# Continuous Contour Mapping in Sensor Networks

Cheng Zhong
National Center for Geographic
Information and Analysis
University of Maine, Orono, ME, 04469, USA
Email: cheng.zhong@umit.maine.edu

Michael Worboys
National Center for Geographic
Information and Analysis
University of Maine, Orono, ME, 04469, USA
Email: worboys@spatial.maine.edu

***Abstract*—A contour map is a useful data representation schema that can be used for monitoring tasks involving wireless sensor networks [1]. This paper reports the development of an algorithm (CCM) for continuous contour mapping that is more energy-efficient than straightforward spatial or temporal suppression algorithms. A subset of contour nodes is chosen to report to the sink, such that no significant contour information is lost. Experiments are reported showing that, depending on node density, our algorithm provides an average reduction of 60% on the total amount of report data, compared to the baseline spatial suppression algorithm. This reduction is achieved while at the same time maintaining accuracy with respect to the monitored source.**

## I. Introduction

Wireless sensor networks (WSNs) can be used to monitor environmental events, such as mud flows and forest fires. Events are usually located in geographic regions, and in most applications it is not practical to continuously collect data from every node due to energy constraints. Contour maps provide a good tradeoff for monitoring tasks.

A contour exists between two adjacent nodes if they are not in the same value range. A node can locally detect whether there exists a contour between it and its one-hop neighbors simply by comparing its reading with neighbors. If a node can detect a contour, we call it contour node. Its one-hop neighbors on the other side of a contour are called contour neighbors of the node.

The straightforward data collection approach is to let all nodes report their IDs and readings periodically. However, sensors are battery powered, and this approach will drain batteries quickly. Because the primary use of energy in WSNs is for node communication [2], in this paper, the focus is to save network energy by reducing data transmission without significantly decreasing contour mapping precision.

Silberstein et al. [3] summarize the definitions of spatial and temporal suppression and their possible combination:

- **Spatial suppression:** A node suppresses its reading if it is identical to those of its neighboring nodes.
- **Temporal suppression:** If a node's reading is not changed since the last transmission, it does not have to report to the sink. The sink can use its previous reading as the current reading.
- **Spatial-temporal suppression:** It is possible to combine spatial and temporal suppression together. If readings do not change, they should not be reported. In addition, if they do change, but the relationship between neighboring nodes remains the same, some reports may be suppressed.

In the case of continuous contour mapping, one possible combination of spatial and temporal suppression is that we only collect contour node information for reporting. Such information is sufficient for the sink to reconstruct contours later.

This paper presents a localized algorithm for contour mapping in WSNs. The algorithm only chooses a few contour nodes to report, and each reporting node collects information from its one-hop neighbors that may then be suppressed. The paper also introduces a data structure that substantially reduces report message size. These approaches help to achieve energy-efficient contour mapping in WSNs.

In summary, our main contributions in this paper are:

- Design of a small bit array to store one-hop neighbor information of a node. If a node reports to the sink, the message contains all one-hop neighbor information by using this structure.
- A localized **C**ontinuous **C**ontour **M**apping (CCM) algorithm which utilizes both spatial and temporal suppression techniques. A few reporting contour nodes are locally selected to report in each sampling round.
- Comparison of our algorithm with existing ones, including the spatial, Isolines [4], and temporal algorithms. The evaluation results demonstrate that our algorithm sends much less data without losing mapping precision significantly.

The remainder of this paper is organized as follows. We summarize related work in section II and give related definitions and describe a straightforward approach in section III. In section IV, we present our algorithm for continuous contour mapping. The algorithm evaluations are discussed in section V. Finally, we conclude this paper.

## II. Related Work

Energy-efficient contour detecting and reporting algorithms provide the subject of much research in the sensor network field. Localized boundary detection are discussed by Chintalapudi and Govindan [5] and Ding et al. [6], but this work does not consider how to transmit the detection results back to the sink. Much of related work takes advantage of in-network data aggregation to achieve energy saving. Spatial

1-4244-1457-1/08/$25.00 © IEEE
*This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2008 proceedings.*

suppression and/or temporal suppression [7], [4], [1], [3], [8], [9] are both approaches to data aggregation. Their advantages and disadvantages are discussed in [10].

## III. Preliminaries

In this section, we present the definitions that will be used in this paper and describe a straightforward contour detection approach that is the foundation of our work. The assumption here is that each sensor node knows its own location.

### A. Contour Neighbor Array

We denote the reading at node $u$ by $R(u)$ and its value range by $Range(u)$. For any two nodes $u$ and $v$, if $u$ and $v$ are in the same predefined value range, we have $Range(u) = Range(v)$, otherwise, $Range(u) \neq Range(v)$ (either $Range(u) > Range(v)$ or $Range(u) < Range(v)$ ).

*Definition 1 (Contour Neighbor Array (CN-array)):* Let $u$ be a node and $v_1, v_2, ..., v_n$ be the one-hop neighbors of $u$, sequenced in counterclockwise cyclic order around $u$, where the start node $v_1$ is randomly assigned in advance. The *CN-array* associated with $u$ is an array of bits $[b_1, b_2, ..., b_n, h]$ where

$$b_i = \begin{cases} 0, & \text{if } Range(u) = Range(v_i) \\ 1, & \text{if } Range(u) \neq Range(v_i) \end{cases}$$

for $1 \leq i \leq n$.

$h$ is set to 1 if all contour neighbors of $u$ are in a higher value range than $u$. Otherwise, if all contour neighbors of $u$ are in a lower value range than $u$, $h$ is set to 0.

From the value of $h$, the sink can know the contour value (higher or lower) between a reporting node and its contour neighbors. For example, suppose the contour values are defined by scale 10 (40, 50 etc.). If $R(u)$ = 47 and $h$ of $u$ is 1, then $u$ detects a contour with value 50. Figure 1 is a contour node representation with its corresponding CN-array. We should point out, in some situations, contours are dense and it is possible that some neighbors of a node are in a higher value range than the node and others are in a lower value range than the node. The CN-array alone cannot deal with such cases correctly. We discuss the solution to this problem in [10].

Each node in the sensor network maintains a CN-array structure and updates it after receiving neighboring node broadcasts. If a node is chosen to report, the CN-array will be included in the report message. The sink will decode the CN-array and interpolate values to the node's neighbors after it receives the report.

*Definition 2 (Neighbor ID Array (NI-array)):* Let $u$ be a node and $v_1, v_2, ..., v_n$ be the one-hop neighbors of $u$, sequenced in counterclockwise cyclic order around $u$, where the start node $v_1$ is the same as the one for $u$'s CN-array. The *NI-array* associated with $u$ is an array of node IDs $[ID_1, ID_2, ..., ID_n]$ where $ID_i$ is the ID of $v_i$ for $1 \leq i \leq n$.

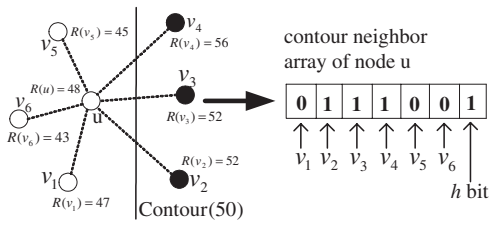


Fig. 1. Contour node $u$ reports the bit array shown, where $v_1$ is the start node. The sink knows the start node and the cyclic order, and so knows that $v_2, v_3, v_4$ are contour neighbors of $u$ and they are in higher value ranges.

### B. A Straightforward Approach

Before presenting our CCM method, we first give an approach which shows the idea of using reporting nodes, and will be used later for comparison purposes.

We can select a few contour nodes to report and suppress their neighbors. If there is a contour between two nodes, their locations and readings are enough for the contour reconstruction. That means a reporting node can add its contour neighbor ID and reading for reporting. This approach is implemented in Isolines [4]. On average, half of the contour nodes will report.

We optimized the Isolines approach by letting each reporting message contains all contour neighbor ID-reading pairs, rather than just one pair. This optimized Isolines approach will be used as one of our evaluation baselines. The ID and reading of a node will usually be at least both 2 bytes, which means that this strategy will also return large amounts of data, especially in a dense sensor network. The CCM algorithm presented in the next section is designed to utilize fewer reporting nodes, and therefore reduce data transmission.

## IV. Continuous Contour Mappings in Sensor Networks

In this section, we introduce the CCM algorithm for contour mapping. Algorithm optimization and non-reporting contour node interpolation are also discussed.

### A. Network Initialization and Query Distribution

Following the convention of TAG [11], after the user inserts a contour mapping query into the WSN at the sink, the sink broadcasts the query on its radio. All nodes that hear the query process it and rebroadcast it on to their neighbors. They keep on rebroadcasting until all nodes in the network have heard the query. The query message will include sampling time and value range setting information. In this process, a data routing tree is built.

Each node randomly chooses a neighboring node as its start node and initializes its CN-array by communicating with its neighbors. Then each node sends its own ID, location, CN-array and corresponding NI-array back along the data routing tree. In this way, the sink knows the whole network topology and the mapping information between each node's CN-array and neighbors.

There are two types of reporting messages. The first type contains the reporting node's ID, reading, and CN-array. The second type only contains the reporting node's ID.

*This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2008 proceedings.*

### B. CCM Algorithm

If a node is chosen to report, it will broadcast a 'report sent' message to its neighbors telling them: "I have sent your information and you have to do nothing in this round". In this way, only some of the contour nodes are required to report to the sink. The pseudocode for the CCM algorithm is given as algorithm 1. Every node $u$ executes this code in each sampling round. The final report sent to the sink by a reporting node includes the node's ID, reading and CN-array. In each step, node $u$ only negotiates with its one-hop neighbors. So the algorithm is completely localized.

**Algorithm 1** CCM

```
1: while (1) do
2:   if (u has a changed value range) then
3:     u broadcasts its reading and ID;
4:   end if
5:   if (u received readings from neighbors) then
6:     u updates CN-array;
7:   end if
8:   if (a sampling time comes) then
9:     if (u is not a contour node) then
10:      u skips the following steps;
11:    end if
12:    u always listens; if u receives any 'report sent message' in this round, u skips the following steps;
13:    if (CN-array and value range are not changed) then
14:      u broadcasts a 'report sent message';
15:      u skips the following steps and only sends its ID back to the sink;
16:    end if
17:    u randomize a timer t1;
18:    After t1 is time-out, u broadcasts a 'report sent message' and reports to the sink;
19:  end if
20: end while
```

The CCM algorithm is somewhat similar to the algorithm mentioned in Isolines [4]. The major difference is the use of the CN-array: In the CCM algorithm, the CN-array keeps the reporting message at a very small size. But each reporting node in Isolines returns contour neighbors' readings and IDs, which need considerable data size to represent. Moreover, in Isolines, a reporting node can only suppress its contour neighbors. But a reporting node chosen by the CCM algorithm can suppress all contour nodes around it (Figure 2).

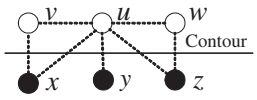


Fig. 2. Suppose node $u$, $v$, $w$, $x$, $y$ and $z$ detected a contour and $u$ is a reporting node. In Isolines [4], $u$ can only suppress $x$, $y$ and $z$. But by the CCM algorithm, $u$ can suppress $x$, $y$, $z$, $v$ and $w$ (all $u$'s neighbors)

### C. CCM Optimization

We optimize the CCM algorithm by a greedy approach. If all nodes are deployed in the sensor network with a uniform probability, then the more contour neighbors a node has, the higher is the probability that this node is near a contour. So, on average, this node will have more neighbors that are contour nodes. This gives a clue to our strategy: let contour nodes with large contour neighbor counts report and suppress their neighbors first. In the implementation process, we only need to randomize timers with short time-outs for such nodes. For example, we randomize a timer ranging from 0-t for contour nodes with more than 3 contour neighbors, and randomize timers ranging from t-2t for other contour nodes.

Figure 4 gives an example of reporting nodes chosen by the optimized CCM algorithm for a contour map snapshot. There are 165 contour nodes (dots and squares) along 2 contours. Only 37 nodes (squares) will report to the sink, and other contour nodes are all suppressed.

When contours change continuously, the CCM algorithm provides additional energy savings. If a contour node's value range and CN-array is not changed since the last transmission, it only needs to transmit its ID back. In this case, the sink will use its previous reading and CN-array as current values.

### D. Interpolation at the Sink

After receiving reports from reporting nodes in each sampling round, the sink will interpolate readings for reporting nodes' neighbors which include all non-reporting contour nodes and some non-contour nodes.

Consider a non-reporting contour node $u$ whose reading $R(u)$ is unknown. Let $S$ be the set which contains all reporting nodes which have sent reporting messages to the sink successfully. Let $S_{u1}$ be a subset of $S$ such that each node in $S_{u1}$ is a neighbor of $u$ and in the same value range as $u$. Let $S_{u2}$ be a subset of $S$ such that each node in $S_{u2}$ be a neighbor of $u$ and in a different value range. $C(u, v)$ denotes the contour value between node $u$ and its contour neighbor $v$. Then $R(u)$ is calculated by equation 1:

$$R(u) = \frac{\sum_{s_i \in S_{u1}} R(s_i) + \sum_{s_j \in S_{u2}} 2 * C(u, s_j) - R(s_j)}{|S_{u1}| + |S_{u2}|} \quad (1)$$

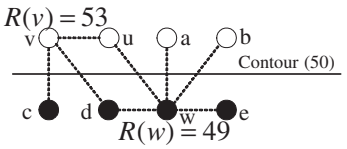


Fig. 3. Reporting nodes $v$ and $w$ of contour 50

For example, in figure 3, node $v$ and node $w$ are reporting nodes. $R(v)$ and $R(w)$ are 53 and 49 respectively. The contour value is 50. then the reading of node $u$ is calculated as $\frac{R(v)+(2*50-R(w))}{2} = 52$.

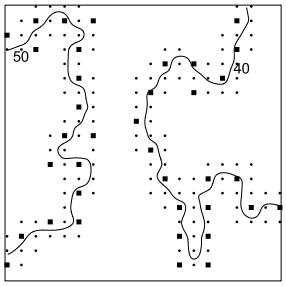


Fig. 4. Reporting nodes of CCM

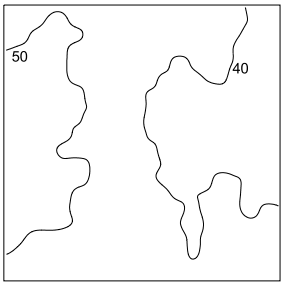


Fig. 5. Baseline contour map

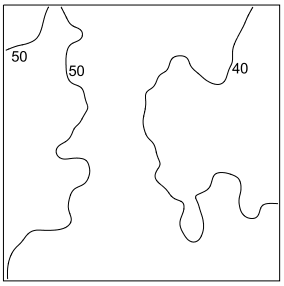


Fig. 6. Contour map of Isolines

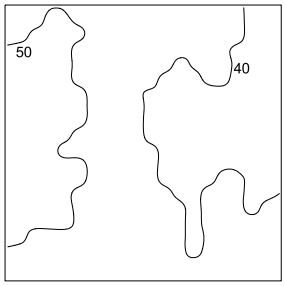


Fig. 7. Contour map of CMM

| | Similarity | Data sent (Bytes) |
|---|---|---|
| Spatial | 97.89% (sd 0.48%) | 12389 (sd 115) |
| Isolines | 97.76% (sd 0.56%) | 9472 (sd 110) |
| CCM* | 96.26% (sd 0.39%) | 6077 (sd 54) |
| CCM | 96.34% (sd 0.36%) | 5053 (sd 47) |

TABLE I
CONTOUR MAP SIMILARITY

| | Similarity | Data sent (Bytes) |
|---|---|---|
| Spatial | 93.1% (sd 2%) | 11008 (sd 379) |
| Isolines | 92.58% (sd 1.93%) | 8128 (sd 327) |
| CCM* | 95.39% (sd 0.69%) | 5336 (sd 322) |
| CCM | 94.59% (sd 0.93%) | 4527 (sd 169) |

TABLE II
CONTOUR MAP SIMILARITY IN HIGH LOSS RATE

## V. EVALUATIONS

We evaluated our algorithm using TOSSIM [12]. In the following experiments, node ID and reading are both set as two bytes long, and two bytes are used to represent CN-array. Contour scale is set by 10. The sensor field is a 400m*400m grid and the node communication radius is 30m. 400 nodes are evenly deployed, except in the second case where we change the network density. The sink is placed in the center of the field. All simulations are run 10 times for average results.

CCM* denotes the CCM algorithm without optimization and CCM denotes the optimized case. We have three baseline algorithms: (1) the spatial suppression (spatial) algorithm which only allows nodes that detect contours to send their own IDs and readings back to the sink; (2) the temporal suppression (temporal) algorithm which lets nodes send their IDs and readings back if their reading ranges are changed since the last transmission. This is only used when we consider the continuous mapping case. (3) The optimized Isolines algorithm (Isolines) in which each reporting node sends its ID, reading and the array of contour neighbor ID-reading pairs.

### A. Contour Mapping Precision

In order to know how accurate the resulting maps are, we take data from a specific sampling round and generate contour map snapshots. The contour similarity is calculated as the percentage of points (80*50 points are placed) that are actually in the correct value ranges when compare to the baseline map.

Figure 5 shows the baseline map generated by all node readings using ArcView GIS. Figure 6 and 7 are example maps generated by reporting data of the Isolines and CMM algorithms respectively. Table I gives the comparison results.

As we can see from the table I, all 4 methods generate contour maps that are highly similar to the baseline map. Because the optimized Isolines algorithm and the spatial suppression are both designed to send all exact contour node readings, they generate slightly better maps than the CCM* and CCM algorithms. But the spatial algorithm sends much more data than all other algorithms. The CCM algorithm sends the least data.

We also evaluate how different algorithms work by introducing high packet losses. In this experiment, the average loss rate of reports is 40%, which means 40% of reports will be dropped before they are received by the sink. Table II shows the results. From the table, all methods still produce good maps. The possible reason is that the network with 400 nodes is dense, and so packet loss does not influence the final result greatly. Furthermore, the maps generated by the CCM* and CCM algorithm are slightly better than the spatial and Isoline algorithms. The reason is that many contour nodes are reported by more than one reporting node, which means a non-reporting contour node can get an interpolated value unless all reporting messages containing this node information cannot reach the sink.

### B. The Impact of Node Densities

We also evaluate the impact of node densities on the total data transmitted. The same contour map data as the above evaluation is used as the input. In order to change node densities, numbers of nodes are made to vary from 300 to 600 and nodes are randomly deployed.

Figure 8 shows that, as the average neighbor number increases, the spatial algorithm sends more and more data to the sink because more nodes detect contours. But data volumes sent by the CCM* and optimized CCM algorithms increase slowly. The Isolines algorithm performs better than the spatial suppression algorithm but worse than the CCM* and CCM algorithms. Figure 9 shows the total number of reporting nodes of different algorithms for different densities. We can see that the CCM* and CCM algorithms will only select a few nodes to report in both sparse and dense networks. In short, the less reporting nodes and the use of CN-arrays

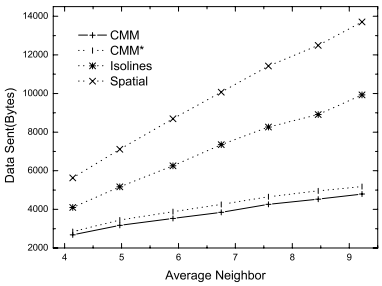


Fig. 8. Data sent at different densities

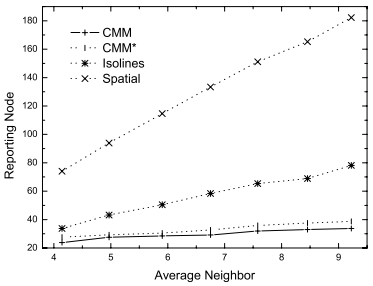


Fig. 9. Reporting nodes at different densities

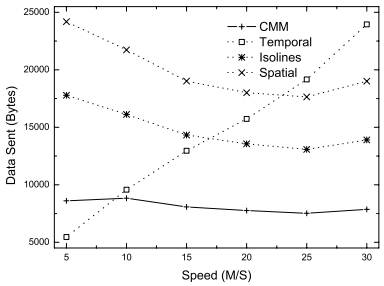


Fig. 10. Impact of contour changing speeds

make the CCM* and CCM algorithms out-perform the spatial and Isolines algorithms at large scales. Compared to the CCM* algorithm, the CCM algorithm gets an additional 8.5% average reduction on the total amount of report data.

### C. The Impact of Contour Changing Speeds

For simplicity, we only compare the CCM algorithm with other algorithms in this case. We simulated a vertical front which enters the network and moves from left to right continuously. The node sampling period is 2 seconds and each node continuously samples 6 times from the initial time. We evaluate the relation between the front moving speed and the total data sent.

Figure 10 shows the total amount of data sent by different algorithms for different front speeds. As seen in figure 10, the temporal algorithm only works well when the speed is low because the low speed lets the moving front only cover a few nodes and change their value ranges. The spatial algorithm is not affected by the speed because the total number of contour nodes is speed independent. The sink is in the center of the field and when a reporting node is far away from the center, more data has to be relayed, as reporting messages take more hops to reach the sink. When speeds are low (5 for example), most reporting nodes are near the left border. This is the reason why more data is sent by the spatial algorithm at low speeds. The overall performance of the spatial algorithm is poor because all contour nodes have to send their IDs and readings. The Isolines and CCM algorithms share similar properties as the spatial algorithm. Higher speeds will not influence the CCM algorithm, and it performs significantly better than all other algorithms.

## VI. Conclusion and Future Work

This paper presents the CCM algorithm for contour mapping in WSNs. The algorithm only chooses a subset of contour nodes to report. By using the CN-array, the report message size is pretty small even though each message contains all neighbor information. All these factors make our contour mapping energy efficient. Evaluation results in section V show that we can use the CCM reporting data to generate precise contour maps, and the CCM algorithm performs better than others in both sparse and dense network. CCM has two problems: (1) Node failure will cause neighbors to update neighbor order information, which brings an additional overhead to the network. (2) If a reporting node and its neighbors are not in adjacent value ranges. The sink cannot interpolate good values to neighbors, which will decrease mapping precision. We are going to working on such questions in the future.

## Acknowledgment

This work is supported by the US National Science Foundation under grant number IIS-0534429. Mike Worboys' work is also supported by the US National Science Foundation under grant numbers IIS-0429644 and DGE-0504494. He is also grateful for the support of the Ordnance Survey of Great Britain.

## References

[1] X. Meng, L. Li, T. Nandagopal, and S. Lu, "Event contour:an efficient and robust mechanism for tasks in sensor networks," *Technical Report, UCLA*, 2004.

[2] V. Shnayder, M. Hempstead, B. rong Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications." in *SenSys*, 2004, pp. 188–200.

[3] A. Silberstein, R. Braynard, and J. Yang, "Constraint chaining: on energy-efficient continuous monitoring in sensor networks." in *Proc. of the 2006 ACM SIGMOD Intl. Conf. on Management of Data*, 2006, pp. 157–168.

[4] I. Solis and K. Obraczka, "Efficient continuous mapping in sensor networks using isolines." in *Proc. of the 2005 MobiQuitous*, 2005, pp. 325–332.

[5] K. Chintalapudi and R. Govindan, "Localized edge detection in sensor fields." *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 273–291, 2003.

[6] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized fault-tolerant event boundary detection in sensor networks." in *INFOCOM*, 2005, pp. 902–913.

[7] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek, "Beyond average: Toward sophisticated sensing with queries." in *IPSN*, 2003, pp. 63–79.

[8] Y. Kotidis, "Snapshot queries: Towards data-centric sensor networks." in *Proc. of the 2005 Intl. Conf. on Data Engineering*, 2005, pp. 131–142.

[9] S. Pattem, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks." in *Proc. of the 2004 Intl. Conf. on Information Processing in Sensor Networks*, 2004, pp. 28–35.

[10] C. Zhong and M. Worboys, "Efficient contour mapping in sensor networks," *Technical Report 2007.5, http://www.spatial.maine.edu/ czhong/tr0701.pdf*, 2007.

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks." in *Proc. of the 2002 USENIX Symp. on Operating Sysems Design and Implementation*, 2002.

[12] P. Levis, N. Lee, M. Welsh, and D. E. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications." in *SenSys*, 2003, pp. 126–137.