Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2009, Article ID 474903, 13 pages
doi:10.1155/2009/474903

Research Article

Reliable Event Detectors for Constrained Resources Wireless Sensor Node Hardware

Marco Antonio López Trinidad[1] and Maurizio Valle[1,2]

[1] *MUlti-SEnsorS Laboratory, Biophysics and Electronics Engineering Department, University of Genoa, Via A'll Opera Pia 11a, 16145 Genova, Italy*
[2] *Microelectronics Group, Biophysics and Electronics Engineering Department, University of Genoa, Via A'll Opera Pia 11a, 16145 Genova, Italy*

Correspondence should be addressed to Marco Antonio López Trinidad, malopez@essex.ac.uk

Received 8 April 2009; Revised 8 July 2009; Accepted 24 August 2009

Recommended by Thomas Kaiser

A novel event detector algorithm, which points out in-door acoustic human activities, for constrained wireless sensor node hardware is proposed in the present paper. In our approach, event detections are computed from the signal energy statistics *change rate* at two instants separated by an $(L-1)$ samples interval. The experimentation is run in two phases: (i) the detector *characterisation* and *tuning* seek detector configurations that enable event detections from three acoustic human activities: *closing a door, dropping a plastic bottle,* and *clapping;* (ii) event detector *validation tests* measure the reliability to signal events from general acoustic activities, *people talking* particularly. The test results, which included emulated node hardware, actual sensor node, and a one-hop WSN, demonstrate the detector implementations signaled successfully events. And for the WSN, we found that event detections decay in a nonlinear fashion as the distance d, between the acoustic signal source and the sensor, is increased.

1. Introduction

A large and important number of Wireless Sensor Network (WSN) applications are event-driven. For instance, applications are found in the monitoring of dangerous environments, detection and classification of individuals or objects, location of static or tracking mobile targets, and structural health monitoring systems [1–3]. Unlike data collection applications, in event-driven applications the sensor nodes or *motes* rarely transmit data and their batteries energy is mainly consumed by intensive signal processing computations. Rather, under the event occurrences the motes transmit event packets upstream to the network data receptacles or sinks. At this point, the application can perform high-level inferences such as to compute the location, speed motion, and orientation of mobile or static targets. Eventually, the application can also make decisions and via the motes switch on/off alarms or actuators.

Due to that WSN event-driven applications work on the base of packet transmissions, the event detector and network failures, false alarm signals, and data packet lost severally deteriorate the sensors network life-time. In fact, for a sensor mote data packet transmissions are energetically the most expensive operations compared against any of the mote microcontroller operation states: CPU computing or energy saving [4]. Therefore, event detectors and event-based routing protocols are crucial WSNs applications services that must reliably and timely *signal* with the lowest possible number of false alarms and *transport* and *deliver* event packets with the highest achievable percentages [5, 6].

In this work a novel events detector algorithm, which points out the detection of acoustic indoor human activities, for constrained wireless sensor hardware is presented. The proposed algorithm computes events on the base of the signal energy statistics *change rate* at two instants separated by $(L-1)$ samples instead the use of a threshold.

Commonly, the event detector *characterisation* and parameters *tuning* processes are performed off-line in a general purpose computer system, to set a proper signal energy threshold value which is employed to compute event

occurrences [1]. Rather, that threshold search is a very complex and time consuming task that hides problems related to the strong constraints that the current wireless sensor hardware features such as the CPU computation speed, memory use availability, sampling rate limits, none hardware floating point support, and energy consumption concerns. Therefore, to show the differences that exist between a general computer system and constrained wireless sensor hardware domains, the experiments are run for two event detector implementations, an Octave program executed in a general purpose computer system by a Matlab kind mathematical environment tool and a TinyOS hardware-oriented that is run on a wireless sensor hardware emulator and an actual sensor mote.

A two-stage experimentation campaign is developed. (i) A *characterisation* and *tuning* processes are run to locate the event detector parameters configurations that enable the event detector signals event occurrences from a set of three acoustic human activities signals (*SS*1), *closing a door*, *dropping a plastic bottle,* and *clapping*. Due to the lack of user friendly interface facilities an actual wireless sensor mote features to debug mote programs, the characterisation, and tuning process results are reported for the Octave program and a TinyOS mote emulated, expecting that the mote emulation behaves closer the actual wireless sensor hardware. (ii) A *validation tests* process shows the event detector performance to detect effectively event occurrences from signals that belong to the acoustic signal *people talking* (*SS*2). In this case, the performance results are presented for the two event detector implementations, Octave and TinyOS, where the event detector execution includes an emulated node and an actual Micaz sensor mote. Finally, the TinyOS event detector is integrated in a one-hop sensor network of Micaz motes, and the detector performance test results are presented as a function of the distance d that exists between the acoustic signal source *SS*2 and the wireless mote.

The structure of this paper is as follows. In Section 2, the *event* and *event detection* terms are defined, and then the event detection scientific background is reviewed. Event detection theoretical basis, main hypothesis, and practical considerations are presented in Section 3. The characterisation, tuning, and validation test descriptions and implementation details are commented in Section 4. The experimental results, characterisation, tuning, and validation test processes, are presented and discussed in Section 5. Finally, Section 6 summarises and concludes the present research work.

2. Background

Throughout the rest of this paper, the *event* and *event detection* terms are employed in a reiterated fashion. Therefore, we provide their definitions that even though they cannot be considered as formal, our definitions are based on specific and qualitative interpretations of the observed input signal energy behaviors. Then, in the next sections, previous works in energy estimation and event detection are introduced and discussed.

Table 1: Number representation and execution times reported for the FFT, Wavelets Daub-4, and Daub-8 implementations on the Mica2 and Tmote Sky motes.

Complex task	Execution time
64-point FFT (Mica2 integer values)	52 ms [11]
512-point FFT (Mica2 floating point values)	30 seconds [9]
512-point FFT (Tmote Sky integer values)	4 seconds [10]
Wavelet Daub-4 (Tmote Sky floating point)	9 seconds [10]
Wavelet Daub-8 (Tmote Sky floating point)	20 seconds [10]

2.1. Definitions.

Definition 2.1. An event is a significant sudden change in the sensor signal energy.

Definition 2.2. Event detection is the capability an electronic system or computing algorithm has to recognise or count events.

It is assumed in Definition 2.1 that over long-time periods, in average the sensor signal energy reading output values experiment minimum changes.

Definition 2.2 refers to the event detector implementation that is developed as a nondeterministic finite state automata (*NFA*) [7].

2.2. Estimators.

Event detection strongly relies on the sensor signal energy estimation that is a function of the performance and complexity of specific algorithm implementations. In particular, estimator accuracy demands several hardware platform requirements such as computing power, memory space usage, and battery energy expenditure [8].

Roughly, estimation algorithms can be classified into two big groups: correlation and average based. In this section, related works are introduced.

2.2.1. Correlation-Based Estimators. The Fast Fourier (FFT) and Wavelets transforms are widely studied algorithms that can be used as estimators. Bhatti et al. [9], Skordylis et al. [10], and Xu [11] developed FFT and Wavelets Daub-4 and Daub-8 implementations in two representative sensor motes, the Mica2 [12] and Tmote Sky [13], for data compression and signal analysis. In Table 1, the algorithms performances are shown in terms of execution times, data window lengths, and number representations.

It can be noted that the execution time is mainly function of two factors: the number of operations developed and the number representation, integer, or floating point values. In particular, for an n elements series the FFT algorithm computational complexity is roughly $O(n^2)$ and in the better of the cases can be optimised to $O(\log_2 n/n)$ operations. Clearly, both correlation-based algorithms cannot provide timely energy estimations required for instance by location or target tracking WSNs applications.

2.2.2. Averaged-Based Estimators. Statistics sliding window and historic- or cumulative-based algorithms are computationally *efficient* and *lightweight* estimators [1, 14–17]. They

can compute online average estimations from a number series with relatively small amount of computing resources. In a window-based approach, the total M elements series s_i average is approximated averaging subsequences $s_j \subset s_i$ of N elements, where N is defined as the window size with $N < M$. The best global average estimation requires large N values. Therefore, application requirements mostly lead the window size selection: the *Moving Average* is an example of such sort of window-based estimators. In a historic or cumulative algorithm, the average is estimated by a weighted sum between the current series value plus the accumulated or old average estimations. The instantaneous average estimation is controlled by the weights that are adjusted to approach at best the actual total average value. The *Exponential Weighted Moving Average* or *EWMA* is an example of such estimators and there exists a large number of variants [17].

2.3. Event Detection. Gu et al. [1] implemented a WSN application for detection and classification of persons, individuals carrying metals, and moving vehicles. Specifically, threshold-based acceleration and magnetic and acoustic event detectors were implemented. To compute online input signal mean m_s, signal energy e_s, energy mean m_e, energy variance var_e, and energy standard deviation std_e estimations, the detectors utilise the EWMA filter. More particularly, the acoustic event detector performs intensive point-to-point comparisons between the microphone output signal energy e_s and an adaptive threshold thr_e. The thr_e is computed as the sum of the energy mean m_e plus the energy standard deviation std_e. If within a detection time period T_D, e_s crosses several times thr_e, then an event detection is flagged. In principle, there exists a *stable* thr_e value that should stand upon e_s for long time periods of null acoustic activity. Meanwhile, e_s should cross the thr_e for very short time periods when acoustic events are on the course.

Liang and Wang (2005) [14] reported two WSN event detectors that do not use a threshold. The first detector implements two sliding windows that compute the mean signal energy value for two contiguous time periods $E_a = \sum_{m=0}^{M-1} |z_{n-m}|^2$ and $E_b = \sum_{m=1}^{M} |z_{n+m}|^2$, where $z_i = \{z_1, z_2, \ldots, z_m\}$ is the sensor output signal samples. An event is signaled if the energies ratio accomplishes the condition $m_n = E_a/E_b \neq 1$. Rather, the detector effectiveness fails when the energy signal presents fast changes; therefore a hybrid fuzzy logic event detector is developed. The hybrid detector is inputted with the crisp or raw signal energy values accumulated E_s in a fixed time period and the signal energy ratio m_n values, that the fuzzificator converts in the semantic rules: *mean*, *weak,* and *strong*. The inference engine applies *IF-THEN* rules to compute the consequent that can take the linguistic values: *very weak*, *weak*, *medium*, *strong,* and *very strong*. Finally, the defuzzifier computes the crisp outputs that are a fuzzy weighted mean of consequent. Particularly, an event is signaled when E_s and m_n produce a *very strong* fuzzy weighted consequent mean. It can be easily noted that this last detector performance boost can have a very elevated computing cost for the 8-bit wireless mote CPU.

To provide high Quality of Service (QoS) levels and low latency wireless link reconnections for stream sensitive applications such as Voice over IP (VoIP) or online video, Mhatre and Papagannaki (2006) [15] developed, for IEEE 802.11 wireless multichannel Access Points (APs) and mobile devices, three hand-off mechanisms that rely on continuous wireless radio Received Signal Strength Indication (RSSI) signals monitoring. In the first algorithm, a mobile client scans all its wireless channels to search the AP that exhibits the instantaneous strongest RSSI levels; once the AP channel is found, a transition request is triggered and both devices commit for the data stream route change. Instantaneous RSSI readings do not guarantee stable wireless links predictions. Therefore, the second algorithm predicts wireless links on the base of RSSI mean values. From a three-region wireless stream throughput versus RSSI mean values map, *bad*, *intermediate,* and *excellent*, it is found that high QoS levels require network throughputs inside the excellent region; therefore the algorithm searches to maintain the network throughput within that region. The future region is predicted on the base of the *change of rate* of RSSI mean trends separated by an $(L - 1)$ samples length. From the comparison between the RSSI mean trend and the corresponding throughput region the next region can be computed. Whether the trend presents a downward behavior, the devices commit to switch the data stream to the channel with the highest RSSI mean trend. The last algorithm fits a linear regression model from the RSSI readings for each AP and computes a prediction of the future RSSI value region.

3. Signal Energy, thr_e, and Event Detection Algorithm

In our work, the EWMA filter is employed to compute online microphone signal mean m_s, energy mean m_e, energy variance var_e, and energy threshold thr_e estimations in the same fashion as in [1]. However, our event detector implementation computes rate changes cr_{thr} on thr_e as in [15] instead of to perform continuous comparisons between e_s and thr_e. In the next section, signal energy e_s and thr_e computing theoretical basis are explained, and then the change of rate concept is introduced.

The statistical lightweight algorithm estimators can compute the average value of series with very few computational resources [16, 17]. In this section, the *Exponentially Weighted Moving Average* (EWMA) principles as energy mean and energy variance estimators [1] and then the event detection approach are presented.

3.1. The Exponentially Weighted Moving Average. The EWMA estimator is presented in (1):

$$m_s[k] = \alpha s[k] + (1 - \alpha) m_s[k - 1], \tag{1}$$

where $s[k]$ is the current sample of the input signal s, and $m_s[k]$ is the current s average estimation. The weight $0 < \alpha < 1$ controls m_s, where a small α value gives more importance to past m_s values and therefore more stable estimations are obtained. On the other hand, large α values produce m_s

estimations that follow the dynamics of the instantaneous input signal s values and hence the EWMA output is more reactive. The EWMA fast convergence can be achieved; if the signal mean value m_s is known in anticipation, then the EWMA initial condition $m_s[0]$ is set to m_s. Rather, if the m_s value is unknown, then a thumb rule is to set $m_s[0] = s[0]$ [1, 15, 16].

In Figure 1 top part, the black line draws 20 seconds of a 2 kHz 10-bit resolution people talking signal series s. In the signal s first section there is approximately an 8-second silence period, from 0 to 16000 samples. In the signal s second part there is the people talking period that lasts around 5 seconds, from 16000 to 25000 samples. In the signal s final section there is a silence period that lasts around 7 seconds, from 25000 to 39999 samples. Finally, the clear line shows the signal s EWMA m_s values. In this case $\alpha = 0.001$ is set to filter the s noise high frequencies with $m_s[0] = s[0]$ set as in [1].

The signal energy is computed by (2):

$$e_s[k] = |s[k] - m_s[k]|. \tag{2}$$

The energy average estimation is obtained from (3):

$$m_e[k] = \beta e_s[k] + (1 - \beta) m_e[k - 1]. \tag{3}$$

Then, the energy variance estimation is computed from (4):

$$\text{var}_e[k] = \gamma (e_s[k] - m_e[k])^2 + (1 - \gamma)\text{var}_e[k - 1]. \tag{4}$$

The energy standard deviation is obtained from (5):

$$\text{std}_e[k] = \sqrt{\text{var}_e[k]}. \tag{5}$$

Finally, an adaptive energy reference or threshold thr_e is computed from (6) [1]:

$$\text{thr}_e[k] = \text{std}_e[k] + m_e[k]. \tag{6}$$

In Figure 1 lower part, the people talking signal energy e_s, EWMA signal energy m_e, and energy reference thr_e are presented in black, green, and yellow, respectively. In this case, four region features can be observed: (1) the signal energy e_s, energy m_e, and thr_e estimations convergence occur inside the 0 to 6000 samples period; (2) the signal energy e_s, m_e, and thr_e estimations of the silence period are shown inside the 6000 to 16000 samples interval; (3) the 5-second period of the speech signal energy e_s, m_e, and thr_e estimations runs from 16000 to 25000 samples; (4) finally, the silence signal energy e_s, m_e, and thr_e estimations are plot for the 25000 to 39999 samples period. In this case, the weights $\beta = \gamma = 0.02$ used in (3) and (4) are chosen in a manner that m_e and var_e present large variations when the instantaneous e_s changes are significant and therefore the event detector algorithm [1] can observe them easily.

An important observation is that an implementation of the (1), (2), (3), (4), and (5) requires five double word variables. Meanwhile, an enough accurate FFT implementation requires at least one set of $N = 256$ input samples data of one word (16-bits), plus operations RAM memory.

3.2. Event Detection Computation. We experimentally have found that the EWMA coefficients tuning, to set a threshold thr_e that achieves the features described in [1], is a very complex and time consuming process. Therefore, in a similar manner as in [15], we avoid the threshold thr_e search and compute events on the threshold change rate values (7):

$$\text{cr}_{\text{thr}} = \frac{\text{thr}_e[k] - \text{thr}_e[k - L + 1]}{L}. \tag{7}$$

Equation (7) defines a straight line that lies on thr_e and has slope parameter $1/L$ with its initial and final points, respectively, located at $\text{thr}_e[k - L + 1]$ and $\text{thr}_e[k]$ separated by a distance of $(L - 1)$ samples.

In this manner under ideal conditions, an event is evaluated on the base of the following logic.

(1) It is expected that for small signal energy variations, the difference $(\text{thr}_e[k - L + 1] - \text{thr}_e[k])$ is zero.

(2) For large variations of the signal energy, the cr_{thr} value must be different of zero.

Similarly as in the threshold approach, an event is signaled whether inside a T_D detection period, the condition $\text{cr}_{\text{thr}} \neq 0$ is achieved at least once. Note that when $L \to 0$, the detector performs large number of cr_{thr} computations and behaves like the threshold based event detection algorithms. On the other hand, when $L \to \infty$, less frequent cr_{thr} computations are performed.

Unfortunately in real world environments, thr_e fluctuates in time and cr_{thr} swings around the zero value. Therefore, a tolerance interval $I_{\text{tol}} = [-\text{tol}, +\text{tol}]$ ought to be introduced on the cr_{thr} changes and the occurrence of an event is evaluated by (8):

$$\text{event } (T_D) = \begin{cases} 1, & \text{if } |\text{cr}_{\text{thr}}| > \text{tol}, \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

where $T_D = M \cdot L$ seconds and M is the times number at which the detector must compute the occurrence of events.

It can be observed, from (7) and (8), that the detector sensibility can be controlled by the tolerance I_{tol} and the cr_{thr} slope $1/L$ parameters.

(1) I_{tol} is the maximum swing limit allowed on the cr_{thr} change rate that is a function of the threshold thr_e variations. From here, low energy intensity events will produce slight cr_{thr} variations that can be more easily detected by narrow I_{tol} values. Rather, also some noise background energy values can produce large cr_{thr} oscillations that will cross very narrow I_{tol} values without difficulty and therefore some nuisances can be signaled.

(2) Within a T_D period, the maximum number of cr_{thr} computations is bounded by the L period. For short L values the detector must more frequently compute cr_{thr}. In this manner, events from fast signal energy variations can be more easily detected. On the other hand, the detector must perform less cr_{thr} computations for large L values. Rather, some fast signal energy events may not be detected.

Therefore, the I_{tol} and L selection is a trade-off based on the application requirements. Once provided event detection theoretical basis and main hypothesis, in the following section specific detector implementation details and experimental performance results are presented.

4. Event Detector Implementation, Characterisation, Tuning, and Validation Test Descriptions

Real world event detectors can signal *False Alarms* (*FA*s) or *True Alarms* (*TA*s) events [18, 19].

(i) *FA*s are unwanted nuisances produced by sudden noise background changes. On one hand, *FA*s represent expensive radio transmissions that reduce importantly the mote life-time. On the other hand, *FA*s introduce system state uncertainty to the WSNs application.

(ii) *TA*s are events generated, for instance, by environment intrusions or system failures that therefore must be signaled timely as much as possible.

For a WSN application, *TA*s and *FA*s are qualitatively indistinguishable. Therefore, a characterisation and tuning process is run to locate the detector parameters configurations that enable the event detector signals minimum number *FA*s and maximum *TA*s events.

In this section we present the following two phases experimental procedure.

(1) *Characterisation and tuning processes*. *FA* and *TA* events are plotted for several event detector parameters I_{tol} and L ranges. Particularly, a set *SS*1 of three acoustic human activities is considered. Then, from the individual parameter ranges that produce the minimum *FA* and maximum *TA* events number, common I_{tol} and L ranges are merged.

(2) *Validation tests process*. It is measured the event detector capability to signal events from an acoustic signal set (*SS*2) that belongs to the *people talking* activity, for diverse detector parameters pairs $\{I_{\text{tol}}, L\}$ selected from the common I_{tol} and L values.

Note that the acoustic signal signature recognition is out of the characterisation and the validation tests processes scope that just points out the detection of acoustic events.

Two event detector implementations are developed: an Octave (`OCT_IMP`) which is a Matlab style program and a TinyOS (`TOS_IMP`) [20] which the actual mote hardware executes. As the actual mote lacks of interface facilities to debug the sensor programs behaviors, the `TOS_IMP` characterisation and tuning experiment outcomes are drawn from AVRora [21], a motes hardware emulator. The validation tests are run for the `OCT_IMP` and `TOS_IMP` where the execution on an emulated and actual mote hardware is included.

4.1. Experimental Setup Elements. In this section, a description is given of the signals sets, *SS*1 and *SS*2, employed in the event detector characterisation, tuning process, and validation tests. Then, the event detector implementation details and the event detector execution environment differences are exposed.

4.1.1. The SS1 and SS2 Signal Set Features. The events are computed from actual acoustic sensor signal data: characterisation $SS1 = \{s_1 = \text{closing a door}, s_2 = \text{dropping a plasticbottle}, s_3 = \text{clapping}\}$; see Figures 2, 3, and 4 respectively. Test $SS2 = \{s = \text{people talking}\}$; see Figure 1. *SS*1 and *SS*2 are 20-second records of 2 kHz sample data rate acquired from the Micaz microphone, stored in the external FLASH memory and eventually downloaded in a PC. The data is recorded in a laboratory environment with the signal source located at 1 m from the mote, and the microphone gain is set to 0 units.

4.1.2. The Event Detector Implementation Description. Figure 5 shows the event detector as a 6-state *Nondeterministic Finite State Automata* (*NFA*) and the particular implementation details follow.

(1) *Start* state. The EWMA estimator coefficients (α, β, and γ), initial conditions, and event detector constants (T_D, L, I_{tol}, etc.) are initialized. In particular, Table 2 shows the EWMA and T_D values that are set as in [1].

(2) *Get next sample and process* state. The current k sensor signal sample s is read and then the m_s, e_s, m_e, std_e, var_e, and thr_e values are computed by (1), (2), (3), (4), (5) and (6). Meanwhile the L period has not elapsed, the next $k+1$ signal sample is acquired and processed.

(3) *Compute* cr_{thr} state. Once an L period has elapsed, the cr_{thr} is evaluated by (7). If $|\text{cr}_{\text{thr}}| > \text{tol}$ (8), then a cr_{thr} changes counter is incremented. Meanwhile the T_D period is not reached, the machine acquires and processes signal samples for another L period.

(4) *Compute event* state. After a $T_D = M \cdot L$ period has elapsed, if the cr_{thr} changes counter is nonzero, then an event flag is activated.

(5) *Signal event* state. Whether the event flag is active, the event detector signals the occurrence of an event. Particularly, in the `TOS_IMP` event detector implementation a data packet is issued.

(6) *Clear detector counters* state. The event detector cr_{thr} changes counter and other auxiliary counters are set to zero, and then a new T_D period is started.

4.1.3. `OCT_IMP` Execution Environment Description. The `OCT_IMP` event detector is executed in a Linux desktop PC. In general terms, the event detector reads and processes one after the other sensor samples (see Figure 5). This procedure is continuously repeated until the last sensor

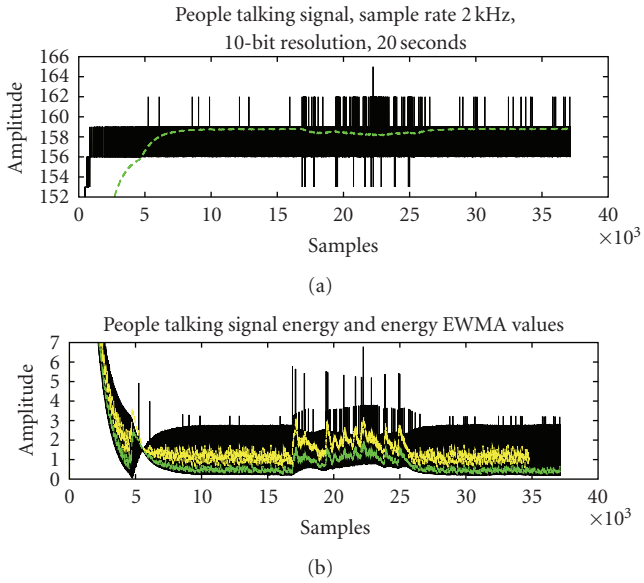

FIGURE 1: (a) In black color the people talking signal series s and (b) in clear color the signal EWMA m_s values for $\alpha = 0.001$.

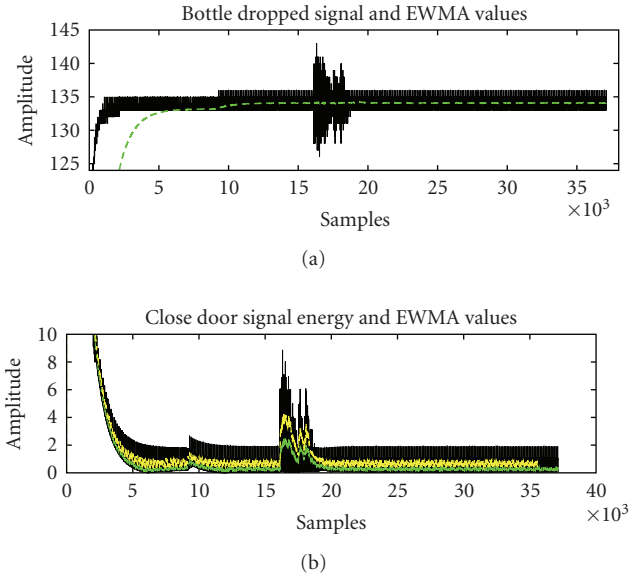

FIGURE 3: (a) In black the *bottle dropped* input signal series $s_2 \in SS1$ and in green the EWMA mean m_2 estimation. (b) In black the s_2 energy, in green the energy EWMA mean m_e, and in yellow the energy thr_e values.

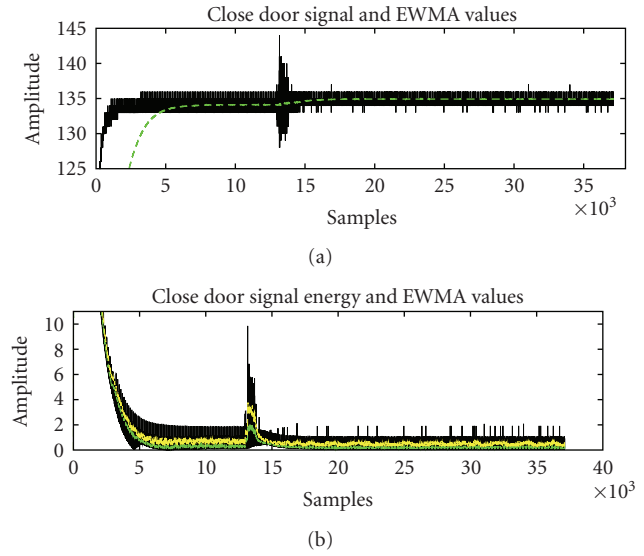

FIGURE 2: (a) In black the *close door* input signal series $s_1 \in SS1$ and in green the EWMA mean m_1 estimation. (b) In black the s_1 energy, in green the energy EWMA mean m_e, and in yellow the energy thr_e values.

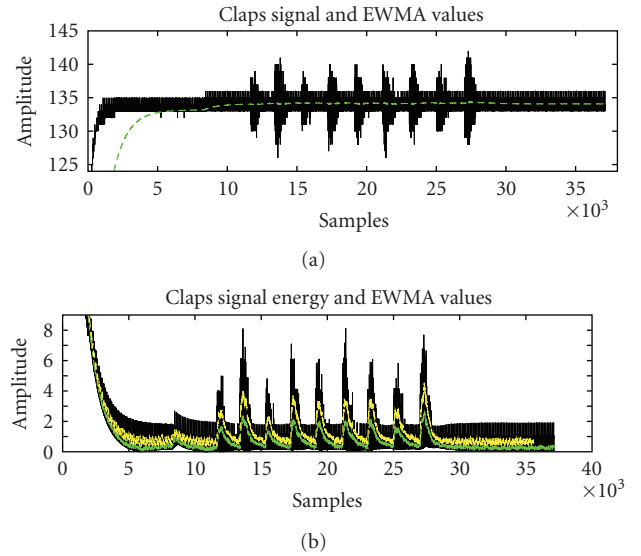

FIGURE 4: (a) In black the *claps* input signal series $s_3 \in SS1$ and in green the EWMA mean m_3 estimation. (b) In black the s_3 energy, in green the energy EWMA mean m_e, and in yellow the energy thr_e values.

data sample is achieved. In particular, the 2 kHz sensor samples are loaded in the PC RAM memory from where the event detector reads the data. In this manner, the algorithm operations are isolated from the acquisition data sample rate, interruption handle, and operations execution latencies issues, for instance. On the other hand, the event detector floating point operations are not limited by the data type precision representation featured by the execution environment and the programming language.

4.1.4. TOS_IMP Execution Environment Description. AVRora [21] is a WSNs simulator with accuracy at mote CPU bit and clock cycle level. In the TOS_IMP implementation, 1.3 kHz sample data rate acquisitions are assumed. To do this, AVRora reads from a text file the sensor data samples and the time between two consecutive sensor readings. On one hand, every sensor sample is sequentially exposed and hold on the corresponding virtual microcontroller ADC port. On the other hand, the time is translated into virtual

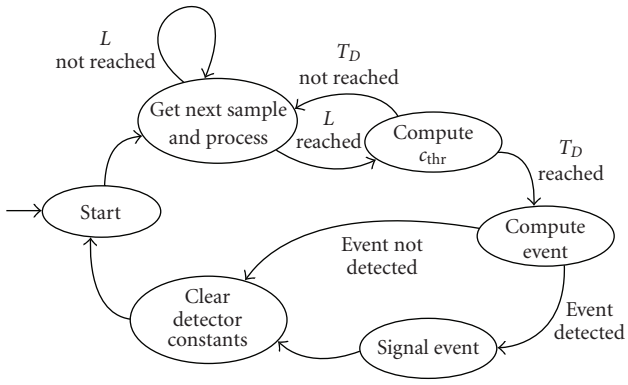

Figure 5: Six states acoustic event detector *Nondeterministic Finite State Automata* (NFA). *Start* state: the event detector constants are initialized. *Get next sample and process* state: the current k signal sample s is acquired and m_s, e_s, m_e, var_e, std_e, and thr_e are computed. *Compute* cr_{thr} state: the $|\text{cr}_{\text{thr}}| > \text{tol}$ condition is evaluated. *Compute event* state: the events occurrence is evaluated. *Signal event* state: event occurrences are signaled. *Clear detector counters*: detector counters are cleared.

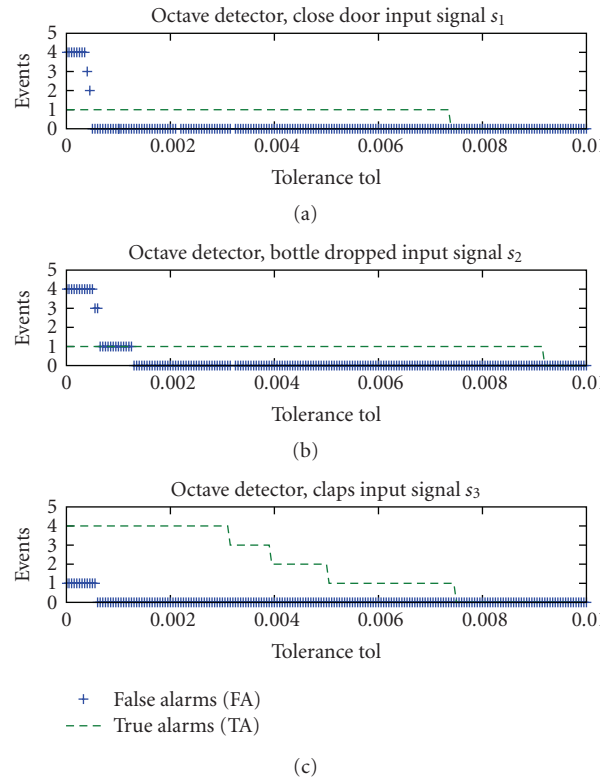

Figure 6: `OCT_IMP` event detector characterisation as function of tol and $L = 160$ milliseconds. (a) the events signaled for the signal $s_1 \in SS1$. (b) the events signaled for the signal $s_2 \in SS1$. (c) the events signaled for the signal $s_3 \in SS1$.

microcontroller CPU clock cycles that once elapsed then the next sensor data sample is introduced. In a similar fashion, the emulated microcontroller interruptions are driven in terms of the virtual microcontroller CPU cycles. In our case, AVRora reads, introduces, and holds the 2 kHz sensor samples every 3686 virtual microcontroller CPU cycles and the interruptions are generated every 5671 cycles. Even when the Micaz mote can sample data faster, the software floating point computations introduce large overheads that limit the data acquisition to a 1.3 kHz sampling rate; otherwise potential data race-condition conflicts can occur.

5. Experimental Results

As stated in Section 3, the event detector sensibility can be controlled by the two detector parameters tol and L. To reduce the event detector characterisation process complexity the event detections, *FA*s and *TA*s, are shown in a two-step procedure: (1) the tol parameter is varied maintaining L fixed, and (2) the tol parameter is fixed and the L parameter is varied.

In all the experiments, $T_D = 1.28$ seconds and in principle the events signaled maximum number is 15 for the 20 seconds signal records duration.

5.1. `OCT_IMP` Event Detector Behavior as Function of the tol Parameter. Figure 6 shows events, *FA*s and *TA*s, signaled by the `OCT_IMP` event detector implementation varying the tol parameter and $L = 160$ milliseconds for the characterisation signals set *SS*1. On the x-axis, the tol values range from 0.000001 to 0.014, in 0.00005 step unit increments. On the y-axis are drawn the total events, *FA*s and *TA*s, signaled within the 20-second signal records duration.

From the event detector execution results, three event regions can be observed: *Region I*, with both events, *FA*s and *TA*s, signaled. *Region II*, with only *TA*s events signaled.

Table 2: EWMA α, β, and γ coefficients and event detector detection period T_D constant values.

Constant name	Value
α	0.001
β	0.02
γ	0.02
T_D	1.28 seconds

Region III, neither *FA*s nor *TA*s events are signaled. Table 3 summarizes the event regions and their respective tol intervals for the signals set *SS*1.

5.1.1. `OCT_IMP` Tuning for the tol Value. From the data in Figure 6 and Table 3, it is clear that to signal effectively *TA* events from the signals sets *SS*1, an event detector implementation ought to work with tol values within the *Region II*. Therefore, a common I_{tol} interval can be merged for the signals set *SS*1 in the following manner:

$$\begin{aligned}\{\text{tol}\} &= (0.00045, 0.00735) \cap (0.00125, 0.00915) \\ &\quad \cap (0.00055, 0.00745) \\ &= (0.00125, 0.00735).\end{aligned} \tag{9}$$

More particularly, it can be noted that a tol $= \pm 0.003$ is one value that enables the event detector signals the

Table 3: OCT_IMP tol interval event regions for the acoustic test signals *SS*1.

Characterisation signal *SS*1	tol intervals		
	Region I	Region II	Region III
s_1	(0.000001, 0.00045)	(0.00045, 0.00735)	(0.00735, 0.01)
s_2	(0.000001, 0.00125)	(0.00125, 0.00915)	(0.00915, 0.01)
s_3	(0.000001, 0.00055)	(0.00055, 0.00745)	(0.00745, 0.01)

Table 4: OCT_IMP L interval event regions for the acoustic test signals SS1.

Characterisation signal *SS*1	L intervals in ms		
	Region I	Region II	Region III
s_1	(40, 53.3)	(53.3, 320.0)	(426.7, ∞)
s_2	(40, 53.3)	(53.3, 320.0)	(426.7, ∞)
s_3	(40, 106.7)	(106.7, 426.7)	(640, ∞)

maximum TA events number for the periodic acoustic signal $s_3 \in SS1$.

5.2. OCT_IMP Event Detector Behavior as Function of the L Parameter. In a similar manner as in the previous section, the OCT_IMP event detector behavior is presented as function of an L values range and tol $= \pm 0.003$. Figure 7 shows the events, FAs and TAs, signaled. On the x-axis, the L values run, from 15 to 700 milliseconds, in two L increments partitions. Inside the x-axis first partition, from 15 to 160 milliseconds, the L increments are steps of $T_D/8 + 2 \cdot n = 1.28/8 + 2 \cdot n$ seconds with $n = \{0, 2, \ldots, 18\}$. It gives cr_{thr} computations that vary from 44 to 8 times inside a T_D period. In the x-axis second partition, from 183 to 700 milliseconds, L increments are steps of $T_D/m = 1.28/m$ seconds with $m = \{3, 4, 5, 6, 7\}$. In this case, the cr_{thr} computations vary from 7 to 3 times inside a T_D period. On the y-axis, for every L value the events signaled total number is drawn for the 20 seconds signals set *SS*1 duration.

The three event regions and the respective L intervals are summarised in Table 4.

5.2.1. OCT_IMP Tuning for the L Value. From *Region II*, $L = [106.7, 320.0]$ milliseconds is the common interval computed in a similar fashion as in Section 5.1 for the common {tol} interval. It in principle means that any $106.7 \leq L \leq 320.0$ milliseconds value enables the event detector signals events from the acoustic signals set *SS*1.

5.3. TOS_IMP Event Detector Behavior as Function of the tol Parameter. Figure 8 shows the events, FAs and TAs, signaled by the TOS_IMP event detector as function of the tol value for the signals set *SS*1. On the x-axis, tol ranges from 0.0001 to 0.01285 in 0.00005 step units. On the y-axis, for every tol value the events' signaled total number is drawn for the 20 seconds signal records duration.

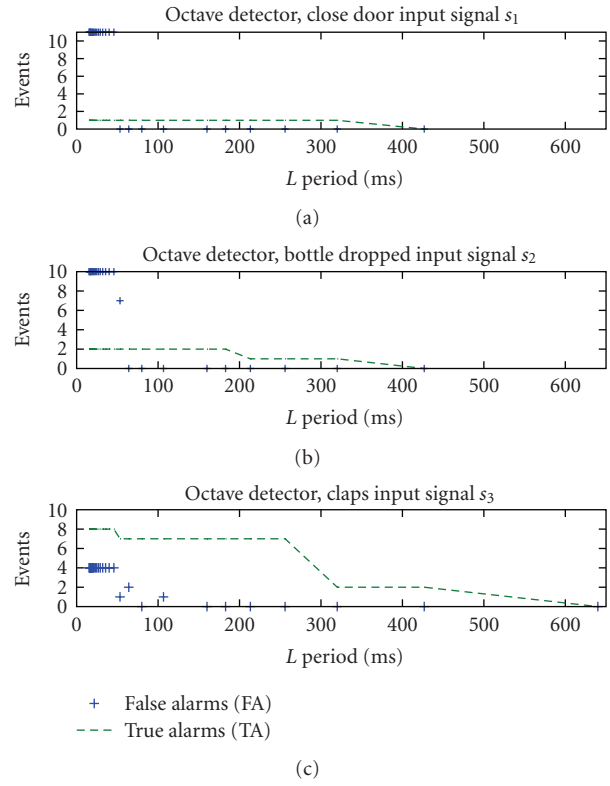

Figure 7: OCT_IMP event detector behavior as function of detector parameters tol $= \pm 0.003$ and L. (a) the events signaled for the signal $s_1 \in SS1$. (b) the events signaled for the signal $s_2 \in SS1$. (c) the events signaled for the signal $s_3 \in SS1$.

Table 5 shows the three event regions for the respective tol intervals of the characterisation signals set *SS*1.

5.3.1. TOS_IMP Tuning for the tol Value. From *Region II*, it is found that {tol} = (0.0047, 0.00885) is the common interval that allows the detector signals events for the signals set *SS*1. In particular, tol $= \pm 0.007$ is one value that allows to signal the most TA events for the signal $s_3 \in SS1$.

5.4. TOS_IMP Event Detector Behavior as Function of the L Parameter. In Figure 9, the TOS_IMP event detector behavior is presented as function of an L range values and tol $= \pm 0.007$. On the x-axis, L ranges from 15 to 300 milliseconds and the increments are distributed in the same fashion as in Section 5.2 is explained. On the y-axis, the events signaled

Table 5: TOS_IMP tol interval event regions for the acoustic test signals SS1.

Characterisation signal *SS*1	tol intervals		
	Region I	Region II	Region III
s_1	(0.0001, 0.0007)	(0.0007, 0.00885)	(0.00885, 0.014)
s_2	(0.0001, 0.00465)	(0.00465, 0.01115)	(0.01115, 0.014)
s_3	(0.0001, 0.0047)	(0.0047, 0.01275)	(0.01275, 0.014)

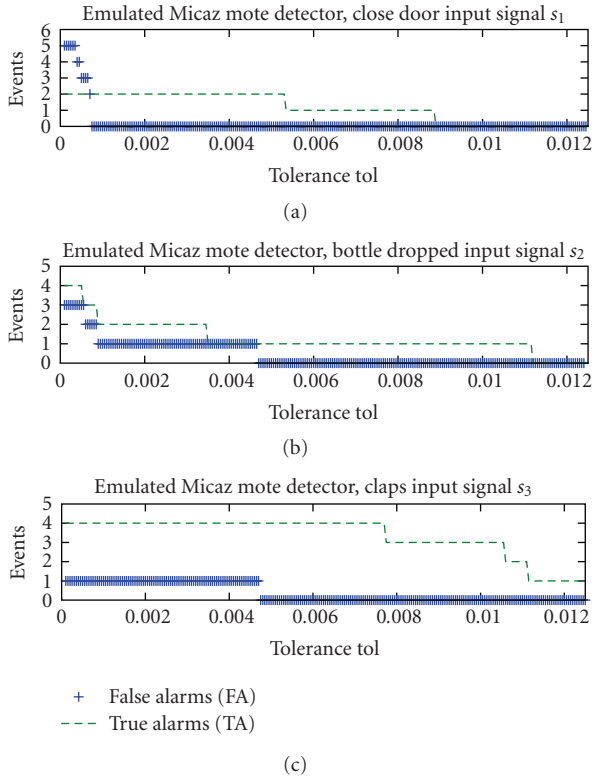

Figure 8: TOS_IMP emulated mote event detector behavior as function of tol and $L = 160$ millissecond. (a) the events signaled for the signal $s_1 \in SS1$. (b) the events signaled for the signal $s_2 \in SS1$. (c) the events signaled for the signal $s_3 \in SS1$.

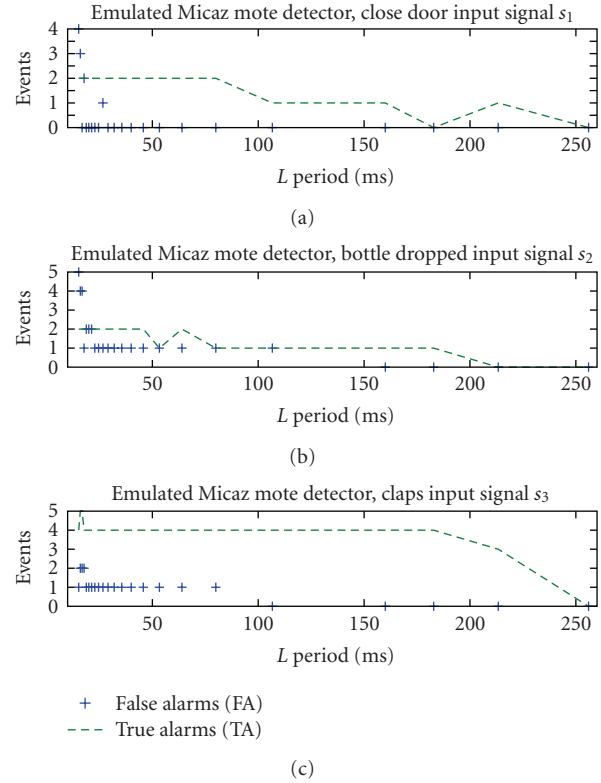

Figure 9: Emulated TOS_IMP event detector behavior as function of detector parameters tol $= \pm 0.007$ and L. (a) the events signaled for the signal $s_1 \in SS1$. (b) the events signaled for the signal $s_2 \in SS1$. (c) the events signaled for the signal $s_3 \in SS1$.

total number is plotted for the 20 seconds signal records duration. Table 6 summarises the respective three L event regions.

5.4.1. TOS_IMP Tuning for the L Value. From *Region II*, it is found that $L = (106.666, 160)$ milliseconds is the common interval that enables the detector signaling correctly the events from the signals set *SS*1.

As a comparison manner, Table 7 summarises the tol and L intervals obtained from the characterisation processes. Particularly, in the *Range size* column, a measurement of the interval lengths is presented, for the OCT_IMP and TOS_IMP emulated node event detector implementations, respectively.

It can be observed, on the one hand, that the OCT_IMP tol interval is 1.4 times wider than the TOS_IMP tol interval and is left shifted by 0.00345 units. On the other hand, the OCT_IMP L interval is 3.9 times wider and includes completely the TOS_IMP L values interval.

5.5. Validation Tests. In this section, the event detector capability is measured to signal events from acoustic signals other than the signals set *SS*1 for several event detector parameter pairs $pp_i = \{\{\text{tol}\}, L\}$. Particularly, the event detector, OCT_IMP and TOS_IMP, is exposed to the *people talking* signal *SS*2 plotted in Figure 1.

5.5.1. Evaluation Assumptions. In our sample tests, events are considered *TA*s if the event detector produces signals inside the time period when the acoustic signal of interest occurs. Otherwise, the events are considered *FA*s. More specifically,

Table 6: TOS_IMP L interval event regions for the acoustic test signals SS1.

Characterisation signal $SS1$	L intervals in ms		
	Region I	Region II	Region III
s_1	(15.238, 17.77)	(17.77, 160)	(160, ∞)
s_2	(15.238, 106.66)	(106.66, 182.857)	(182.857, ∞)
s_3	(15.238, 106.666)	(106.666, 213.333)	(213.333, ∞)

Table 7: Detector tol and L intervals for the OCT_IMP and TOS_IMP emulated mote event detector implementations.

Detector implementation	tol	Range size	L (ms)	Range size
OCT_IMP	(0.00125, 0.00735)	0.0061	[106.7, 320.0]	213.3 ms
TOS_IMP emulated hardware	(0.0047, 0.00885)	0.00415	(106.666, 160.0)	54.666 ms

we expect that the detector only signals TA events inside the *speech* signal period, from 16000 to 25000 samples, with a theoretical maximum of 5 TAs for a $T_D = 1.28$ seconds period. In this manner, the successful detection percentage is in every case computed with respect the maximum signals number which has assigned the 100%.

5.5.2. Evaluation Special Cases. For the TOS_IMP implementation case, the event detector is executed on an emulated sensor mote and on an actual Crossbow/Berkeley Micaz mote. Finally, the TOS_IMP event detector is integrated in a one-hop wireless sensor network of Micaz, and the events are presented as a function of the distance d that exists between the acoustic signal $SS2$ and the Micaz mote.

5.5.3. OCT_IMP Event Detector Validation Tests. The OCT_IMP event detector performances are shown in Table 8, for the signal $SS2$ and the detector parameters pairs: $pp_1 = \{\text{tol} = \pm 0.003, L = 160 \text{ milliseconds}\}$, $pp_2 = \{\text{tol} = \pm 0.007, L = 160 \text{ milliseconds}\}$, $pp_3 = \{\text{tol} = \pm 0.003, L = 320 \text{ milliseconds}\}$ and $pp_4 = \{\text{tol} = \pm 0.007, L = 320 \text{ milliseconds}\}$.

pp_1: the event detector computes one cr_{thr} change for three consecutive T_D periods inside the *speech* period of the signal set $SS2$. Therefore, the detector signals three events that are counted as TAs giving a 60% successful detection.

pp_2: the event detector computes $\text{cr}_{\text{thr}} = 0$ changes for the whole signal $SS2$ period duration. Therefore, neither FA nor TA events are signaled and a 0% successful detection is obtained. This behavior is expected, becouse tol $= \pm 0.007$ is one among the more restrictive values inside the merged tol set. That is, the tolerance interval is pretty ample that cr_{thr} can not cross it, even though the signal energy presents periodic fluctuations as in the case of the signal $s_3 \in SS1$; see Figure 6.

pp_3: the detector computes one cr_{thr} change inside the *speech* period of the signal set $SS2$. Therefore, only one event is signaled and counted as TA giving a 20% successful detection.

pp_4: the detector computes $\text{cr}_{\text{thr}} = 0$ changes within all the signal $SS2$ period duration; therefore neither FA nor TA events are signaled, giving a 0% successful detection. This result is expected, as the detector parameter values tol and L are the most restrictive for any signal from the characterisation set $SS1$; see Figures 6 and 7.

Table 8: OCT_IMP event detector performance for the signal set SS2.

Detector parameters values	Detector event signals		
	FA	TA	Successful detection %
$\{\text{tol} = \pm 0.003, L = 160 \text{ ms}\}$	0	3	60
$\{\text{tol} = \pm 0.007, L = 160 \text{ ms}\}$	0	0	0
$\{\text{tol} = \pm 0.003, L = 256 \text{ ms}\}$	0	1	20
$\{\text{tol} = \pm 0.007, L = 256 \text{ ms}\}$	0	0	0

5.5.4. TOS_IMP Emulated Sensor Mote Validation Tests. In Table 9, the TOS_IMP event detector-emulated mote execution performances are presented for the detector parameter pairs: $pp_1 = \{\text{tol} = \pm 0.003, L = 160 \text{ milliseconds}\}$ and $pp_2 = \{\text{tol} = \pm 0.007, L = 160 \text{ milliseconds}\}$.

pp_1: inside the *speech* period of the signal set $SS2$, there are 5 events signaled and counted as TAs giving a 100% successful detection.

pp_2: inside the *speech* period of the signal set $SS2$, there are 3 events signaled and counted as TAs giving a 60% successful detection.

Additionally, Table 10 shows the event detector energy consumptions that AVRora predicts for the emulated Micaz mote. It can be seen that for both parameter pairs, pp_1 and pp_2, the CPU computations and transmission operations

Table 9: TOS_IMP emulated mote event detector performances for the event detections from the signal set SS2.

Detector parameters values	Detector event signals		
	FA	*TA*	Successful detection %
{tol = ±0.0055, L = 160 ms}	0	5	100
{tol = ±0.007, L = 160 ms}	0	3	60

Table 10: TOS_IMP emulated mote event detector energy consumptions for the event detections from the signal set SS2.

Detector parameter values	Energy Consumption (mJoules)	
	CPU	Radio Tx
{tol = ±0.0055, L = 160 ms}	438.189	575.78
{tol = ±0.007, L = 160 ms}	438.607	575.686

Table 11: TOS_IMP actual Micaz mote event detector performances for the event detections from the signal set SS2.

Detector parameters values	Detector event signals		
	FA	*TA*	Successful detection %
{tol = ±0.0055, L = 160 ms}	0	3.5	70
{tol = ±0.007, L = 160 ms}	0	3.7	80

energy expenditures do not present significant differences. On one hand, we think that this is because the L period length is equal for both event detector parameter settings. On the other hand, there is not a large difference on the packets number the mote transmits for the whole signal period.

5.5.5. TOS_IMP Actual Sensor Mote Hardware Validation Tests. Table 11 presents the TOS_IMP event detector performances for an actual Micaz mote with the detector parameter pairs: pp_1 = {tol = ±0.0055, L = 160 milliseconds} and pp_2 = {tol = ±0.007, L = 160 milliseconds}, where the results are averaged over events detected from ten trial *SS2* replays.

From these last results and the presented in Tables 8 and 9, we can observe that the emulated mote event detector behaves closer to the actual Micaz mote detector than to the OCT_IMP implementation, for the parameters pairs pp_1 and pp_2.

5.5.6. TOS_IMP Event Detector Performance for a One-Hope WSN of Micaz. In the present section, the TOS_IMP event detections are presented for a one-hop WSN of Micaz motes. The network consists of one mote data sink and one sensor mote deployed along a passage at different distances d from the test acoustic source *SS2*. The experiments are run for the detector parameter pairs: pp_1 = {tol = ±0.0055, L = 160 milliseconds} and pp_2 = {tol = ±0.007, L = 160 milliseconds}. In both cases, the event detector continuously computes event occurrences, whether

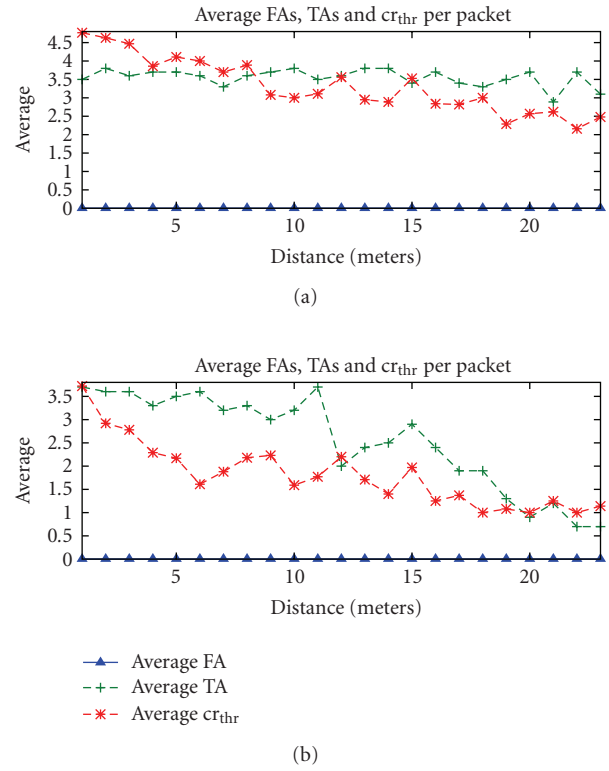

Figure 10: Micaz event detector average *FA*s, *TA*s signaled events and cr_{thr} changes per packet as function of the distance for the *speech* period of the signal set *SS2*. (a) event detector performance for the parameters {tol = ±0.0055, L = 160 milliseconds}. (b) event detector performance for the parameters {tol = ±0.007, L = 160 milliseconds}.

an event is detected, the sensor mote issues a packet that contains the mote ID, and the cr_{thr} changes counted into the respective T_D period. The event detections are averaged from the received event packets and cr_{thr} changes for ten acoustic signal *SS2* replays.

For the detector parameters pair pp_1, see Figure 10(a). It can be noted that the computed event packets average number is produced only by *TA*s inside the range of 3 to 4 along the twenty-three meters of distance d. Meanwhile, the reported per packet average cr_{thr} changes number ranges from 2 to 5. Moreover, within the distance d range 1 to 8 meters, the average cr_{thr} is larger than the average event packets number and this condition is inverted for distances equal or larger to nine meters.

In the Figure 10(b), the event detector performances are presented for the detector parameters pair pp_2. Like in the detector parameters pair pp_1 case, the detector events packets are only produced by *TA*s. Unlike, the case for the detector parameters pair pp_1, the average event packets and cr_{thr} changes number is significantly decreased from 4 to 1 along all the twenty-three meters of distance, as the sensor mote, and the acoustic signal source distance d is increased. Moreover, the average event packets number is larger than the average cr_{thr} changes number along the twenty-two meters of distance d.

Table 12: Acoustic target detection algorithms performances [18].

Detector	Energy cost	Accuracy	False positive rate
Threshold	2.71 mJ	70.5%	25.0%
High pass	13.95 mJ	84.0%	16.0%
FFT	49.12 mJ	100.0%	0.0%

As a final note, Lorincz et al. [18] reported three outdoors acoustic detectors, *threshold*, *high-pass filter* and *FFT* based, to detect marmot calls. The detectors accuracy is measured in terms of True detections (*TA*s) and False positive (*FA*s) rate percentages, and the energy consumptions are computed from data windows of 512 data. The performance results obtained by the detectors are reproduced in Table 12. In particular, the sensor mote is based on the iMote2 platform which features a Marwell PXA271 XScale processor and can run at several frequencies ranges from 13 and 416 Mhz.

Though the environments conditions are different and direct comparisons may not be applied, It can be observed that our event detector does not signal *FA*s for the acoustic signals of interest with respect to the threshold-based detector. Moreover, the change rate event detector performs better than a high-pass-based detector. Finally, it is clear that we can not compare our event detector against the FFT that additionally can identify the target signature.

With respect to the energy consumptions, the mote emulator estimates that our event detector consumes a total CPU energy amount of 438.189 mJ for the whole *SS*2 duration. If in the 20-second signal period there are 50.781 subperiods of 512 samples, then the energy consumption for one subperiod is 8.6290 mJ. It means that our detector has an energy consumption performance which is between the event detector threshold and high-pass filter based; see Table 12.

6. Conclusions and Future Research

An events detection algorithm for constrained wireless sensor hardware has been presented. The events are computed from the change of rate of two signal energy statistics values separated by an $(L-1)$ samples.

Two process results, (a) the event detector characterisation and tuning and (b) validation tests, are reported for two detector implementations, an Octave and a TinyOS. The characterisation and tuning process searches event detector parameter configurations that enable the reliable event detection from three acoustic human activities signals: *closing a door*, *dropping a plastic bottle,* and *clapping*. The validation tests demonstrate the detector capability-to-signal events from other acoustic sources, in our case the events from *people talking* signals.

The validation test results show that both event detector implementations, Octave and TinyOS which included emulated and an actual Micaz mote, succeed to detect the events from the people talking signal. Moreover, the TinyOS emulated mote approximated closer to the actual Micaz hardware performances. The results also included the event detector integration in a one-hop WSN of Micaz. In this case, we show that the events decay in a nonlinear fashion when the distance d between the acoustic signal source and the mote is increased.

6.1. Future Research Work. In the current TinyOS event detector implementation, the parameter tol is set to a fixed value. As future work, we plan to develop an adaptable event detector which modifies on the fly tol, according to the WSN application requirements, packet transmission rates, or energy consumptions.

Acknowledgment

Marco Antonio López Trinidad was supported by the National Council of Science and Technology of México (CONACyT), scholarship fellow no. 70208.

References

[1] T. Abdelzaher, L. Gu, D. Jia, et al., "Lightweight detection and classification for wireless sensor networks in realistic environments," in *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys '05)*, San Diego, Calif, USA, November 2005.

[2] D. Culler, J. Demmel, G. Fenves, S. Glaser, S. Kim, and M. Turon, "Structure monitoring using wireless sensor networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN '07)*, Cambridge, Mass, USA, April 2007.

[3] J. Johnson, J. Lees, M. Ruiz, M. Welsh, and G. Werner-Allen, "Monitoring volcanic eruptions with a wireless sensor network," in *Proceedings of the European Workshop on Sensor Networks (EWSN '05)*, January 2005.

[4] D. Culler, J. Polastre, and R. Szewczyk, "Telos: enabling ultra-low power wireless research," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, Los Angeles, Calif, USA, April 2005.

[5] O. B. Akan and L. F. Akyldiz, "Event-to-sink reliable transport in wireless sensor networks," in *Proceedings of the IEEE/ACM Transactions on Networking (TON '05)*, vol. 13, pp. 1003–1016, October 2005.

[6] L. Barolli, G. De Marco, M. Ikeda, and T. Yang, "Performance analysis of the event-reliability approach for a lattice wireless sensor network with radio irregularities," in *Proceedings of the IEEE Region 10 Conference (TENCON '06)*, vol. 4, pp. 2–116, Hong Kong, November 2006.

[7] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Regular Expressions and Languages. In Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Mass, USA, 2nd edition, 2000.

[8] D. Estrin, S. Farshchi, B. Greenstein, et al., "Capturing high frequency-phenomena using a bandwidth-limited sensor network," in *Proceedings of the 4th ACM International Conference on Embedded Networked Sensor Systems (SenSys '06)*, pp. 279–292, Boulder, Colo, USA, October-November 2006.

[9] S. Bhatti, J. Carlson, H. Dai, et al., "MANTIS OS: an embedded multithreaded operating system for wireless micro sensor platforms," *ACM Mobile Networks & Applications*, vol. 10, no. 4, pp. 563–579, 2005.

[10] A. Guitton, A. Skordylis, and N. Trigoni, "Correlation-based data dissemination in traffic monitoring sensor networks," in *Proceedings of the 2nd Conference on Future Networking Technologies (CoNEXT '06)*, Lisboa, Portugal, December 2006.

[11] N. Xu, "Implementation of data compression and 6 FFT on TinyOS," Embedded In Networks Laboratory, Computer Science Dept. USC. Los Angeles, Calif, USA, http://enl.usc.edu/~om_p/lzfft.pdf.

[12] Crossbow Technology, the mica2 platform, http://www.xbow.com/Products/productdetails.aspx?sid=174.

[13] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, pp. 364–369, UCLA, Los Angeles, Calif, USA, April 2005.

[14] O. Liang and L. Wang, "Event detection in wireless sensor networks using fuzzy logic system," in *Proceedings of the IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety (CIHSPS '05)*, Orlando, Fla, USA, March 2005.

[15] V. Mhatre and K. Papagiannaki, "Using smart triggers for improved user performance in 802.11 wireless networks," in *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services (MobiSys '06)*, Uppsala, Sweden, June 2006.

[16] S. H. Steiner, "Exponentially weighted moving average control charts with time varying control limits and fast initial response," *Journal of Quality Technology*, vol. 31, pp. 75–86, 1999.

[17] D. Culler and A. Woo, "Evaluation of efficient link reliability estimators for low-power wireless networks," Tech. Rep. UCB/CSD-03-1270, EECS Department, University of California, Berkeley, Calif, USA, 2003, http://www.eecs.berkeley.edu/Pubs/TechRpts/2003/CSD-03-1270.pdf .

[18] K. Lorincz, B. Chen, J. Waterman, G. Werner-Allen, and M. Welsh, "Resource aware programming in the pixie OS," in *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys '08)*, November 2008.

[19] A. Sharma, L. Golubchick, and R. Govidan, "On the prevalence of sensor faults in real-world deployments," in *Proceedings of the IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '07)*, 2007.

[20] D. Culler, J. Hill, S. Hollar, K. Pister, R. Szewczyk, and A. Woo, "System architecture directions for networked sensors," in *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '00)*, Cambridge, Mass, USA, November 2000.

[21] D. Lee, J. Palsberg, and B. Titzer, "Avrora: scalable sensor network simulation with precise timing," in *Proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN '05)*, Sunset Village, UCLA, Los Angeles, Calif, USA, April 2005.