Emulation of Wireless Sensor Networks for Object Tracking

T. Andrew Yang Deepesh Jain

Table of Contents

Abstract

Wireless Sensor network (WSN) is a new technology, it could be a way to achieve ubiquitous computing and embedded Internet. WSN applications are efficient for deep monitoring of a deployment area [1]. This project is an object tracking application of WSN; it will detect presence and will track motion of a human. Project is a system using a JAVA application on top of Motes and Sensors provided by Crossbow Inc. and EasySen Inc respectively. Basic idea of project was to first develop an object tracking algorithm. Then to implement and test developed algorithm into laboratory. Related work has been done in past but because of their limitations I tried to come up with new algorithm discussed in this report.

1. Introduction

Before delving into the details of the project I would like to discuss the basic concepts of WSN. This will help us to understand the peculiarities of the project. A wireless sensor network (WSN) is a network made of a set of independent sensor nodes. The sensor nodes are self-contained units consisting of a battery, a radio module, sensors, and a lower speed on-board processor. A WSN system can be 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 [2][3]. Wireless sensor networks have significant impact upon the efficiency of military and civilian applications, which may be classified into three classes [2]:

1. Data collection,
2. Surveillance, and
3. 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 to be lost because of landmines. And lot more examples can be discussed regarding this application. This project is to demonstrate how a wireless sensor network may be employed to detect and track objects. I have developed an object tracking algorithm that will first detect the human presence and will start tracking its motion.

1.1 Basic requirements of WSN

Following is the hardware requirement for deployment of a wireless sensor network:

1. Motes: A mote is a low powered computer with a radio transmitter capable of forming ad hoc communication with other modes [5]. A mote may be connected to one or more sensor boards.
2. Sensor boards: A sensor board is a chip on which one or more sensors are present.
3. Gateway: A gateway is a device which is 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].
4. Monitoring workstation: The monitoring workstation is 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.

1.2 Prominent available hardware devices of WSN:

Crossbow, EasySen, MoteIV are some of the manufacturing companies that provides 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. 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. This goal could be accomplished in three steps [7][8][9]:

1. Testing of available sensors with simple applications, and selection of sensor/mote for further deployment.
2. Deployment of a WSN and testing of simple wireless sensor networks applications with the selected sensor and mote.
3. Testing of object tracking applications in the deployed wireless sensor network.

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

2. Related work in the field:

This section talks about various works done related to object tracking using WSN. First we will discuss the pros and cons of various approaches and in later section we will about the method used in this project. Following are the related work:

1. In his master thesis, Salatas 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 [1]. The thesis talks about 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.

Major issues addressed in this thesis are the evaluation and efficient use of WSN products with no changes while using in real world application. Also address efficient ways to algorithmically analyze the collected the raw data from the specific wireless sensor networks product. However the main focus is the deployment of a real world object tracking application using wireless sensor networks, it also provides idea to explore overall wireless communication [1].

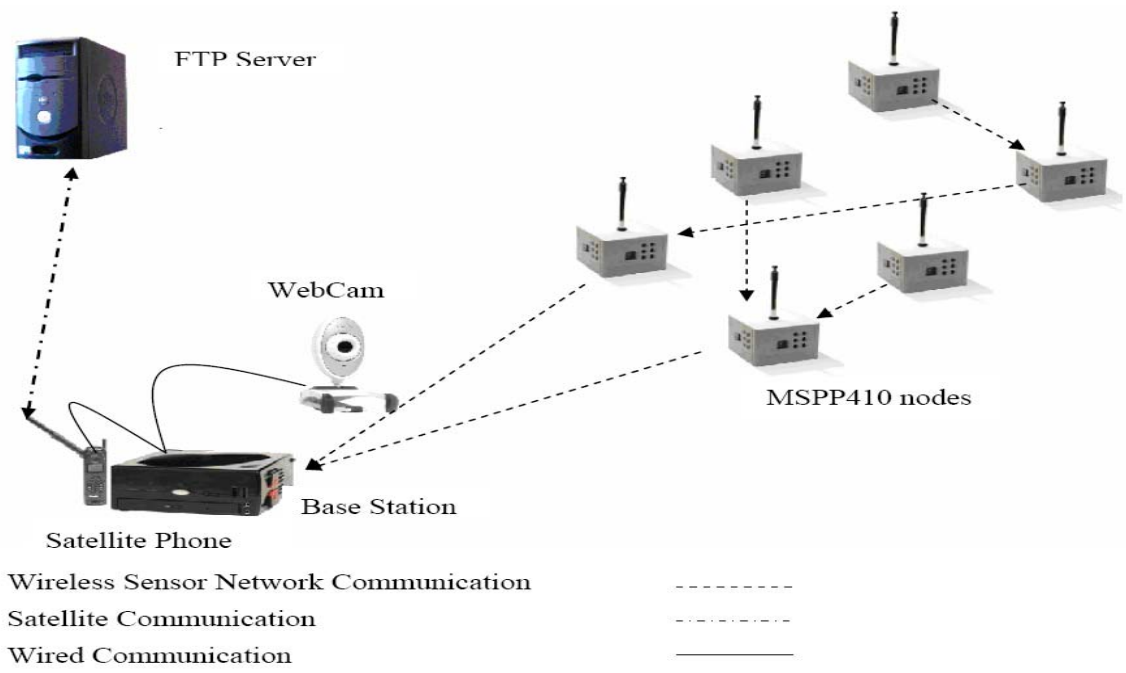


Figure 1. Overall System High-Level View [1]

Figure 1 above shows the overall system High-Level View. It shows the wireless sensor network (formed by the nodes: white colored boxes) is connected with some another system Tactical Remote Sensor System version 3 (TSSRv3). The TSSR is a subsystem here, which consists of both hardware and software parts. The TRSSv3 system currently includes three sets of a 4XEM Elite miniPC, Globalstar Phone, and Creative WebCam. In addition, the system includes one File Transfer Protocol (FTP) Server. This subsystem detects objects by performing picture comparisons. It then transmits those pictures to the control station by using satellite or cellular communications. In the overall system, the subsystem is responsible for taking photos, acting, at the proper time after the object's detection, and tracking [1].

This Object-tracking application receives and uses the raw values returned by the sensor network system to produce clear and meaningful outputs. The outputs are then easily integrated into a larger application developed in JAVA, or are used independently as output of a specific isolated application. As mentioned in the thesis this application is based on an existing Crossbow wireless sensor network product without making any additional internal changes. All data evaluation, filtering and manipulation takes place in the base station (PC) where the network's data finally arrives. Thus application supports developer's choice to deploy application by using a specific product with a minimum amount of changes. Deployment of application includes the design and the implementation phase. The design phase is an iterative process and as the most critical phase requires theoretical knowledge of wireless sensor network technology. It requires sufficient knowledge of the abilities and specifications of the wireless sensor network product. The implementation phase has some programming difficulties; here choice of programming language is Java. Java program implements precisely the algorithmic part of the design. At last, the evaluation of the Crossbow product and the object-tracking application indoor and outdoor experiments conducts are mentioned [1].

Application basically detects the human or vehicle type of objects on the basis of infrared or magnetism data sensed by the sensor. Sensors collect raw data which is processed at base station. If in consecutive values of raw data magnetism and infra red values are changed and are greater then the normal environmental magnetism and infra red values of deployed area, application infers object as a vehicle. If changed value is only infra red and not magnetism application

consider that object as a human. This application has its own limitation; it works only for a single object. It is also restricted for its deployment scenario which can be a straight road, T-Shaped road or a square i.e. cross road [1].

2. 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 on events of interest such as threats occurring with a certain temporal distribution, while accommodating the varying reliability and accuracy of information sources.

3. Tainan, Tsai, Chua, and Chen, 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/she knows the target's position. Paper proposes a binary sensor model, in which each sensor's value is converted reliably to one bit of information only whether the object is moving toward the sensor or away from the sensor.

4. Chih-Yu Lin and Yu-Chee Tseng, in their paper Structures for in-network moving object tracking in wireless sensor networks [13], is an extension of a work [14]. In [14], author considers a sensor network as a distributed database. He proposes 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. In [13], work is an extension of the work 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. Then paper proposes two message-pruning tree structures called *DAT* (*Deviation-Avoidance Tree*) and *Z-DAT* (*Zone-based DAT*).

Author has formulated communication costs associated with trees. Through simulations, he demonstrates the advantage of our approach [13].

5. Javed Aslam, Zack Butler, Florin Constantin, Valentino Crespi, George Cybenko, Daniela Rus in their paper Management: Tracking a moving object with a binary sensor network [15], have examine the role of very simple and noisy sensors for the tracking problem. 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 is toward the sensor or away from the sensor. 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 present resulting algorithms and simulation experiments. Authors develop a particle filtering style algorithm for target tracking using such minimalist 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 for their extensive simulations, which show low error that decreases with sensor density [15].

6. Sam Phu Manh Tran and T. Andrew Yang, in a paper Evaluations of target tracking in wireless sensor networks [17], evaluates different algorithms for target tracking in wireless sensor networks. Since optimized computation and energy dissipation are critical requirements to maximize the lifetime of the wireless sensor network. There exists a demand for self-organizing and routing capabilities in the algorithm of wireless sensor network. Existing methods attempts to achieve these requirements, like LEACH based algorithms, nevertheless, suffer redundancy in data or sensor node deployment, or complex computation incurred in the sensor nodes. These drawbacks tend to result inefficient use of energy and/or complex computation overhead [17].

Authors' previous work Optimized Communication and Organization (OCO) 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. To evaluate the performance simulation models used by the authors consist of three sub modules: a sensor-node model, a sensor-network model, and an intruder-object model. In [17] these models are explained as follows: "The sensor-node model is

used to simulate a sensor node. It includes three layers: application, MAC, and physical layer. It contains several modules, including the coordinator module, the radio module, the sensor module, and the energy module. The sensor-network model contains a set of sensor nodes, and is used to represent the sensor network. Two nodes can communicate with each other if the distance between them is smaller than their communication radius. The intruder-object model represents the intruding object. Although being similar to the sensor node, an object has only two layers: the application layer on top of the physical layer."

7. Another work in emulation of WSN is a master thesis "Real-time Object Tracking with Wireless Sensor Networks" by Inderjit Singh [18]. Author has developed a dynamic system with WSN that uses the Time Division Multiple Access (TDMA) with Extended Adaptive Slot Assignment Protocol (EASAP). Thesis shows that EASAP is a very suitable protocol that can be used in a dynamic WSN. The complete functionality of the system is not implemented. Nevertheless, continuous work has been suggested with the multi hop technique for passing data throughout the network. 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. X-Configuration is shown as shown in figure2.

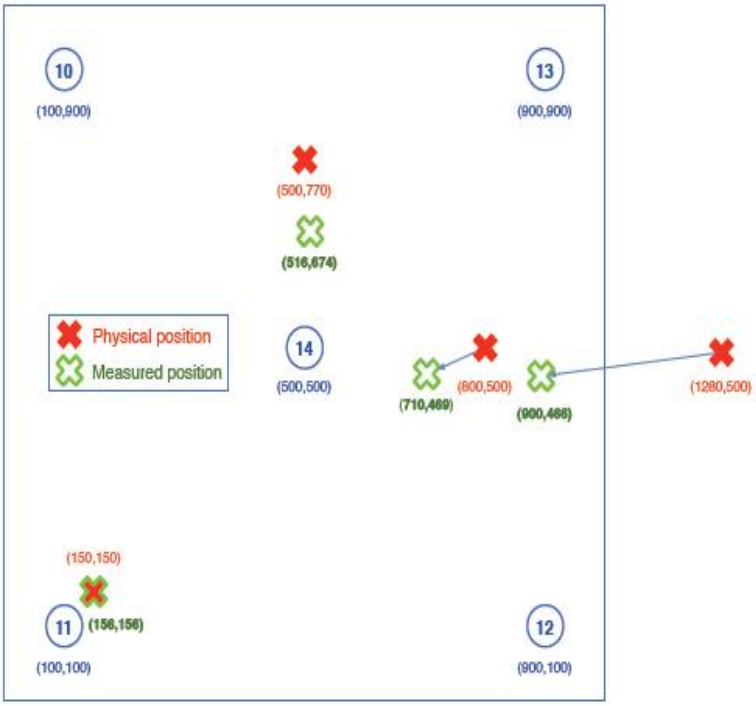


Figure 2. X-Configuration

Source in the demonstration is light generated by a lamp. Source localization algorithm uses averaging of signals strength detected by sensors and is proven to be well suited for this application. As described briefly in the thesis localizing algorithm with the help of three nodes as

shown in figure 3. Three 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 by the averaging algorithm will result to:

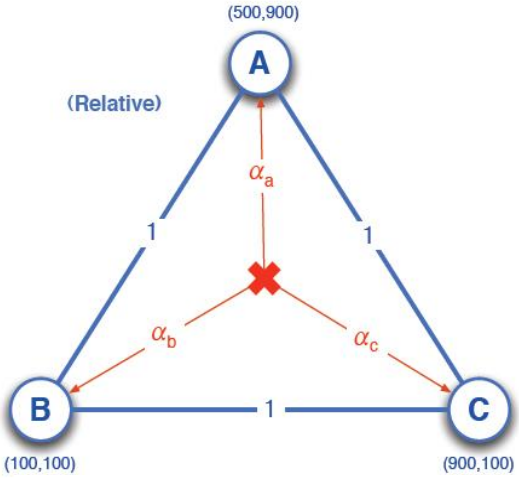


Figure 3. Source localization example with three nodes

$x = (500.\alpha_a+100.\alpha_b+900.\alpha_c)/ (\alpha_a+\alpha_b+\alpha_c)$ for the ***x*** position and $y = (900.\alpha_a+100.\alpha_b+100.\alpha_c)/ (\alpha_a+\alpha_b+\alpha_c)$ for the ***y*** position.

However, there was a resulted failure in accuracy while using this algorithm. Failure was showed to be mainly the lack of a proper calibration to the source that was overlooked during the test procedure. This project is tested for maximum number of six nodes and results show some failure in measured position of object compared to actual physical position of object.

However basic idea for calculating coordinates of object in the deployment in my algorithm is nearly same as mentioned in this piece of work. Tracing the complex motion of human in the deployment area is somewhat typical, which will become little bit simpler when nodes will be deployed symmetrically.

3. Description of project and methodology used

This project is build to track the motion of a human when he/she comes in the vicinity of sensor used in the system. To recognize a human I had developed a detection application which runs on the base station shown in figure. Application collects the data from the network and analyzes the collected data to detect the presence of human. PIR sensor is used to collect the data from the environment. Data is collected and from the environment initially and a threshold value is set when no human is present. Later if human comes in the vicinity of sensor it compares the current collected value from the network with threshold value and makes decision of human presence.

Once a human is detected tracking algorithm starts, in parallel nodes keep sending raw data to the base station. At base station collected data is stored in a queue data structure. There is a queue for each node present in the network. Data in the queue is stored with according to the timestamp. Timestamp for fetched data is assigned on the base station when base station application receives that particular data. Application checks the node-id, i.e which node has sends this particular piece of data, and stores the data in the corresponding queue. This fetching and storing data keeps going on until user interrupts or object is no more present in the observation area.

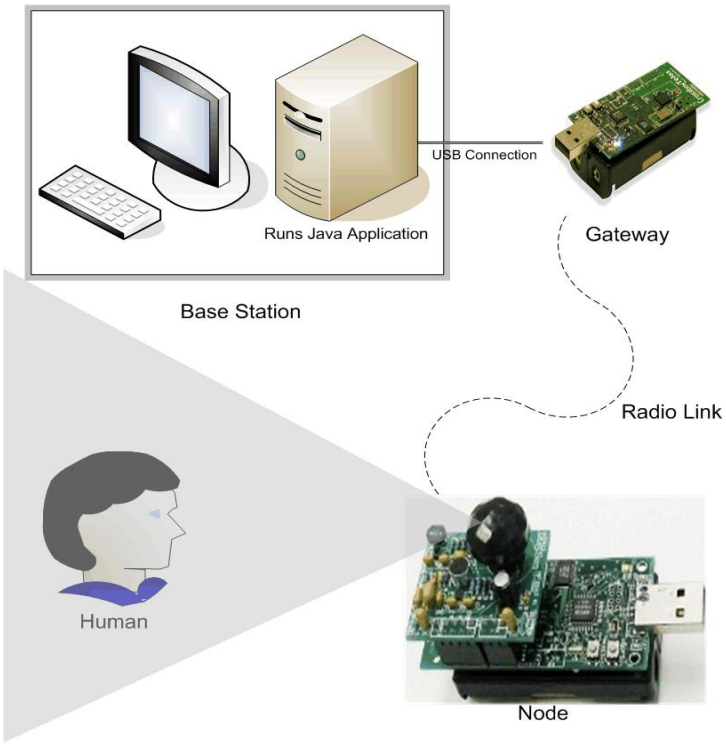


Figure 4. Design of a WSN Detecting Human Presence

Tracking algorithm uses the fact that three nodes are required to determine a 2-dimensional position of an object, as in GPS three satellites are required to determine a 2-dimensional position of an object [16]. Once object presence is detected it looks for the data in queues. Data is retrieved from each queue for the n^{th}, $n\text{-}1^{th}$, and $n\text{-}2^{nd}$ instance of time. For each node this three data value is averaged, now these averaged values are sort in the descending order. Top three are picked up, For example if node i, j and k has the highest value of data and received signal values are α_i, α_j and α_k. Then the coordinate of the object will be calculated accruing to the formula:

$$X = \frac{(x_i * \alpha_i) + (x_j * \alpha_j) + (x_k * \alpha_k)}{(\alpha_i + \alpha_j + \alpha_k)}$$

$$Y = \frac{(y_i * \alpha_i) + (y_j * \alpha_j) + (y_k * \alpha_k)}{(\alpha_i + \alpha_j + \alpha_k)}$$

Data Design

1. Boolean tracking_enabled;
   // initialized by false;

2. Queue to store data received from nodes.
   // It stores data received along with timestamp values.
   // Timestamp is a time value set by the sensor node, which sends the sensed data to the base station.

3. int n // keep tracks of currently active number of nodes in network.

4. int threshold[5]; // PIR values set to be threshold, top five sensed value will be stored in this array.

5. int window; // variable used to take the number of data values to be averaged.
   // for example if window = 3 and t represents the current time; the recently sensed values at times t, t-1, and t-2 will be taken in consideration when calculating the sensed value

6. Class Node
   Members:
   int nodeId;
   int xCord; //X Coordinate of the node
   int yCord; // Y Coordinate of the node
   Queue dataQueue; // To store the signals received by the node along with the timestamp.

7. HashTable: collectionOfNodes;
   // It will be used to store the node object with the nodeId as key.

8. Integer Array: sortingArray[n][2];
   // Each row of this two dimensional array will consist of nodeId and averaged data value.

9. Integer Array: resultArray[n][4];
   // Each row of this two dimensional array will consist of:
   1.1. nodeId
   1.2. Difference of physical and calculated X-Coordinate of object at time t.
   1.3. Difference of physical and calculated Y-Coordinate of object at time t.
   1.4. Value of timestamp at that time t.

ObjectTracking algorithm

Inputs:

a. **Node id's and their coordinates**

b. **Data value from the network (consisting of nodeId and strength of signal)**

Output:

a. Calculated coordinates of object (X,Y)

b. Time t for which (X,Y) is calculated

1. Set *tracking_enabled* = false
2. Read the Node id's and their coordinates

   // Create the object and store that into hash table with key value
3. Set the *threshold* after T seconds

   // Keep fetching the values from the network and set the average of highest five values received to the threshold array.
4. Fetch data from the network
5. If data value > *threshold* (for any nodeId)

   Got to step 4

   else go to step 6.
6. Set *tracking_enabled* = true

   6.1. Create a new child process (the *tracking* process) to do the following:

   6.1.1. Call the **tracking** sub-procedure.

   6.1.2. Display the result.

   6.1.3. If user wants to stop tracking,

   6.1.4. Set *tracking_enabled* = false and stop the *tracking* child process.

   6.1.5. else go to step 6.1.1.

   6.2. Continue the main process in parallel with the *tracking* child process:

   6.2.1. Fetch data from network

   6.2.1.1. Retrieve the *node* object from *nodeCollections* corresponding to the nodeId for which data is received.

   6.2.1.2. Store the received value with the timestamp into the *dataQueue* of the *node* object.

   6.2.2. If tracking_enabled is = true

   go to step 6.2.1.

else stop.

Tracking sub-procedure

Input: Data from the queue for t^{th}, $(t-1)^{th}$, $(t-2)^{th}$, …, $(t-j)^{th}$ second, where *j* = the size of the scanning window)

Output:

a. Calculated coordinates of the intruder object (X,Y)

b. Time t represents the point of time at which (X,Y) is calculated.

1. Set *i* to be the size of the scanning window.
2. Use Enumeration to retrieve node object from the *nodeCollection*.
   2.1. Retrieve *i* data values (strengths of the received signals) stored in the *dataQueue*.
   2.2. Average those values and put it into the *sortArray* indexed with the *nodeId*.
3. Sort (using quick sort) data values of *sortArray* in descending order of signals received.
4. Pick the top three values from the *sortArray*.
5. Calculate the coordinates of the object using the proper formula[2] according to number of nodes.
6. Store the nodeId, calculated X and Y value and also timestamp (average value of $t-2^{nd}$, $t-1^{st}$ and t^{th}) into *resultArray*.
7. Call the ***AccuracyAnalysis*** sub-procedure.
8. Return

The AccuracyAnalysis sub-procedure:

Input: *resultArray,* actual trajectory of object X-coordinates and Y-Coordinates with time t.

Output: X-difference, Y-difference with time t.

[2] Note: Formula to calculate the coordinates is same as calculating the Magnetic field or electric field generated by sources of different strength. Where xi and yi are the coordinates of ith node and αi is the strength of the signal calculated by that node.

$$X = (x1 * \alpha1) + (x2 * \alpha2) + (x3 * \alpha3) \,/\, (\alpha1 + \alpha2 + \alpha3)$$

$$Y = (y1 * \alpha1) + (y2 * \alpha2) + (y3 * \alpha3) \,/\, (\alpha1 + \alpha2 + \alpha3)$$

1. Calculate the difference between the calculated coordinates of the object and the physical[3] position of the object.
2. Store the nodeId, X-difference, Y-difference, and timestamp (average value of t-2nd, t-1st and tth) into accuracyArray.

3.1 Design of the system:

To recognize a human and track his/her motion a Java application runs on the base station. Application collects the data from the network and analyzes the collected data to detect the presence of human. A gateway is connected to the base station which is a TelosB mote from Crossbow Inc. Gateway collects the data from the network. Node in the network consists of a mote powered by battery and a WiEye sensor board from EasySen Inc. is mounted on the mote.

3.2 Hardware Used:

a. Mote: TelosB(TPR2400) mote is used manufactured by Crossbow Inc, it has USB support [7].
b. Sensor Board: WiEye sensor board used manufactured by 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]. Sensor board is mounted on a TelosB mote as shown in figure2.
c. Gateway: No special gate way is used. TelosB is connected on USB port of the base station to collect the data from the network as shown in figure2.
d. Base Station: is a normal Desktop Computer on which Cygwin and Java virtual machine is installed.

3.3 Software Used:

a. Cygwin: Cygwin is used to configure the WSN. Applications in motes are installed through this command based software. Base station application is also invoked from this software.
b. Programmers Notepad: To write, compile and debug the Nesc program. NesC program is installed in the motes to collect the data from the environment.
c. Java SDK 1.6.0: This is used to develop a Java application which runs on the base station. This java application collects the data from the network, process it and send the output to the screen of base station.

[3] For test purpose we will be having records of object position with respect to time.

4. Experiments conducted:

This section describes the experiment conducted on the implemented algorithm in the laboratory:

a. Evaluation experiments: Several experiments were conducted in lab to test the working of application. Currently algorithm is tested with sample data generated by application another program which simulates the behavior of network. Application is able to display to locate the coordinates of human in the deployment area and thus successfully tracks his/her motion. Application is also able to track the motion of a human in light and dark both. It's independent of light present in the environment. Still I have to test the algorithm on actual network because of limited number of device in lab this task is incomplete.

b. Anticipated evaluation results: If human is not present in the vicinity of sensor a message nothing will show up. If an object is detected sub process tracking-algorithm will be invoked and will display the trajectory of object in the deployment area.

5. Conclusion and Future Work

Project is accomplished to detect the tracking of a human motion. This project is tested with a sample data generated by program which simulates the network.

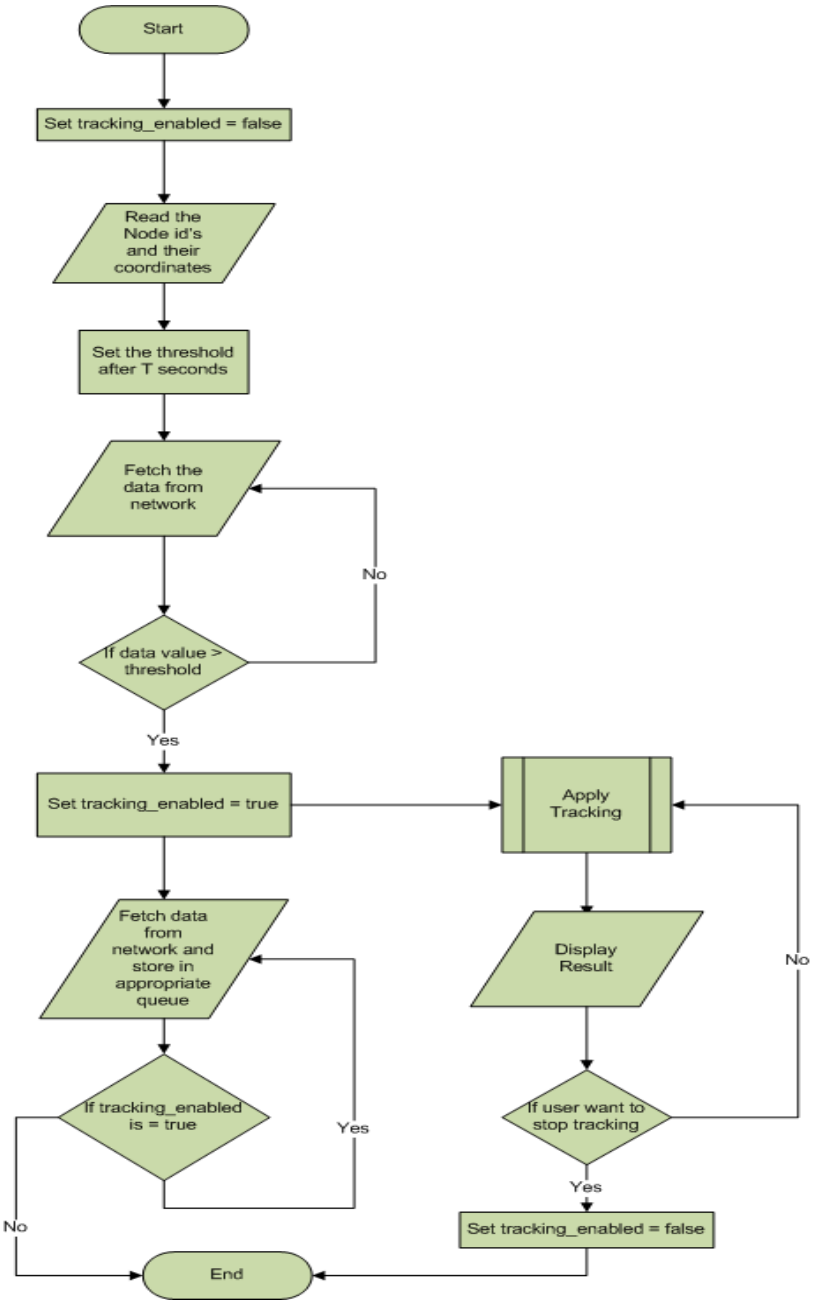
References:

[1] Salatas, V. (2005). "Object tracking using wireless sensor networks". M.S. Thesis. Naval Postgraduate School, California.
[2] Wireless Sensor Network from Wikipedia, http://en.wikipedia.org/wiki/Wsn (September 20, 2007)
[3] 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).
[4] Tech-q.com. "Technical FAQ". Retrieved September 21, 2007, from http://www.tech-q.com/motes.shtml.
[5] 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 Emulation of Wireless Sensor Networks for Object Tracking (DRAFT) 9/11
[6] Products overview of Crossbow Technology on xbow.com, http://www.xbow.com/Products/wproductsoverview.aspx (September 15, 2007)
[7] Crossbow. "Products overview of Crossbow Technology". Retrieved September 15, 2007, from http://www.xbow.com/Products/wproductsoverview.aspx.

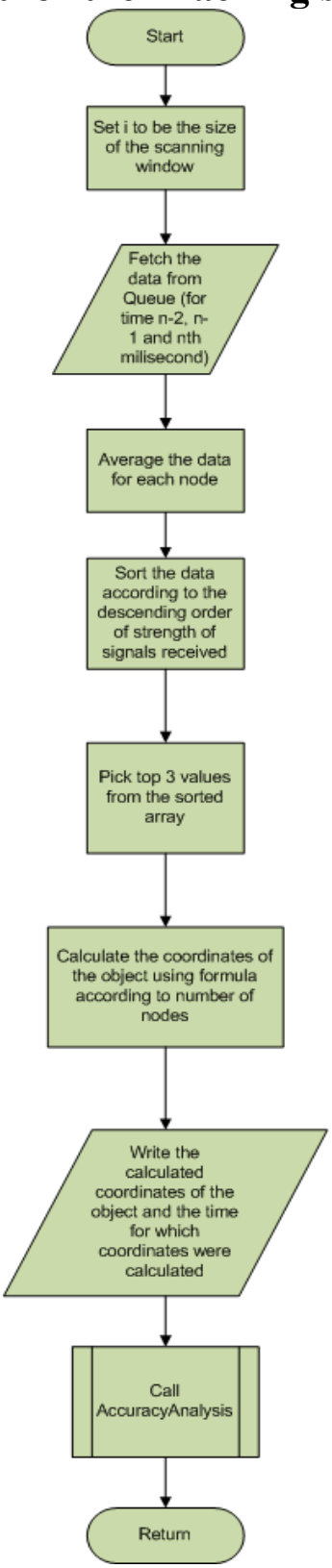
[8] MoteIV. "Products overview of MoteIV". Retrieved September 15, 2007, from www.moteiv.com/products/tmotesky.php.
[9] Wikipedia. "Dempster-Shafer theory Wiki". Retrieved September 20, 2007, from http://en.wikipedia.org/wiki/Dempster-Shafer_theory.
[10] Dewasurendra, D. A., P. H. Bauer, and K. Premaratne (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] Rappaport T.S., Reed, J.H. and Woerner, B.D., Position location using wireless communications on highways of the future, Communications Magazine, IEEE Volume 34, Issue 10.
[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] Singh, I. (2007). "Real-time Object Tracking with Wireless Sensor Networks". M.S. Thesis. Luleå University of Technology.

Appendix A

Flow chart for object tracking algorithm:


Start
Set tracking_enabled = false
Read the Node id's and their coordinates
Set the threshold after T seconds
Fetch the data from network
No
If data value > threshold
Yes
Set tracking_enabled = true
Apply Tracking
Fetch data from network and store in appropriate queue
Display Result
No
If tracking_enabled is = true
Yes
If user want to stop tracking
Yes
No
Set tracking_enabled = false
End

Flow chart for the Tracking sub-procedure:



Note: Formula to calculate the coordinates is same as calculating the Magnetic field or electric field generated by sources of different strength.

Here formula will be:

$$X = (x1 * \alpha1) + (x2 * \alpha2) + (x3 * \alpha3) / (\alpha1 + \alpha2 + \alpha3)$$

$$Y = (y1 * \alpha1) + (y2 * \alpha2) + (y3 * \alpha3) / (\alpha1 + \alpha2 + \alpha3)$$

Where xi and yi are the coordinates of ith node and αi is the strength of the signal calculated by that node.

Flow chart for the AccuracyAnalysis sub-procedure: