A Wireless Body Area Sensor Network for Posture Detection

Conference Paper *in* Proceedings - International Symposium on Computers and Communications · January 2006

DOI: 10.1109/ISCC.2006.21 · Source: DBLP

A Wireless Body Area Sensor Network for Posture Detection

Elisabetta Farella, Augusto Pieracci, Luca Benini
DEIS - University of Bologna, Italy
{efarella, apieracci, lbenini}@deis.unibo.it

Andrea Acquaviva
ISTI - Urbino University, Italy
acquaviva@sti.uniurb.it

Abstract

Body Area Sensor Networks (BASN) are an emerging technology enabling the design of natural Human Computer Interfaces (HCI) in the context of Ambient Intelligence. This class of interactive applications poses new challenges on sensor network design that are hard to be faced using traditional solutions optimized for environmental monitoring-like applications. In this paper we present a novel solution for wireless and wearable posture recognition based on a custom-designed wireless body area sensor network, called WiMoCA. Nodes of the network, mounted on different parts of the human body, exploit tri-axial accelerometers to detect body postures. Afterwards we discuss results of interactive performance and power consumption optimizations required to match application constraints.

1. Introduction

Sensor networks have been extensively studied in the last decade in the context of environmental monitoring and target detection applications. Body area networks represent a recent evolution of this technology for the development of a new generation of human-computer interfaces (HCIs) to provide natural and context-aware access to personalized services. Sensor technology enables the development of small form-factor devices that can be mounted on wearable nodes communicating to each other with the purpose of motion tracking and gesture recognition.

Even if the idea of exploiting user postures to implement alternative input systems for cell-phones and personal digital assistants (PDA) has been explored in the last few years [6], many challenges are still to be faced. From an implementation viewpoint, recent work showed that wearability and low-power consumption requirements can be faced through an efficient design of sensor nodes [4, 7]. However, these requirements must be satisfied in conjunction with ever increasing application's demand for fast response time and flexibility. As a consequence, a complete implementation has not been presented up to now to the best of our knowledge.

In this paper we propose the first complete implementation of a distributed posture recognition application developed on top of a body area wireless sensor network called WiMoCA [4] where sensors are represented by tri-axial integrated MEMS accelerometers. This kind of accelerometers are effective for recognition purposes because of their small size and because they are immune from electromagnetic interferences and obstacles [2, 5, 1]. The system exploits accelerometers to detect postures and gestures of different parts of the human body. Compared to traditional sensor networks designed for environmental monitoring that are commonly designed to handle sporadic events, original design requirements are imposed to achieve an efficient posture recognition implementation both from the node software design and network organization.

From a network perspective, being the nodes mounted on the human body, they are all in the same coverage area and directly connected to a base station (installed as an extension board on a palmtop PC). The base station represents also a gateway that connects the network to a PC where the user application is located. On the other end, tight design constraints are imposed to match fast response time requirements of interactive applications while providing configurability and flexibility depending on the set of postures to be recognized, which in turns depends on the application context. In this work we describe a complete custom posture recognition solution that satisfies these trade-offs on top of WiMocA WBASN. Second, we describe the network architecture as well as node processing and communication software. We demonstrate its effectiveness through performance and power measurements on the application field.

Contribution of this paper is twofold. First, we describe a working implementation of the posture recognition system. Second, we present a custom software and network architecture for wireless body area sensor networks.

2 WBASN Architecture

WiMoCa [4] sensing node is designed to be wearable and low-power. It has a modular architecture to ease fast

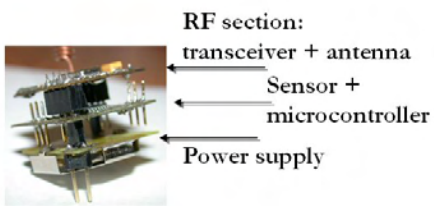

Figure 1. WiMoCa system structure

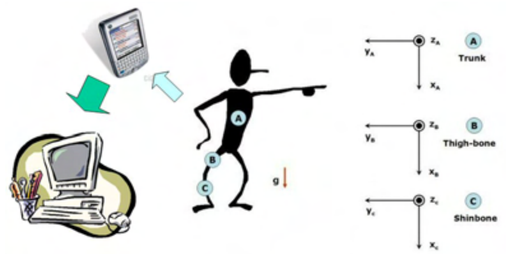

Figure 2. Setup on user of a three-node body network and general architecture

replacement and update of each component. It is composed by three sections (Figure 1) namely MCU/sensors, RF and power supply. The module sensibility obtained is 2mg and maximum throughput is 36Kbps. The core of the WiMoCa node is the low-cost, low-power ATmega8 8-bit microcontroller based on the AVR RISC architecture. The sensing is based on a MEMS tri-axial digital accelerometer by ST Microelectronics (LIS3L02DQ) which maximum power consumption is 5mW in active mode and 1 μW in sleep mode (at supply voltage of 3.3V). The RF section is based on the RFM TR1001 transceiver operating at 868 MHz. Among other nodes, the gateway is a special one which mounts two layers dedicated to communication interface. Similarly to other nodes, it has a RF section for exchanging data with the body network, but it also interfaces to the external world, e.g. a PC or workstation, through an interchangeable link such as RS232, Bluetooth or Ethernet.

The WBASN is organized with a star topology and it can be easily worn. WiMoCA nodes are end-points of the star and they are responsible for sensing and acquiring data from the environment before sending them to the gateway node (Figure 2). The gateway performs bridging functions and has not sensing device. It is used to form the backbone of the system by connecting the WBASN to a host machine or to connect an external network. Coordination among transmissions of various nodes to the gateway is performed at MAC-level.

3 Posture Recognition Application

In this section an accelerometer-based posture monitoring application is described as an example of the posture detection capability. The user is equipped with 3 sensing modules placed along the body (Figure 2), precisely on the trunk (node A), on the thigh-bone (node B) and the shinbone (node C). If we refer to the plane defined by the direction of gravity and the ideal line between the shoulders, the projection onto this plane of the axes relative to each sensor module is shown in the rightmost part of Figure 2. The application implemented aims at detecting user posture among seven different possibilities (sitting, standing and lying in four different manners). A certain sequence can indicate that the user fell down, thus an alarm can be generated to contact assistence.

Each of the three modules monitors the inclination of a certain part of the body, acquiring acceleration samples along three axis. Acceleration values are averaged first, then module tilt with respect to the gravity is computed and encoded. According to the schedule imposed by the contention-free MAC protocol implemented on each node (see Section 4), data are sent to the gateway. Thus, each module inclination collected at gateway side frame by frame can be combined to interpret body posture. This can be applied in a similar way to other body parts (arms, head, feet, etc.). Finally, the gateway communicates the result of detection to a Java application running on the host machine, where a Graphical User Interface (GUI) is updated to display current user posture.

3.1 End-nodes firmware

The firmware running on each end-node perform the following tasks: (i) it acquires sensing data and computes the average acceleration value for each axis within a time window; (ii) it combines the accelerations along the three axes to identify a known configuration; (iii) it implements the communication protocol with gateway-node to send node tilt in that frame.

Since accelerometers are sensitive to gravity acceleration and speed variations due to object movements, in steady-state conditions the gravity components along the three axis of the sensor can be used to determine the orientation of the sensor. In order to minimize false positives in the recognition accelerations are observed and averaged within an 8-samples time window.

We are interested in detecting three conditions: directed as, orthogonal to or opposed to the gravitational acceleration. Thus three subranges are identified. Combining the discretized value for each axis, the set of three acceleration is associated to a specific 4-bit configuration identifying the tilt of the module (see Table 1). The number of possible tilts

X	Y	Z	Module Tilt (4 bit)
p	o	o	0000
o	o	g	0001
o	p	o	0010
o	g	o	0100
o	o	p	1000
g	o	o	1010
none	none	none	0101

Table 1. End-node possible inclination: p = parallel but opposed to $\vec{g}$, g = directed as $\vec{g}$, o = orthogonal to $\vec{g}$

recognized can be varied according to application need, introducing more than 3 subranges.

At the end of the tilt identification process, the module tilt code is sent to the gateway according to the schedule imposed by the MAC protocol described in Section 4. Data are organized as shown in Figure 3.a, where we refer to the packet sent as *body messages*, reflecting their placement. Each unit organizes payload between two marking stamps indicating the beginning (start bit) and end (stop bit) of one acquisition and processing step. Between them four bits are dedicated to the node identifier (Node Id) and other four bits contain the module tilt code. Note that the network can be implemented with up to 14 nodes, enough for typical WBASN applications. A parity bit is also present. The data packet time (T_{dp}) is computed as shown in Figure, where T_{XON}, T_{XOFF} and T_{bit} are quantified in Section 5.

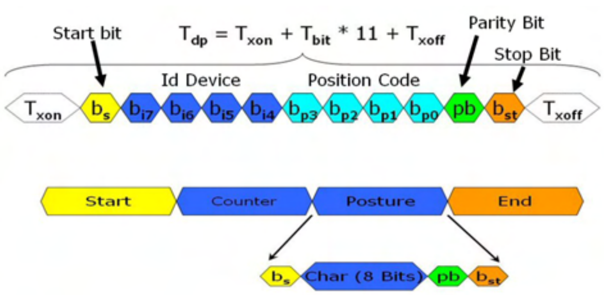

Figure 3. Body message (up); Backbone messages (down)

3.2 Firmware at gateway side

The firmware on the gateway microcontroller implements a set of task:(i) it implements the communication protocol with end-nodes at one side and communication with the general purpose system at the other; (ii) it acquires and accumulates tilt data from each end-module for a given frame; (iii) it combines all module tilts to identify a known body configuration; (iv) it sends to the application the identified position; (v) when necessary, it handles set-up of nodes and network (e.g. Message Table transmission,

Trunk	Thigh-bone	Shinbone	User posture (ASCII))
0000	0000	0000	1 (Standing)
0000	0001	0000	2 (Sitting)
0000	0001	0001	3 (Sitting with leg up)
1000	1000	1000	4 (Prone)
0001	0001	0001	5 (Supine)
0010	0010	0010	6 (Laying on right side)
0100	0100	0100	7 (Laying on left side)

Table 2. Posture Table in case of Figure 2

association of each node with a body part).

According to the Message Table resident identically on end-nodes and gateway, the latter waits for reception of tilt data from each end-node. On the gateway a table, referred here as Body Location Table (BLT), containing correspondence among node ID and position on the body is stored. A default BLT is initialized each time the gateway is reset, nevertheless the table can be programmed by the application running on the general purpose system at any time. The gateway can recognize a control command for reprogramming the BLT coming from the application, followed by the new data to fill the table. Supported by the BLT, data coming from different nodes are combined to determine user posture among a set of possible postures contained in another table, the Posture Table (Table 2).

After encoding, the user posture is sent through the RS232 or Bluetooth interface to the Java application. Data flowing from gateway to the processing unit where the application resides are called *Backbone messages* (Figure 3).

3.3 Java Application

The Java application accomplishes a set of simple tasks: (i) continuous data acquisition from the serial or Bluetooth port exploiting the Java COMM API; (ii) user posture code extraction from data stream; (iii) real-time display on the GUI of the image corresponding to actual user posture; (iv) initialization control commands, set-up of all Tables (BLT, Message Table, etc.) resident on the network. The communication from application to gateway is handled at start-up, when the application send a control character to the gateway to start the initialization of the WSN. Subsequently, the application sends the Message Table, if different from a default one stored in the gateway. At this time, but also later as consequence of user explicit intervention, the application can send the BLT. The application listens to the COM port (Bluetooth or RS232) for acquiring data coming from the gateway. The extraction of the encoded posture is committed to a specific thread that first identifies the start byte of the packet sent by the gateway, then the time-frame through the counter byte and finally the posture code. The code extracted is communicated to the GUI thread, which updates the display. The GUI offers to the user the chance to enter a new BLT through specific text fields and send buttons. This can be useful if a certain module is placed in a different location w.r.t. the default one at run-time.

4. WBASN Communication Layer

In this section main features of the communication layer are presented, focusing on the MAC-layer and synchronization aspects. Detailed timing and power characterization is given in Section 5.

4.1 MAC Layer

WiMoCA software implements a collision free MAC protocol, inspired by real-time MAC presented in [3]. Compared to collision avoidance protocols developed for sensor networks such as TMAC [9], collision free protocols are suitable to handle real-time traffic, since they avoid the overhead imposed by collision detection packets by allocating a time frame to each message [8]. In fact, the typical usage of the body area sensor network imposes a periodic sampling of acceleration values from a subset of nodes.

The CFRT (Collision Free Real-Time) protocol basically divides time into frames in which only one node is allowed to transmit. The scheduling order is derived by a Message Table stored in each node and identical for all the nodes, so that each of them knows when it has the right to transmit. The table contains an entry for each node allowed to transmit or receive in a frame. Fields in the entry specify source, destination nodes, message length and message period. Compared with [3], where the table is built by applying earliest deadline first scheduling protocol (EDF) to a queue of waiting messages, here the table is application dependent. The scheduling order is programmed depending on the job that must be performed by the body network.

For this reason, a cross-layer approach has been followed where the application is responsible to perform message scheduling. The Message Table at each node can be dynamically re-programmed via wireless channel at the beginning of each frame. This solution allows applications to program the Message Table depending on the sensing pattern, which in turn depends on the movement or gesture to be recognized. In general, the table keeps constant during an acquisition session, limiting time/energy spent for re-programming. During table-updating phase, a time synchronization message is also sent by the gateway to each node, since all the nodes must be synchronized to share the bandwidth. The synchronization message is sent by the gateway after a predefined time interval, as explained in the next section. In general, in each round of the table, called *table period*, a time slot (*management slot*) is allocated for synchronization and table updates. When it is not used for synchronization or table-updates, transceivers of all the nodes are switched off for its entire duration. The timeline of MAC protocol operations is shown in Figure 4. In the Section 5 we show the overhead for table update.

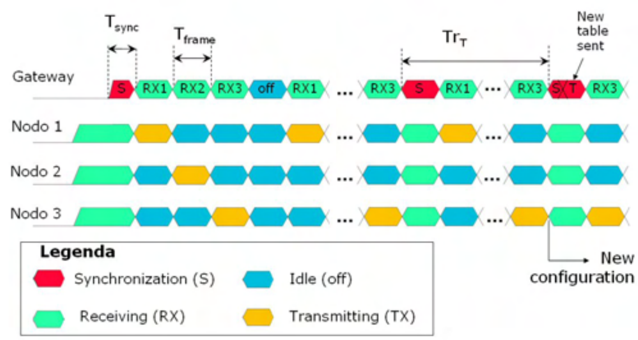

Figure 4. Timeline of the MAC protocol

When nodes are powered-up an initialization phase takes place. Here the microcontroller of each node programs a default table which contains an entry for each node in the network with message source corresponding to the gateway node. In this way each node can receive the start message. Nodes involved in the acquisition session receive the table containing the message order list through a *table-update* packet. The format of the table-update packet is shown in Figure 5. Other nodes are programmed to periodically listen to the channel at the end of each table period, so that they are ready to be involved in a new acquisition session later on. For the rest of the time their transceiver is shut-off to save power.

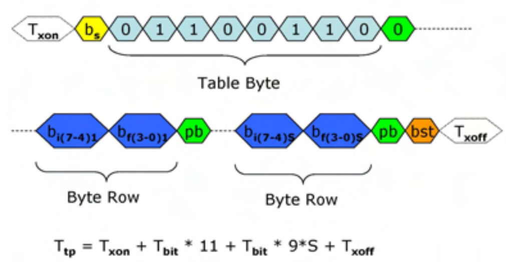

Figure 5. Format of the table-update packet

4.2. Synchronization Phase

The time synchronization is implemented with a global and unique timescale throughout the network, where every node maintains a clock that is synchronized with respect to a reference node. We elected the gateway node as the reference node. Every S frames a *Synchronization Phase* takes place (Figure 4). During this phase, nodes listen for the gateway that sends a broadcast synchronization packet that is used to synchronize the internal timer.

The number of frames between two synchronization phases is determined by the synchronization error (Figure 6). To compute it, we first consider that data must be sampled in the middle of T_{bit} that is the inverse of the transceiver bit rate. We focus our attention to a node that has to receive in a particular frame. Since receiving node is programmed to sample the incoming data in the middle of T_{bit}, the maximum synchronization delay between the beginning of the communication and the actual reading is $T_{bit}/2$. After this time, a reading error due to loss of synchronization occurs. To avoid synchronization errors, the internal timer

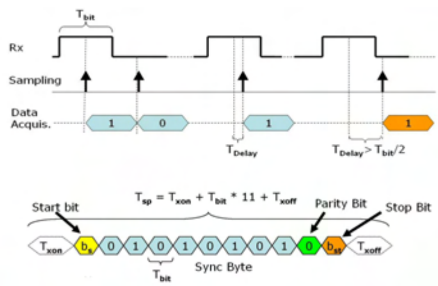

Figure 6. Synchronization Error (up); Format of the synchronization packet (down).

microcontrollers must be synchronized with each other before that the synchronization delay becomes larger than $T_{bit}/2$. In every node, the timescale is kept by an internal microcontroller counter. Counter is programmed to elapse each frame period and to generate an interrupt signal at the end time. The interrupt execution (accuracy) response is four clock cycles minimum, corresponding to 100nsec with 4MHz clock. We refer to this time as $DELTA$ from here on. In the worst case, a $DELTA$ is added every frame to the synchronization delay between the timescale of the gateway and other nodes. As a consequence, after S frames the total delay T_{delay} is $T_{delay} = DELTA \cdot S$. Hence, to avoid synchronization error, time condition is $T_{delay} \leq T_{bit}/2$ and $S \leq (T_{bit}/2)/DELTA$. In current implementation where $T\_bit = 31,25usec$, a suitable value for S is 156 frames. The format of the synchronization packet is shown in Figure 6. Depending on the bit rate of the transceiver the time required to send this packet varies from 1.19 to 0.384 msec. As shown in Figure, this time depends also on the time required to wake-up and shut-down the transceiver.

5. Experiments

5.1. Communication Layer Characterization

The communication architecture used in the W-BASN has been characterized. In the Table in Figure 7 we summarized some details concerning the communication protocol as well as timing and power characteristics. As shown in the table, we set a bitrate for the transceiver of 32Kbps. This choice has been made because the power efficiency of the transceiver is maximum for this rate. This is described in Figure 8. where we plotted the average power consumption per frame as a function of the bitrate of the transceiver. It can be noted that the average power decreases as a function of the bitrate, because the amount of time the transceiver is in transmitting state is lower. It can be also noted that power for transmitting a "1" bit is larger than the power spent for transmitting a "0" bit. This suggests the development of data encoding techniques aimed at reducing the number of ones to be transmitted. The design of such an encoding strategy is under study and will be object of future work.

Parameter	Expression	Our case
Bitrate (br)	---	32Kbps
Number of nodes	N (max = 14)	3
Number of rows in a table	S	3
Number of node positions	P (max = 16)	5 (4 + don't care)
Bits in a synchronization packet	----	11
Bits in a data packet	----	11
Bits in a table-update packet	11 + 9*S	38
Transceiver On to Off time	T_{XOFF}	20 usec (TR1001)
Transceiver Off to On time	T_{XON}	20 usec (TR1001)
Transmission power needed to send "1"	P_{TX1}	151,8 mW
Transmission power to send "0"	P_{TX0}	21,38 mW
Rx power	P_{RX}	23,70 mW
Power in OFF state	P_{OFF}	15,27 mW
Transceiver ON to OFF Power = Transceiver OFF to ON Power	$P_{XON}=P_{XOFF}=P_{RX}$	23,70 mW
Latency	T_L	1,536 msec
Battery Energy	E_B	100 mAh

Figure 7. Communication layer characteristics.

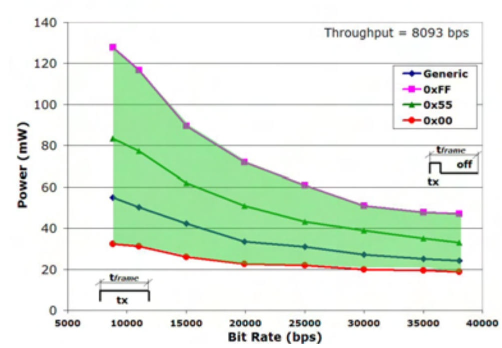

Figure 8. Average power consumption per frame as a function of bit rate

Data latency from acquisition to reception at the gateway is shown in Table in Figure 7. This value depends on the number of nodes in the network. This values increases as the number of nodes in the network because of the increasing of the table period.

We evaluated both table-update and synchronization packet overhead (see Table 9). Overheads are expressed in terms of average power spent for both transmission (at the gateway side) and reception (at the end-node side) of a synchronization packet and a table-update packet. It can be noted that for each packet we have an additional energy overhead due to OFF-ON and ON-OFF transitions of the transceiver. However, as shown in Table in Figure 7 this is negligible because the transition time is on the order of microseconds. The format of the complete packet has been shown in the lower part of Figure 3.

	Phase	Our case
power	TX syncronization packet (gateway)	$P_{TX\text{-}SP} = 74{,}65$mW
	RX syncronization packet (node)	$P_{RX\text{-}SP} = P_{RX\text{-}DP} = 23{,}70$mW
	RX table-update packet (node)	$P_{RX} = 23{,}7$ mW
	TX table-update packet (node)	$P_{TX\text{-}TP\,(M=11)} = 147$ mW $P_{TX\text{-}TP\,(M=0)} = 21{,}42$ mW
	Power overhead of synch packet	0,8%
time	Synchronization packet	0,384msec (br=32000)
	Table-update packet	1,23msec (br=32000)

Figure 9. Power and time overhead as a function of the packet type.

In Table in Figure 9 it is also shown the power overhead in terms of percentage of the total power due to the synchronization packets. As discussed in Section 4, power overhead for synchronization is accounted only each 156 frames because transceivers are otherwise shut-off during management slots.

5.2. Posture Detection Performance and Power Consumption

To assess power consumption of the node while running the posture recognition algorithm we performed additional characterization tests. Total power consumption at maximum posture sampling is 46 mW in the worst case (transmission of all ones) and 16,85 mW in the best case (all zeros). With 3 nodes we can achieve a maximum sample rate of 651 position per second (latency = 1,536msec). In practical cases, the maximum frequency of human movement is 30Hz, so that 60 positions per second (latency = 16,7msec) is a sufficient rate for detecting postures without loosing information. Battery lifetime depends on the frequency of samples. We note that the strong dependency of the transmission power on the type of bit affects also battery lifetime in a considerable way at full node activity. In addition, it must be noted that battery lifetime decreases much lower (from 19h to 7h) as the position sample rate increases (from 60 pos/sec to 651 pos/sec). This is because of the higher power efficiency of the transceiver at higher bitrate.

6. Conclusion

In this paper we presented the design and implementation of WiMoCA, a wireless sensor node based on tri-axial integrated accelerometers, aimed at detecting human gesture and postures to implement a human computer interface system. We also introduce the concept of wireless body area sensor network and we present its implementation exploiting WiMoCA nodes. The sensor network enables the implementation of a distributed recognition system to detect combined body postures and movements. We described all the software layers needed to support the acquisition, to implement the coordination and synchronization among nodes, to transfer gesture data up to the user application.

References

[1] E. Bachmann, X. Yun, D. McKinney, R. McGhee, and M. Zyda. Design and implementation of marg sensors for 3-dof orientation measurement of rigid bodies. In *'September*, pages 14–19, 2003.

[2] R. Barbieri, E. Farella, A. Acquaviva, L. Benini, and B. Riccó. A low-power motion capture system with integrated accelerometer. *January*, pages 418– 423, 2004.

[3] M. Caccamo, L. Y. Zhang, L. Sha, and G. C. Buttazzo. An implicit prioritized access protocol for wireless sensor networks. In *IEEE Real-Time Systems Symposium*, pages 39–48, December 2002.

[4] E. Farella, A. Pieracci, D. Brunelli, A. Acquaviva, L. Benini, and B. Riccó. Design and implementation of wimoca node for a body area wireless sensor network. pages 342–347, August 2005.

[5] T. Harada, H. Uchino, T. Mori, and T. Sato. Portable orientation estimation device based on accelerometers, magnetometers and gyroscope sensors for sensor network. *2003*, pages 191– 196, July 2003.

[6] R. Headon and G. Coulouris. Supporting gestural input for users on the move. In *Proceedings Eurowearable*, pages 107–112, September 2003.

[7] E. Jovanov, A. Milenkovic, C. Otto, and P. C. de Groen. A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation.

[8] J. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou. Real-time communication and coordination in embedded sensor networks. In *Proceedings of the IEEE*, volume 91, pages 1002–1022, July 2003.

[9] T. van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171–180. ACM Press, 2003.