

# The Design of a Spreadsheet Programming Interface for Sensor Networks

James Horey, Patrick Bridges, and Arthur Maccabe  
Department of Computer Science  
University of New Mexico  
Email: {jhorey, bridges, maccabe}@cs.unm.edu

Angela Mielke  
Los Alamos National Laboratories  
Email: amielke@lanl.gov

**Abstract**—Sensor networks are currently difficult to program and maintain. To simplify programming we present an interface for programming sensor networks based on the concept of spreadsheets. Our interface is explicitly designed to handle habitat monitoring and animal tracking scenarios, which we believe constitute a large portion of sensor network applications. Our approach is simple enough so that scientists without strong programming backgrounds can still program and maintain a network of sensors.

This Work-in-Progress explores the design of our interface and scenarios in which our design does well. This WIP also briefly explores some challenges we faced while designing our interface. We also discuss possible extensions to our design and applications that may benefit from those extensions.

## I. INTRODUCTION

Sensor network technology has the potential to greatly simplify data collection in many of the natural sciences. However, sensor networks largely remain difficult to program, especially for non-Computer Scientists. Languages such as NesC, although useful for research and low level programming, poses a large programming barrier. Current approaches to simplify sensor network programming take advantage of the idea that a sensor network can be treated as an online multi-dimensional database. This virtual database is then queried using multi-dimensional range queries [1]. Current implementations involve using a language similar to SQL to express such queries [2]. Although these SQL variants work well for networks with a small number of dimensions that exhibit regular behavior, we argue that such languages are unsuitable for more complex sensor networks. For this Work-in-Progress, we discuss the design of a spreadsheet programming interface for sensor networks. We show that by employing a spreadsheet programming model, we can alleviate much of the complexity of designing multi-dimensional range queries while remaining flexible and powerful.

## II. USAGE SCENARIOS

Our programming interface was explicitly designed to easily handle habitat monitoring and animal tracking scenarios. These two application types can encapsulate many different deployment scenarios while allowing flexibility in the details of the deployment. Since our interface is designed for non-Computer Scientists dealing with ecology oriented deployments, we felt that a spreadsheet interface was the most appropriate interface.

Habitat monitoring scenarios often involve stationary sensors placed in the environment that regularly take readings. Oftentimes, the environment is static enough so that the sensors and basestation can be placed near each other to ensure communication. Ecology-based examples of such deployments include Great Duck Island [3] and a more recent JPL microclimate study at the Sevilleta National Wildlife Refuge in New Mexico. <sup>1</sup>.

<sup>1</sup><http://sensorwebs.jpl.nasa.gov/resources/sevilleta.shtml>

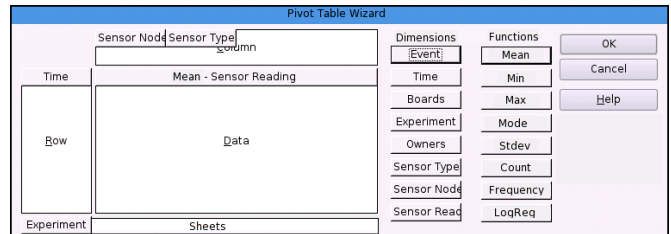


Fig. 1. The user is able to specify row, column, and sheet dimensions by dragging items from the Dimension column. He is also able to specify the data he is interested in viewing and what, if any, aggregation functions to perform on the data.

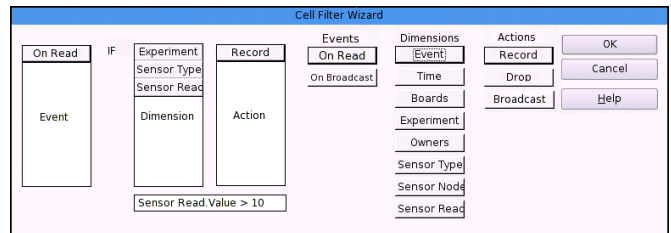


Fig. 2. The user is able to specify which dimensions to filter on and possible actions to take on a particular event. Here the user has instructed the mote to record the sensor reading if that value exceeds 10.

Animal tracking scenarios often involve several sensors attached to animals and one or more basestations. Stationary basestations are often placed near areas where the animals are known to visit. Unlike habitat monitoring, the animals may not visit the basestation at precise time intervals. As such, it is necessary for the basestation to pull data from the sensors whenever the sensors come within range. The sensors on the animals may collect data at well defined intervals and store the data for some period of time. The ZebraNet [4] deployment is the most popular example of this scenario.

## III. SPREADSHEET APPLICATION PROGRAMMING

In this section, we will briefly describe our interface and how a typical user may interact with this interface. Our interface exploits the natural division between the basestation functionality and mote functionality in habitat monitoring and animal tracking by logically differentiating basestation and mote programming. This simplifies programming while ensuring flexibility.

Programming the sensor network is an iterative process. Initially, the user is able to specify how often the data is to be collected for each sensor type. Afterwards, the user must specify which data he is interested in and how he wants to view this data on a series of two dimensional sheets. This is accomplished using a tool based

on the concept of Pivot Tables (Fig. 1). The user is also able to specify ranges on the various dimensions. By specifying the pivot table and dimension ranges, the user has effectively created a multi-dimensional range query. At this stage, the pivot table request and the sensor collection intervals are distributed onto the sensor network. Since these rules must reach every node, a simple flooding algorithm is used. Future implementations may explore alternative, more energy efficient algorithms.

Because pivot tables do not specify which particular dimension must be used in the data dimension, our interface allows the construction of complex, interesting queries. A user can specify that he is interested in collecting all the times when a particular event occurred from some group of sensors. Just as easily, he can ask the network to retrieve all the sensors that have a particular capability by specifying 'sensor ID' as the data dimension and specifying 'sensor types' along the sheet dimension. By organizing different subsets of the sensor network along the sheet dimension, the user can also specify different rules for different subsets of the network.

A mote rule (Fig. 2) consists of a simple if-statement that is triggered by some event and then performs some action. In order to evaluate whether an action should take place or not, the rule can use values from the event or from spreadsheet cells. If the cell is from another sheet, the values are transmitted from the appropriate motes to the currently selected subset. Currently the only available events are ON READ and ON BROADCAST, although we are exploring other events. ON READ events are triggered by sensor readings while ON BROADCAST events are triggered by specific radio readings. Available actions include DROP, RECORD, and BROADCAST. The RECORD action instructs to the mote to take the value specified in the filter and commit it to memory. The DROP action simply instructs the mote to disregard the data value. Finally, the BROADCAST action broadcasts a cell's contents over the radio. Using these rules, the user can implement simple data filter rules that conserve energy by dropping poor data values.

Once the user has specified all the rules for a particular subset of the sensor network, the user can then redistribute these rules onto the network. The user can also create new pivot tables to collect newly acquired data. In this manner, our interface encourages iterative programming of the sensor network.

#### IV. DEPLOYMENT

In collaboration with the Department of Biology at UNM, we expect to test our interface with other Computer Scientists and Biologists on an actual deployment scenario within the next several months. We will place Mica2Dots on several porcupines and regularly measure temperature and movement via accelerometer. We will have several stationary basestations consisting of Mica2 motes that collect the data whenever the porcupines are within range. These basestations will be placed in areas where the porcupines are known to visit. The stationary Mica2s will also measure various environmental conditions including temperature, humidity, air pressure, and light. The experiment will take place near the Rio Grande River in Albuquerque, New Mexico.

Since our deployment scenario involves elements of both animal tracking and habitat monitoring, we believe that it will offer a reasonable test of our interface. At the moment, the motes do not contain any explicit rules, and as such all sensor readings perform the default RECORD action. Once we have deployed the system and gathered some initial data, we will implement some basic rules. Since our system encourages iterative programming, we should be able to extend the initial programming with relative ease.

#### V. RELATED WORKS

Our work is most similar to SQL-based query systems, such as TinyDB [2]. However, in comparison with these systems, our interface provides a more intuitive interface while preserving flexibility. Our interface also allows the creation of a complex set of filtering rules that is difficult to create and manage using other systems.

#### VI. CURRENT STATUS

We have currently finished the initial design and begun the implementation of our interface. The primary spreadsheet interface will run on Linux based machines, although we hope to port the software to other platforms in the future. Our programming environment will target hardware running the TinyOS system. We are exploring different implementation strategies for pivot table requests including extending or modifying existing SQL-based query languages. For the future, we hope to integrate a form of distributed indexing to more efficiently route rules and queries to the appropriate motes [1]. We are also exploring different routing strategies to route the data from the motes to the basestation including directed diffusion [5].

Although we believe that our current design is sufficient for the applications mentioned, we would like to extend our initial design to accommodate more dynamic applications. Applications such as advanced fire monitoring and nuclear safeguarding may require complex, dynamic computation that is currently difficult to implement and reason about. We are exploring the possibility of adding new mote actions and events that may help implement these complex applications. This part of the work is still early and won't be included in the initial implementation.

#### VII. CONCLUSION

Constructing a flexible programming environment for non-programmers is difficult but necessary. Although current deployments often involve several Computer Scientists along with researchers from other fields, this will not always be the case. As sensor network technology becomes cheaper and more popular, the need for non-programmers to use these devices will increase. We believe that our spreadsheet interface significantly contributes to the goal of making these devices more accessible to the broader research community.

#### VIII. ACKNOWLEDGEMENTS

We would like to acknowledge Los Alamos National Laboratory for partially sponsoring our work under contract 0409J-123-04.

#### REFERENCES

- [1] X. Li, Y. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *Sensys*, 2003.
- [2] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The Design of an Acquisitional Query Processor for Sensor Networks," in *SIGMOD*, 2003, pp. 491–501.
- [3] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, GA, Sept. 2002.
- [4] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experience with ZebraNet," in *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, San Jose, CA, 2002.
- [5] C. Intanagonwivat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *Mobile Computing and Networking*, 2000, pp. 56–67.