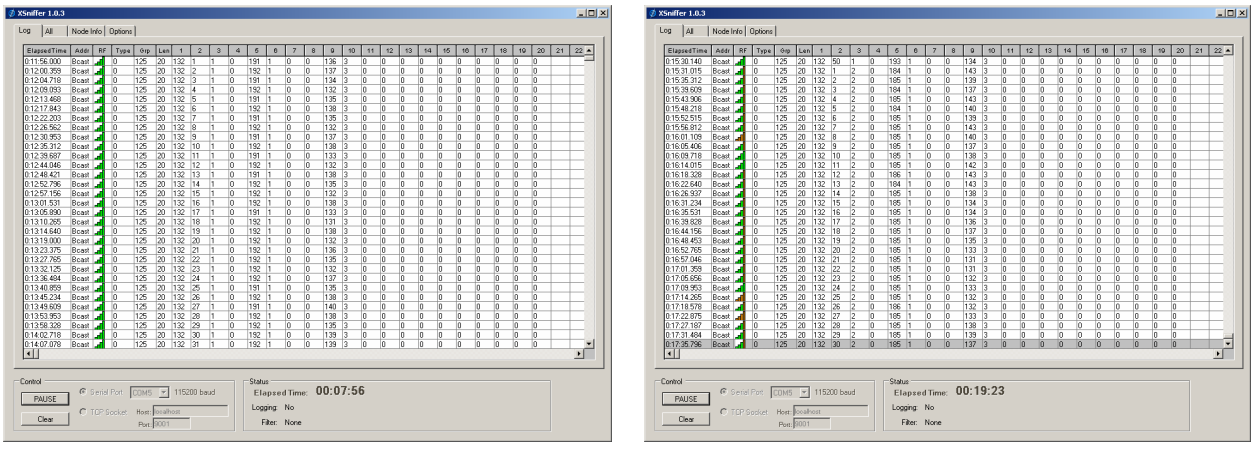
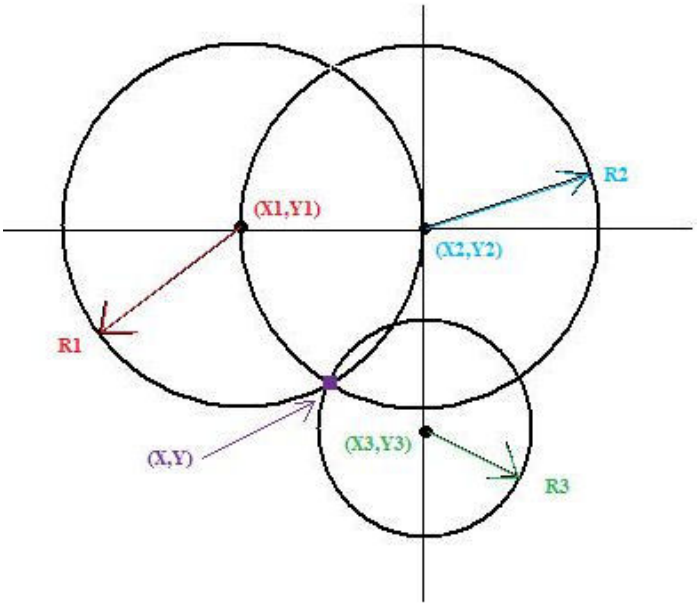


Figure 2. Snapshots from XSniffer for the tracking program

Implementation of the Computation Algorithm

The computation of the location of the light source in the network grid could have been done with several different algorithms. One method collects data based on the received signal strength (RSSI). However, the object being tracked would have to have a radio signal, which the light source would not put out. Another method, based on time of arrival (ToA) could have been used, but again, a signal was necessary on the object being tracked. Because the idea of this project is to be oriented towards security systems, and most security systems do not track radio signals, tracking the light itself became necessary. To do this, trilateration was used. Trilateration is a method that involves finding the distance from three different nodes that the object is, and then using those distances to pinpoint the location. Using the image and equations below, the location of an object can be found.



$$x = \frac{r_1^2 - r_2^2 + (x_2 - x_1)^2}{2(x_2 - x_1)} \qquad (1)$$

$$y = \frac{r_1^2 - r_3^2 - x^2 + (y_3 - y_1)^2}{2(y_3 - y_1)}$$

Figure 3. The trilateration method

The Computational Algorithm that has been created using MATLAB will take user input from the XSniffer, and output the x and y location of the light source, however, there are several things that need to be accounted for once the trilateration equations are used for more than one square of node locations. One thing that needed to be taken into consideration is that the three nodes nearest the light source may all be in a row in the x or y direction, meaning that the location is somewhere in that line, in the middle of those nodes. Another thing that needs to be accounted for, is that the basic equations listed above do not tell the correct location if data from nodes not in the square of 4 nodes around (0,0). Therefore, the data must be shifted along the axis. The program also ensures that it does not attempt to divide by zero.

Experimental Results and Discussion

As the light source moves around within the sensor network, the motes continuously monitor its location. The three closest nodes are indicated by LEDs on the motes. The closest one is indicated by the yellow LED, second closest by the green LED, and the third closest by the red LED. The LEDs on the other motes in the network remain off. The light source used emits a

somewhat concentrated beam of light but also has a very dim outer ring. Sometimes we noticed the red LED flicker on some of the motes that did not appear to be one of the closest three to the light source. This may be attributed to that mote's photocell being more sensitive than the others and picking up small amounts of light from the outer ring. Even though they are small amounts of light, with the photocell being more sensitive than the others it may cause that mote to be computed as the third closest to the light source which would be the reason for the flickering of the red LED.

The only issue we have is that the radio signal strength is very low. When XSniffer displays the signal strength it shows a maximum of four bars and all of the data sent shows either one or zero bars. The network was about 20 feet away from the base station when the experiment was done. Our concern is that since all the data is sent with such low signal that maybe some of the data packets did not get sent to the base station and this could possibly cause errors in reading the values of the three closest motes. This could mean that the distance estimates given from the computation algorithm may be slightly inaccurate.

Our computation algorithm gives the location of the object as x and y coordinates. We can enter the node ID and data of the three closest nodes to the object as it is observed using XSniffer. Below is an example of the algorithm being executed and displaying the proper results:

TIME	TYPE	NODE #	LIGHT
0:06:25	Base	2	117
0:06:25	Bcast	2	116
0:06:25	Base	5	251
0:06:25	Bcast	8	15
0:06:25	Bcast	5	137
0:06:25	Bcast	7	255
0:06:25	Base	6	166

```
If there is not a third head node, pick
different data points!!!
if you do not have three, ctrl-c to quit.
insert first node number then press enter:
2
enter light data for this node: 117
insert second node number then press
enter: 5
enter light data for this node: 251
insert third node number then press
enter: 6
enter light data for this node: 166
x= 2.6834, y=1.6783
```

Figure 4. Example of computation and tracking algorithm

Conclusion and Future Works

In today's fast changing technology a dynamic research environment for students in general and undergraduate students in particular has proven to be the way for breakthroughs and knowledge transfer. This paper presented the research experiences of undergraduate students in the application of wireless sensor networks for object tracking. The project involved the development and implementation of a wireless sensor network that successfully tracks the movement of an object. Three LEDs on the motes were used to indicate the three closest motes to the object in order to find the distance of the object from the three mote locations. The only data that is sent to the base station is the data from the three closest motes. This is critical because communi-

cation is the biggest power drain on the motes. We believe such excitement and involvement in undergraduate research motivates the classroom and is a major step forward in the improvement of the curriculum for quality instruction eventually.

References

1. Arpaia, P., Baccigalupi, A., Cennamo, F. and Daponte, P. "A measurement laboratory on geographic network for remote test experiments, IEEE Trans. Inst. Meas., 49, (5) 2000, 992-997.
2. Fortino, G., Grimald, D., Nigro, L. " An agent based measurement laboratory over internet, Proc. of the IEEE Automated Test Conference, San Antonio, TX, USA, 1999, pp 61-71.
3. Thiriet, J., Robert, M., Martins, M. and Hoffmann, M. "Pedagogical resources reachable via internet for teaching intelligent instruments: developments within a European thematic networks, Proc. of the IEEE IMTC 2002, AK, USA, 1475 – 1479.
4. G.J. Pottie andW.J. Kaiser.Wireless Integrated network sensors *Communications of the ACM,2000.*
5. M. A. Moges and Thomas G. Robertazzi, "Wireless Sensor Networks: Scheduling for Measurement and Data Reporting", IEEE Transactions on Aerospace and Electronic Systems, Vol. 42, No. 1, pp. 327-340, January 2006.
6. RF Monolithics. http://www.rfm.com/products/data/tr1000.pdf.
7. Crossbow Technology, Inc. http://www.crossbow.com.
8. ATMEL 8-bit RISC Processor. http://www.atmel.com/products/prod23.htm.