Emulation of Wireless Sensor Networks for Object Tracking

T. Andrew Yang[+] (yang@UHCL.edu)
Deepesh Jain[+] (deeptansh@gmail.com)
Sadegh Davari[+] (Davari@UHCL.edu)
Bo Sun[++] (bsun@my.lamar.edu)

[+] Division of Computing & Mathematics, University of Houston – Clear Lake, Houston, Texas

[++] Department of Computer Science, Lamar University, Beaumont, Texas

Abstract

Wireless sensor networks (WSN) represent a new technology that could be adopted to achieve ubiquitous computing and embedded Internet. One example of WSN applications is surveillance and monitoring of a deployment area. The goal of our project is the development of effective WSN for detecting and tracking human and/or vehicle presence. In this paper, our focus is two-folded. We first examine the issues of tracking objects using wireless sensor networks, including a brief introduction to object tracking, a survey of work related to object tracking using WSN, and the challenges of designing an object tracking sensor network using the emulation approach (in contrast to the simulation approach). The second focus of this paper is the design of a sample wireless sensor network application capable of detecting human presence. In the sample application, motes and sensor nodes capture data from the surrounding environment and send the data to the base station. A Java application running at the base station analyzes the data to determine whether a human is present. In addition to discussing our experience of developing the project, we examine the challenges of emulating a wireless sensor network, difficulties encountered during the project, and the algorithm implemented in the application. We also propose the design of a multi-node WSN for detecting/tracking humans. Projects are currently underway to implement and evaluate the design.

1. Introduction

A wireless sensor network (WSN) is a network made of a set of independent sensor nodes. The sensor node is a self-contained unit consisting of a battery, a radio module, sensors, and a low speed on-board processor. A WSN system is deployed over an area in an attempt to sense and monitor events of interest or to track people or objects as they move through the area [1] [2].

Wireless sensor networks have significant impact upon the efficiency of military and civilian applications, which may be classified into three classes [2]: (a) Data collection, (b) Surveillance, and (c) Object tracking. Object tracking is one of the most prominent applications of wireless sensor networks. As an example, a large quantity of sensor nodes could be deployed over a battlefield to detect enemy intrusion instead of using landmines. Thus it can save lives of civilians from being lost due to disasters caused by landmines. This project is to demonstrate how a wireless sensor network may be employed to detect and track objects.

Wireless sensor networks represent a relatively new research filed with many research projects going on. However, most of the WSN projects are implemented using simulation, rather than emulation. One reason behind this phenomenon is the increased difficulty when actual devices are used in implementing a WSN research project. Emulating a WSN, actual hardware devices are needed; therefore the project cost and complexity are increased. In contrast, in the widely adopted simulation approach, the researcher is only

required to acquire and use the simulation software. Emulation becomes even more complicated when the deployed WSN consists of a large number of sensor nodes. Another limitation of the emulation approach is the constraint of limited battery power; exhausted batteries of sensor nodes must be replaced and, to ensure accurate sensor readings, proper level of power must be maintained throughout the running of a WSN project.

Designing a wireless sensor network requires a proper understanding of the interplay between network protocols, energy aware design, and signal processing algorithms and distributed programming [3]. Nevertheless, different types of WSN applications pose different level of challenges. Developing a sensor network for sensing temperature or light, for example, is not as challenging as developing a sensor network for detecting and tracking motion of objects. Detecting temperature or light does not require much computing or logic. Data is directly available in the packets sent by the sensor nodes; the only task that remains is to present the data in readable form to the user. However, detecting and tracking motion of objects typically involves complicated algorithms. Since sensor nodes are not equipped with powerful processors, complicated computations are usually done somewhere else, whether in the base station or in one of the aggregation nodes, which aggregate the raw data fetched from the sensor nodes and produce a local result. Furthermore, temperature and light in general can be detected by the sensors smoothly because their distribution in the environment is typically ubiquitous or even homogenous. In contrast, in an object tracking sensor network, the object to be detected/tracked occupies a limited area in the environment, depending upon its shape and size. To make it worse, the speed of a moving object may be so fast that the sensors may not be able to accurately detect its location. Therefore, because all of the reasons mentioned above, emulation is typically not the chosen approach when developing an object detecting and tracking WSN application; it incurs many challenges, involving various tasks including detecting the relevant quantities, monitoring and collecting the data, assessing and evaluating the information, and presenting meaningful information [4].

When deploying a wireless sensor network, the following list represents the typical hardware components needed:

a) Motes: A mote is a low powered computer with a radio transmitter, capable of forming ad hoc communication with other motes [5]. A mote may be connected to one or more sensor boards and therefore may perform multiple sensing functions.

b) Sensor boards: A sensor board is a chip on which one or more sensors are present.

c) Gateway: A gateway is a device responsible for injecting queries into the sensor network, gathering responses from the network, and presenting the responses to the user's workstation. The gateway communicates with the WSN through short-range wireless links, and interacts with the user directly or remotely through a wired or mobile communication network [6].

d) Monitoring workstation: The monitoring workstation is usually a PC with required compatible software installed, and is used by the user to configure the WSN, to submit queries to the network, or to view the data collected by the network.

Crossbow, EasySen, and MoteIV are some of the manufacturing companies that provide hardware and software solutions for constructing wireless sensor networks. For object tracking application using WSN, Crossbow provides the MSP410[1] system, which is a packaged system that consists of motes and sensors capable of detecting moving objects. Our findings indicate that the TelosB mote from Crossbow and the Tmote mote from MoteIV can be used together with the WiEye or the SBT80 sensors from EasySen to construct a wireless sensor node capable of detecting objects. With the help of the available hardware, this project is to build a wireless sensor network to track objects. Based on information collected from the vendors' websites [7] [8] [9], this goal could be accomplished in three steps:

[1] Note: MSP410 has been discontinued by Crossbow, Inc. since early 2007.

(1) Testing of available sensors with simple applications, and selection of sensors and motes for further deployment.

(2) Deployment of a WSN and testing of simple wireless sensor networks applications with the selected sensors and motes.

(3) Testing of object tracking applications in the deployed wireless sensor network.

The rest of this paper is organized as follows. Section 2 review existing work related to tracking objects using WSN. Section 3 examines the unique challenges of developing a wireless sensor network for object tracking using the emulation approach. In section 4, we present our design of a WSN using emulation, by discussing the hardware, the software, the Java application that we developed for detecting human presence, the experiments we conducted, and the results of the experiments. Section 5 concludes the paper, with discussions of future work.

2. Work Related to Object Tracking in WSN

In this section, we discuss the results of our surveying work related to tracking objects in WSN.

At the University of Notre Dame, the team *Moses* has presented a novel, unified approach named non-recursive evidence filtering [11]. Their approach is based on the Dempster-Shafer formalism [10] for evidence representation. The system they built is capable of selectively fusing partial evidence in a network to directly infer events of interest such as threats occurring with a certain temporal distribution, while accommodating the varying reliability and accuracy of information sources [11].

Tsai, etc. at the National Cheng Kung University propose a protocol to track a mobile object in a sensor network dynamically [12]. They claim that previous researches mostly focus on how to track objects accurately and do not consider the query for mobile sources. Additionally, the sensors need not report the tracking information to the user. Therefore, they propose to concentrate on mobile user's capability to query target tracks and obtain the target position effectively. The mobile user can obtain the object position without broadcast queries. The user is moving and approaching the target when he knows the target's position. Their paper proposes a binary sensor model, in which each sensor's value is converted to a single bit, which represents whether the object is moving toward or away from the sensor [12].

Kung and Vlah [14] consider a sensor network as a distributed database, and propose a scalable message-pruning hierarchy tree called *DAB* (*Drain-And-Balance*) for object tracking. The tree is a logical tree to connect all sensors. Each internal node in the tree maintains a set containing its descendants' coverage. Lin and Tseng [13] extend the approach proposed in [14]. Instead of assuming the existence of a logical tree, they try to realize the logical tree by the sensors directly. In this manner the real communication cost can be evaluated more accurately. Their paper proposes two message-pruning tree structures called *DAT* (*Deviation-Avoidance Tree*) and *Z-DAT* (*Zone-based DAT*). The authors formulate communication costs associated with trees. Through simulations, they demonstrate the advantage of their approach [13].

Aslam, Butler, etc. [15] examine the role of very simple and noisy sensors for the tracking problem. Their paper proposes a binary sensor model, in which each sensor's value is converted reliably to one bit of information only, independent of the object's moving direction. Their paper shows that a network of binary sensors has geometric properties that can be used to develop a solution for tracking with binary sensors, and presents the resulting algorithms and simulation experiments. They develop a particle filtering style algorithm for target tracking using minimal number of sensors. They presents an analysis of a fundamental tracking limitation under this sensor model, and show how this limitation can be overcome through the use of a single bit of proximity information at each sensor node. They claim that their extensive simulations show low error, which decreases with sensor density [15].

He, etc. [16] presents the design and analysis of VigilNet, a large-scale outdoor WSN which tracks, detects and classifies targets in a timely and efficient manner. Through simulation and experiments, He, etc. demonstrate that their system can meet the real-time requirement and its tradeoffs are validated. On the basis of deadline partition method and theoretical derivations to guarantee each sub-deadline, they make guided engineering decisions to meet the end-to-end tracking deadline.

Tran and Yang [17] consider optimized computation and energy dissipation as critical requirements to maximize the lifetime of the wireless sensor network. They consider that the existing methods attempting to achieve these requirements, like LEACH based algorithms, suffer either redundancy in data or sensor node deployment, or complex computation incurred in the sensor nodes. They therefore propose a new method called Optimized Communication and Organization (OCO), which is an algorithm that ensures maximum accuracy of target tracking, efficient energy dissipation, and low computation overhead on the sensor nodes. Simulation evaluations of OCO are compared with other two methods under various scenarios [17].

3. The Challenges of Developing WSN for Object Tracking using Emulation

As evident in the earlier discussions, most WSN research and educational projects rely on simulation. In particular, very few WSN research projects adopt emulation when the project involves the detection and tracking of moving objects, such as vehicles or humans. As reported by Singh [4], object tracking using WSN involves a set of tasks, including collecting data from the WSN to the main station, designing and implementing algorithms for the tracking application, ensuring real-time communication, and interpreting the data for object tracking; these tasks make developing a WSN for object tracking using emulation more challenging than the typically adopted simulation approach.

In this section, we examine WSN research projects that adopt the emulation approach to deploy a network for detecting an object in motion. We first give overview of each project, and then analyze their strengths and weakness.

Salatas [1] constructs an object-tracking system that demonstrates a real-world application using a WSN to track objects and communicate the tracking information to the base station. The system consists of an event-driven application implemented in Java, built on top of the Crossbow MSP 410 wireless sensor system. The algorithm implemented in the application processes the data returned from the WSN detection signals and tracks the object's motion. Figure 1 shows the design of Salatas' system. The wireless sensor network is connected with a system called the Tactical Remote Sensor System, which includes several miniPCs, a Globalstar Phone, and a WebCam. In addition, the system includes an FTP server. The system detects objects by

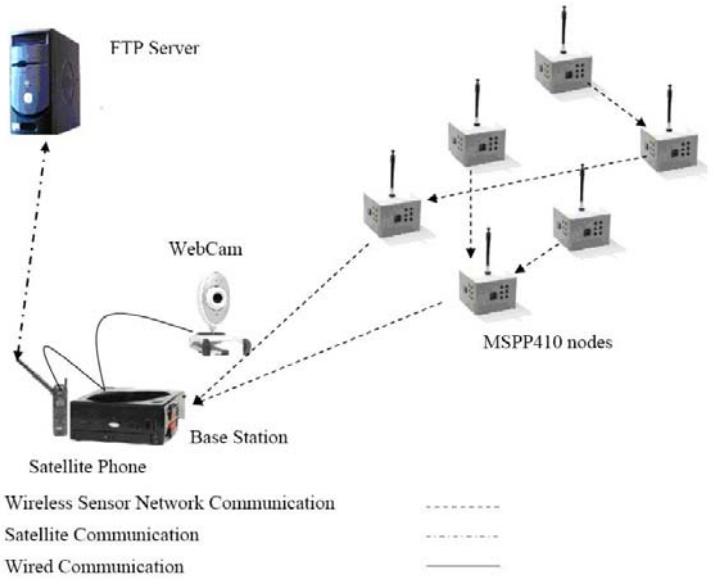

Figure 1. Salatas' WSN Object Tracking System [1]

performing picture comparisons. It then transmits those pictures to the control station by using satellite or cellular communications [1].

This object-tracking application receives and uses the raw values returned by the sensor network to produce meaningful outputs. The application basically detects human or vehicle type of objects on the basis of infrared or magnetism data sensed by the sensors. If, in consecutive values of raw data, the magnetism and infrared values are changed and are greater then the normal environmental magnetism and infrared values of the deployed area, the application infers the detected object as a vehicle. If, on the other hand, the changed value is only the infrared values and not the magnetism, the application considers that the detected object as a human [1]. Salatas' project is one of the few WSN object tracking projects using emulation. The application has certain limitations: (a) it is only tested for a small number (8) of nodes; (b) scalability of this project was not explored; (c) it has limited deployment scenarios, in which nodes are deployed on a straight road, on a T-Shaped road, or a cross road intersection. Detection of free motion of objects is therefore not considered in this application.

In Singh's project "Real-time Object Tracking with Wireless Sensor Networks" [4], he developed a dynamic system with WSN, which uses the Time Division Multiple Access (TDMA) protocol with the Extended Adaptive Slot Assignment Protocol (EASAP). EASAP was shown to be a suitable protocol to be used in a dynamic WSN. Localization of source is presented by using an algorithm that is demonstrated using different configurations named X-Configuration, T-Configuration and Y-Configuration. In these configurations nodes are placed according to the alphabets. The data source in the demonstration is light generated by a lamp. Source localization algorithm uses averaging of signal strength detected by sensors and is proven to be well suited for this application [4].

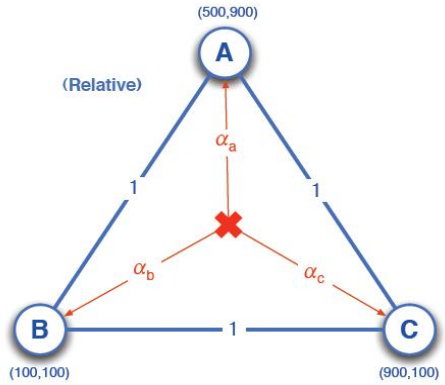

Figure 3. Source localization example with three nodes [4]

A source localization example is illustrated in Figure 3. As shown, three sensor nodes are placed with an equal distance relative to each other. When a signal is received of strength $\boldsymbol{\alpha_j}$ by each ***j***th sensor, the coordinates of the source is calculated by the averaging algorithm as follows:

$$x = (500.\alpha_a+100.\alpha_b+900.\alpha_c)/ (\alpha_a+\alpha_b+\alpha_c), \text{ and}$$

$$y = (900.\alpha_a+100.\alpha_b+100.\alpha_c)/ (\alpha_a+\alpha_b+\alpha_c) \text{ [4].}$$

Singh's project was tested with a maximum number of six nodes, and the results show some failure in measured positions of objects compared to the actual physical positions of the objects.

As shown by the above discussed examples, emulation of WSN is a challenging task, especially when it comes to object tracking. Each of the examined projects exhibits certain limitations. Although still in its infantry stage, our project aims to develop a wireless sensor network system that incorporates scalability into its design, and is not restricted to fixed deployment scenarios but capable of tracking the free motion of objects under consideration.

4. Description of a Sample Human Detection Project

This project is built to detect the presence of a human when he/she comes in the vicinity of the sensors used in the system. To recognize a human, we had developed a Java application, which runs on the base

station. The application collects data from the network and analyzes the collected data to detect the presence of human. PIR (Passive InfraRed) sensors are used to collect data. Initial data is collected from the environment when no human is present, and a threshold value is set. Later when a human comes into the vicinity of the sensor, it compares the current collected value with the threshold value and makes decision of human presence. The rest of this section describes the design of the project, and the hardware and software used. Evaluation of the project is briefly described.

4.1. Design of the system

Figure 4 depicts the design of our system. A sensor node is comprised of a battery-powered mote and a sensor board, which is mounted on top of the mote. A gateway connected to the base station is in charge of passing the collected data from the network to the base station. The Java application running on the base station analyzes the data collected from the network to detect the presence of a human.

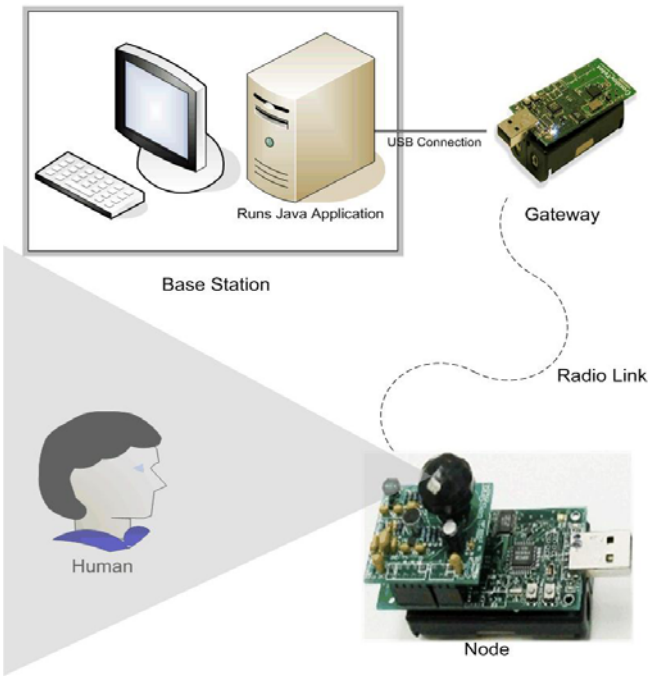

Figure 4. Design of a WSN Detecting Human Presence

The specific hardware components employed in the system are summarized as follow:

- The mote used is the TelosB mote (TPR2400 [7]) manufactured by the Crossbow, Inc.
- The sensor board is the WiEye sensor board manufactured by the EasySen, Inc. It has long-range passive infrared (PIR) sensor with 90-100° wide detection cone, 20-30 feet detection range for human presence [8]. As shown in Figure 4, the sensor board is mounted on a TelosB mote to form a node.
- No special gateway is used. As shown in Figure 4, a TelosB mote is connected to the USB port of the base station to pass data between the base station and the network.
- The base station is a desktop computer, on which Cygwin and Java virtual machine are installed.

The software employed in the system includes the following:

- Cygwin is used to configure the WSN. Applications in motes are installed through its command based interface. The base station application is also invoked from this software.
- Programmer's Notepad is used to write, compile and debug the NesC program, which is installed in the motes to collect data from the environment.
- Java SDK 1.6.0 is used to develop the Java application, which processes data collected from the network and outputs the results to the screen of the base station.

4.2. A Java application capable of detecting human presence

The TinyOS and Java code for reading sensor channel of the EasySen sensor board (SBT80) over the TmoteSky / TelosB motes are available at the EasySen website[2]. We modified those codes to suit the need of our project.

As shown in Figure 5, the application initially fetches data from the environment in absence of any object. Once the application starts fetching data with less variation or constant data, a threshold value is set. In this experiment the PIR threshold value was set to be 10. Afterwards, the application continuously receives data from the network. If the detected value becomes greater than the threshold value, it indicates an object is detected. When that happens, the tracking algorithm may be applied (if tracking is enabled), or otherwise the application continue to collect data. The tracking algorithm is yet to be developed. The current application only deals with object detection.

The procedure of configuring the system and running the application is discussed in details in Appendix A.

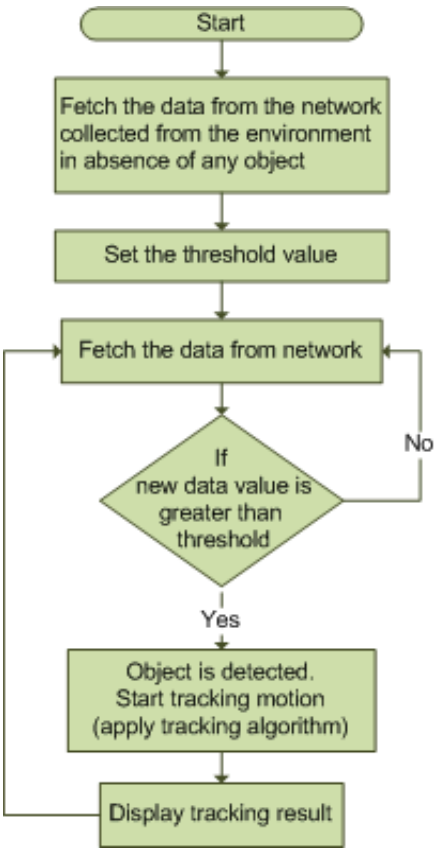

Figure 5. Flowchart of Human Presence Detection Program

4.3. Experiments Conducted

Several experiments were conducted in the lab to test the working of the application. Currently the project is tested with a single node. The application is able to detect the presence of a human in front of a node. The application is also able to detect the presence of a human in both lighted and dark environments. It is independent of the level of light present in the environment.

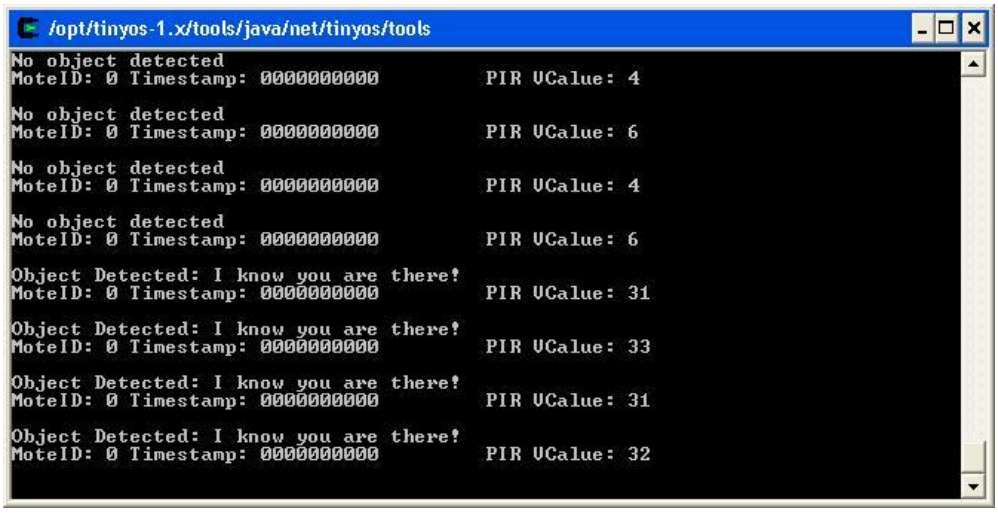

Figure 6. Screen snapshot of Cygwin, showing the output of the Java application detecting human presence

Anticipated evaluation results: If a human is not present in the vicinity of the sensor, a message "No object detected" is shown on the screen. On the

[2] EasySen SBT80 Multi-Modality Wireless Sensor Board, http://www.easysen.com/support/SBT80v2 (9-15-2007)

other hand, if an object is detected, a message "Object detected: I know you are there!" is shown on the screen as shown in Figure 6.

5. A WSN for Detecting and Tracking

Further enhancements of the sample application are in progress, including testing its working with multiple sensor nodes and the addition of the tracking capability. Figure 7 illustrates our design of a WSN with multiple nodes for object tracking.

As shown, twelve sensor nodes will be mounted on the ceiling of the lab. The WiEye Long-range passive infrared (PIR) sensor with 90-100° wide detection cone, 20-30 feet detection range for human presence [18] are to be used for tacking a human motion in the room. TelosB are to be used as the motes; their radio frequency outdoor range is about 75 to 100 meters and indoor range is about 20 to 30m [19]. However, for accuracy and redundant coverage of the monitored area, the distances between the nodes are kept at 20 feet as shown in the figure.

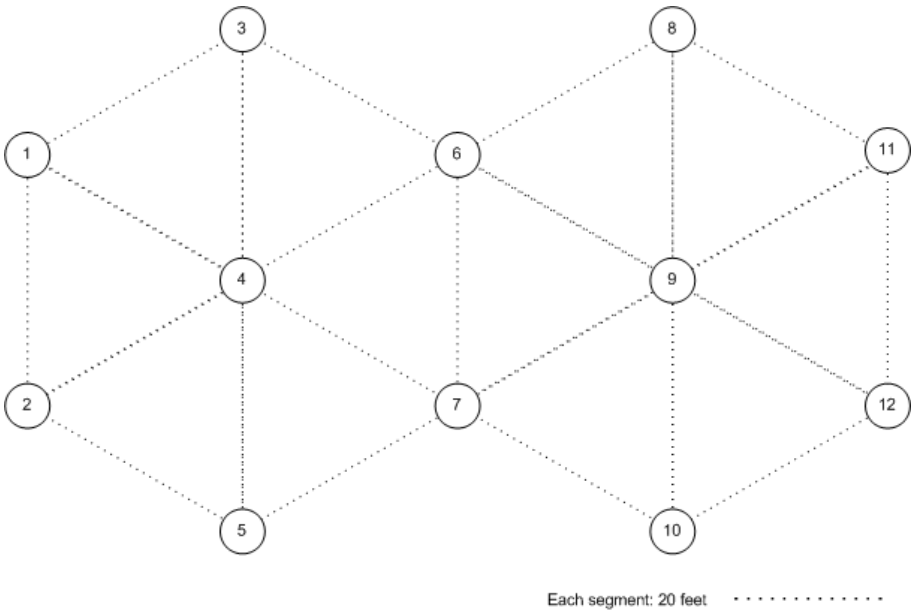

Figure 7. Design of an Object Tracking WSN Testbed

(Twelve nodes are mounted on the ceiling of the lab as adjacent equilateral triangles.)

This set of nodes will form a WSN communicating with the base station. An algorithm is to be developed for the base station such that it will use queue data structure for storing data collected from each node. Data for each node will be stored in a separate queue. When a sensor node detects data variation in the environment (i.e., the collected value is greater than the threshold value), it sends that information to the base station, and the information will be stored in the queue associated to that particular node. When all the different nodes have sent their data, their corresponding queues on the base station will be filled with data with the same (or nearly the same) timestamps. Afterwards, calculation will be done to detect the position of the detected object.

The algorithm may use averaging functions similar to those employed in [4] to calculate the position of the detected object, by using strength of the signals detected by the sensor nodes. The algorithm used in [4], however, was tested on a plane (i.e., the object and the sensors are on the same plane). In contrast, in the design illustrated in Figure 7, the object will be on the floor of the room while the sensors are on the ceiling. Therefore, in the first test of the design, the object and the sensors will be placed on the same plane (all on the floor or on the wall). Later in the second test, we will consider the case where the object is somewhere perpendicular to the plane where the sensors are located, by placing the sensor nodes on the ceiling while keeping the object on the floor.

To achieve better accuracy, the nodes in the proposed design (as illustrated in Figure 7) form symmetrical figures. For example, if an object is below the triangle formed by nodes 1, 2, and 4, calculation can be done by using the triangle. Similar calculations may be carried out by using the rectangle formed by nodes 1, 6, 7, and 2, or by using the hexagon formed by nodes 1, 3, 6, 7, 5, and 2. Complex calculations can be done with the help of rhombus, trapezium, etc.

The proposed design in Figure 7 draws strengths from the two WSN emulation projects discussed in section 3. In addition, the topology of the WSN takes into account redundant coverage of the surveilled area.

6. Conclusion and Future Work

In this paper, we have surveyed research projects related to detecting/tracking objects using wireless sensor networks. Most of the existing projects adopt the simulation approach, by using a network simulator to construct the sensor network. Fewer research projects adopt the emulation approach, which demands actual devices to be used to construct the sensor network. To illustrate the construction of a wireless sensor network application, we constructed a sample WSN application to detect the presence of a human. The application demonstrates how various hardware and software components are integrated to build a WSN application capable of detecting human presence. As exhibited by the experiments, the application is capable of accurately fulfilling its mission, indicating that the passive infrared sensor employed in the application is sufficient for the detection of human presence.

Parts of our future work include the implementation of the design illustrated in Figure 7, and the testing and evaluation of the design. The evaluation of the performance of the constructed wireless sensor network will examine features such as the delay of messaging, the accuracy of object detection, etc. The development of the tracking algorithm(s) is another significant part of our future project. Another extension of the WSN application is to detect and track both humans and vehicles, by using both magnetism and passive infrared data.

References

[1] Salatas, V. (2005). “Object tracking using wireless sensor networks”. M.S. Thesis. Naval Postgraduate School, California.

[2] Wikipedia. "Wireless Sensor Network". Retrieved September 20, 2007, from http://en.wikipedia.org/wiki/Wsn.

[3] Estrin, D., A. Sayeed, and M. Srivastava (2002). “Mobicom 2002 Tutorial - Wireless Sensor Networks". Retrieved October 10, 2007, from http://nesl.ee.ucla.edu/tutorials/mobicom02.

[4] Singh, I. (2007). “Real-time Object Tracking with Wireless Sensor Networks”. M.S. Thesis. Luleå University of Technology.

[5] Tech-q.com. "Technical FAQ". Retrieved September 21, 2007, from http://www.tech-q.com/motes.shtml.

[6] European Commission. "ANGEL: Advanced Networked embedded platform as a Gateway to Enhance quality of Life". Embedded Systems Unit – G3, Directorate General Information Society, European Commission. Retrieved September 20, 2007, from ftp://ftp.cordis.europa.eu/pub/ist/docs/dir_c/ems/angel-v1.pdf

[7] Crossbow. "Products overview of Crossbow Technology". Retrieved September 15, 2007, from http://www.xbow.com/Products/wproductsoverview.aspx.

[8] EasySen. "Products overview of EasySen". Retrieved September 15, 2007, from http://easysen.com/products.htm.

[9] MoteIV. "Products overview of MoteIV". Retrieved September 15, 2007, from www.moteiv.com/products/tmotesky.php.

[10] Wikipedia. "Dempster-Shafer theory Wiki". Retrieved September 20, 2007, from http://en.wikipedia.org/wiki/Dempster-Shafer_theory.

[11] Dewasurendra, D. A., P. H. Bauer, and K. Premaratne (2006). "Distributed evidence filtering in networked embedded systems". *Networked Embedded Sensing and Control*, ser. *Lecture Notes in Control and Information Sciences* 331: 183–198.

[12] Tsai, H., C. Chu, and T. Chen (2007). "Mobile object tracking in wireless sensor networks". *Science Direct – Computer Communications 30(8)*: 1811-1825.

[13] Lin, C. and Y. Tseng (2004). "Structures for in-network moving object tracking in wireless sensor networks". *Proceedings of the First International Conference on Broadband Networks (BROADNETS'04)*.

[14] Kung, H. T. and D. Vlah (2003). "Efficient Location Tracking Using Sensor Networks". *Proc. of 2003 IEEE Wireless Communications and Networking Conference (WCNC)*, March 2003.

[15] Aslam, J., Z. Butler, F. Constantin, V. Crespi, G. Cybenko and D. Rus (2003). "Management: Tracking a moving object with a binary sensor network". *Proceedings of the 1st international conference on Embedded networked sensor systems* (SenSys '03).

[16] He, T., P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J. A. Stankovic, and T. Abdelzaher (2006). "Achieving Real-Time Target Tracking Using Wireless Sensor Networks". *Proceedings of the Twelfth IEEE Real-Time and Embedded Technology and Applications Symposium*, 2006.

[17] Tran, S. P. M. and T. A. Yang (2006). "Evaluations of target tracking in wireless sensor networks". *Proceedings of the 37th SIGCSE technical symposium on Computer science education SIGCSE '06, ACM SIGCSE Bulletin,* 2006.

[18] EasySen. "EasySen Products - WiEye". Retrieved September 20, 2007, from http://easysen.com/WiEye.htm

[19] Crossbow. "Crossbow, TelosB, mote platform". Retrieved December 15, 2007, from http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf.

[20] EasySen. "EasySen SBT80 Datasheet". Retrieved September 15, 2007, from http://www.easysen.com/support/SBT80v2/DatasheetSBT80v2.pdf

[21] EasySen. "EasySen WiEye Datasheet". Retrieved September 15, 2007, from http://www.easysen.com/support/WiEye/DatasheetWiEye.pdf

[22] TinyOS. "TinyOS Tutorial, Lesson 6". Retrieved October 1, 2007, from http://www.tinyos.net/tinyos-1.x/doc/tutorial/lesson6.html

Appendix A: Procedure of running the application

The following is the programming sequence as outlined in the data sheet of the SBT80 sensor [20]. The sample application folder SBT80app consists of 2 header files (*SBT80ADCmap.h* and *SBT80Msg.h*), a configuration file (*MobileNode.nc*), and a module file (*MobileNodeM.nc*). It samples all 8 sensor channels in a sequence and transmits data to a PC. The file *ListenSBT80v2.java* reads and displays the packets transmitted using the application discussed above.

Step 1: Program one TmoteSky / TelosB mote with the MobileNode application given here.

(a) Copy and unzip the folder SBT80app into the /opt/tinyos1.x/apps folder in your TinyOS installation

(b) Connect TmoteSky mote to the PC's USB connector and execute the command **make tmote install** inside the folder *SBT80app* to install the application

(c) Connect the sensor board SBT80v2 to the expansion connector of the TmoteSky mote.

Step 2: Program a base station with the generic TOSBase application. Connect another TmoteSky mote to the PC's USB connector and execute the command **make Tmote install** inside the folder /opt/tinyos1.x/apps/TOSBase.

Step 3: Display sensor readings on the controlling workstation.

(a) Copy the java program ListenSBT80v2.java to the following folder in your TinyOS installation: /opt/tinyos1.x/tools/java/net/tinyos/tools.

(b) Compile by executing the command "javac ListenSBT80v2.java" inside the same folder.

(c) Set the *MOTECOMM* variable to read the USB port properly according to the TinyOS tutorial.

(d) Run the java application by executing the command "java net.tinyos.tools.ListenSBT80v2".

This should produce a display of sensor readings from all 8 channels on the Cygwin window. These steps are given to test the working of SBT80 and the same procedure is used to configure the WiEye sensors [21].

In step 3.C, in order to configure the *MOTECOM* variable, first run the "motelist" command to get the port number on which TelosB is connected. Once you get the port number, use the export command to configure the *MOTECOM* variable. The command is "*export MOTECOM=serial@COM4:telosb*" or "*export MOTECOM=serial@COM4:telos*". The number after COM is the port number which is written 4 here [22].