Research Article

A New Movement Recognition Technique for Flight Mode Detection

**Youssef Tawk,¹ Aleksandar Jovanovic,¹ Phillip Tomé,¹ Jérôme Leclère,¹
Cyril Botteron,¹ Pierre-André Farine,¹ Ruud Riem-Vis,² and Bertrand Spaeth²**

¹ *Electronics and Signal Processing Laboratory, Institute of Microengineering (IMT), École Polytechnique Fédérale de Lausanne, Breguet 2, 2000 Neuchâtel, Switzerland*

² *Jiiva, Stadtbachstrasse 40, 3012 Bern, Switzerland*

Correspondence should be addressed to Youssef Tawk; youssef.tawk@epfl.ch

Received 8 October 2012; Accepted 1 December 2012

Academic Editor: Aboelmagd Noureldin

Copyright © 2013 Youssef Tawk et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Nowadays, in the aeronautical environments, the use of mobile communication and other wireless technologies is restricted. More specifically, the Federal Communications Commission (FCC) and the Federal Aviation Administration (FAA) prohibit the use of cellular phones and other wireless devices on airborne aircraft because of potential interference with wireless networks on the ground, and with the aircraft's navigation and communication systems. Within this context, we propose in this paper a movement recognition algorithm that will switch off a module including a GSM (Global System for Mobile Communications) device or any other mobile cellular technology as soon as it senses movement and thereby will prevent any forbidden transmissions that could occur in a moving airplane. The algorithm is based solely on measurements of a low-cost accelerometer and is easy to implement with a high degree of reliability.

1. Introduction

GSM localization or mobile phone tracking is a technology used to locate the position of a mobile phone. Localization uses the concept of multilateration of radio signals, where the phone must communicate wirelessly with at least three of the nearby radio base stations (RBSs). Knowing the position of the RBSs, and using a triangulation method, an approximation of the geographical location of the mobile phone can be calculated. This technology is based generally on four different techniques: network based, handset based, hybrid and subscriber identity module (SIM) based [1]. The SIM-based technique is of interest in this paper, where by using the SIM in mobile communication handsets it is possible to obtain raw radio measurements that include the serving cell ID, round trip time and signal strength. Different applications already use this service for localization, for example, resource tracking with dynamic distribution such as taxis, rental equipment, or fleet scheduling.

Within this context, Swisscom AutoID Services (SIS) in collaboration with La Poste Suisse aims to pioneer a service

that will allow clients to be able to track and trace their packages in near real-time mode anywhere in the world. The service will consist of including a tracker with the package that operates on the GSM network for location and communication [2]. Therefore, a key issue arises when the package is transported via airplane. Indeed, it is well known that aircraft remains one of the few places where the use of mobile communication signals is prohibited [3, 4]. In fact, the aircraft is not a good Faraday cage and cannot prevent transmissions to reach terrestrial cellular networks. Moreover, there is a possibility of interference with some avionics instruments in the aircraft. As a result, international regulation bodies have forbidden the use of mobile phones onboard an aircraft and hence any module used for tracking a package should be turned off as soon as it is located inside an airplane. In order to respect these regulations, the main idea we develop in this paper is to consider the use of low cost MEMS accelerometers, such as those found in today's smartphones, to allow the tracker to detect the movement and to turn off the GSM module automatically without any human intervention. For the envisioned application, this

means that once a shipment equipped with a tracker is in the moving aircraft, the device will detect the aircraft's movements; it will automatically switch off the GSM module until the aircraft has come to a halt after landing and the device detects a static condition, allowing it to turn on the GSM module again.

In the recent literature, several papers already proposed the use of smartphone accelerometers for movement recognition. For example in [5], the authors investigated the suitability of the built-in smartphones accelerometer to provide good accuracy for common human activity recognition. In [6], the authors introduced a new method to implement a motion recognition process using a mobile phone fitted with an accelerometer to effectively recognize different human activities with a high-level accuracy. Human movement detection using an accelerometer was also the subject for several other papers [7–11]. The accelerometer was also used for several types of movement detection, for example, sign language detection [12], handwritten digit recognition [13], fast fall detection [14], and wireless motion sensing [15]. In these cases, the detection is based on different techniques and the degree of algorithm complexity varies depending on the application, alongside with the quality of the accelerometer. However, none of the above papers considered the use of smartphones' accelerometers for flight mode detection. Therefore, this paper fills a gap in the current literature by proposing a new movement recognition algorithm specifically tailored for flight mode detection, that is, taking in consideration the reliability requirements that must be satisfied for civil aviation applications. In particular, the proposed algorithm is specially conceived for detecting any movement that a package can be subject to inside an airplane, with a sufficiently high reliability to respect all the specifications and requirements from international airspace regulations. The analytical and statistical studies of the proposed algorithm are based on raw acceleration measurements acquired from static data and from dynamic data onboard different flights. These measurements are used to define the parameters required by the algorithm to detect either static or dynamic condition. Furthermore, the reliability is analyzed in terms of probability of misdetection and false alarm. In addition, implementation complexity of the algorithm is evaluated in terms of response time and resources, and finally power consumption considerations are presented.

It is important to note that the main contribution of the paper is not in devising the methods used in the algorithm, but instead results from the combination of these methods in one algorithm that is tailored to provide a highly reliable movement detection technique using low cost MEMS accelerometers. Moreover, the proposed algorithm is not intended to classify the source of movement (e.g., airplane, train, car, truck, etc.), as this would decrease the reliability of correctly detecting movement while in an airplane. Indeed, it is sufficient for the considered package tracking application to only get information about the position of the package when it is not moving and prohibit transmissions otherwise. Finally, the paper is not mainly involved from the theoretical point of view, but it has a clear engineering application while emphasizing the use of real acceleration measurements. For

TABLE 1: Main characteristics of LIS302DL accelerometer [16].

Noise (1σ)	2.23 mg
Measurement range (g)	± 2 or ± 8
Output data rate (Hz)	100 or 400
Temperature range	-40°C to $+85^{\circ}\text{C}$
Power consumption	<1 mW

more details about the final product where the proposed algorithm is part of, the interested reader is referred to [2].

2. Measurement Data Collection

In order to analyze the accelerometer behavior under different conditions a database was created including acceleration measurements from different experiments consisted of real flights, of different duration and in different airplanes. An accelerometer is a device that measures the acceleration associated with the phenomenon of weight experienced by a test mass that resides in the frame of reference of the accelerometer device. The accelerometer chosen for conducting the analyses in this paper is the LIS302DL accelerometer, the same sensor as the one integrated inside an iPhone [16]. This accelerometer is an ST ultracompact low-power three-axis linear accelerometer. Its main characteristics are shown in Table 1.

The raw measurements were collected by putting an iPhone, which is set to flight-mode, inside a handbag and taking it onboard different flights. The X, Y, and Z measurements from the built-in accelerometer are saved using an application on the phone with a sampling frequency of 100 Hz. The recorded measurements are stored later using a data logging software in a database with a resolution of 16 bits. The measurements were taken before, during, and after every flight. In addition, with each series of flight data, there is a metadata description containing the flight duration time, the take-off time, landing time, as well as description about the turbulences, and other conditions occurring during the flight and all other data necessary to detect transitions and unusual events. The metadata were collected through the observations of the person holding the accelerometer onboard the flight. Hence, it has a couple of seconds of time accuracy, which is not critical because the proposed algorithm is independent of this information that is only used to better understand the behavior of the accelerometer. An example of a raw measurement collected is displayed in Figure 1, which shows the accelerometer three-axis output before, after, and during a flight from Geneva to London onboard an Airbus A319. The flight time is defined as the time from the push-back manoeuvre before takeoff until the time the plane engines are turned off after landing, which is the time when the GSM module should not transmit any signals. It can be noted that outside the flight time the accelerometer axes orientation is often changing and the measurements are so noisy. This is due to the fact that the used iPhone was inside a handbag that is subject to all types of movements that occur during normal activities of a travel.

FIGURE 1: Acceleration measurements.

Static data was also collected when the accelerometer was not moving for a long period of time and stored in the database.

The first step in order to interpret the data collected was to identify the metadata provided with each set within the sequence of acceleration measurements. For example, for the flight above (to be considered throughout the paper), the sudden sharp jumps during flight time are due to strong turbulences. Also, the different parts of the takeoff manoeuvre are shown in Figure 2 where each part is identified based on the metadata provided. This helped to better understand the behavior of the accelerometer under different events. For instance, it can be seen that once the “aircraft push back maneuver” starts and the engines are turned on, the accelerometer outputs start to fluctuate and we can see immediately the change in the static condition of the sensor. Also the acceleration of the plane in the runway and the liftoff are identified easily as the amplitude of the acceleration on the 3 axes changes suddenly.

This short overview on the raw acceleration measurements shows that as soon as the sensor senses a movement, the amplitude and the variance of the accelerometer outputs increase. Bearing this in mind and computing a parameter that is dependent on these two values can provide a criterion to find the status of the sensor. Consequently, we chose the moving variance (MV) of the incoming measurements on the three axes as this parameter. But before analyzing the collected data with respect to this parameter, a short overview of the MV is given in the next section.

3. Moving Variance

A moving variance is used to analyze a set of data points by creating a series of variances of different subsets of the full data set [17]. Given a series of numbers and a fixed subset

FIGURE 2: Takeoff metadata identification where the three traces represent the 3-axis acceleration during takeoff.

size, the MV can be obtained by first taking the variance of the first subset. The fixed subset size is then shifted forward, creating a new subset of numbers, and consequently a new variance is computed. This process is repeated over the entire data series. The plot line connecting all the (fixed) variances is the moving variance. Thus, an MV is not a single number, but it is a set of values, each of which is the variance of the corresponding subset of a larger set of data points. An MV may also use unequal weights for each data value in the subset to emphasize particular values in the subset.

In this paper, two types of MV computations are used: (1) the normal moving variance (NMV) and (2) the exponential moving variance (EMV) also known as the exponentially weighted moving variance. The difference between the two is that the NMV calculates the variance of the fixed subsets without any weighting, in other words all the data in a specific subset have the same importance. As for the EMV, it applies weighting factors which decrease exponentially [17]. The weighting for each older data point decreases exponentially, never reaching zero. Alternatively, the EMV gives more weighting to the new data in the computation of the subset variance.

The formula for calculating the NMV is given by [17]:

$$\begin{aligned}\sigma_{\text{NMV}}^2(n) &= \frac{1}{N} \sum_{k=n-N+1}^n [x(k) - \bar{x}(n)]^2 \\ &= \left[\frac{1}{N} \sum_{k=n-N+1}^n x^2(k) \right] - \bar{x}(n) = \overline{x^2}(n) - \bar{x}(n),\end{aligned}\quad (1)$$

where $\bar{x}(n)$ corresponds to the average of the subset, $\overline{x^2}(n)$ is the sum of the squares over the number of samples in the subset N , and n is the subset's index number. To reduce

FIGURE 3: NMV moving variance computation.

the amount of addition and multiplication and hence lower the computational load, data processing can be applied. So, instead of calculating in each subset the sum of the squares and the square of the sum for all the samples over a fixed window length, the two sums are calculated only for the first subset and kept in a buffer along with the samples in the subset. Moreover with the upcoming of a new sample (i.e., moving to a new subset), the sample and its square are added to the two sums computed before and the old sample and its square are subtracted and a new variance is calculated. Thus $\bar{x}(n)$ and $\bar{x}^2(n)$ can also be expressed as

$$\bar{x}(n) = \frac{x(n) - x(n - N + 1)}{N} + \bar{x}(n - 1), \quad (2)$$

$$\bar{x}^2(n) = \frac{x^2(n) - x^2(n - N + 1)}{N} + \bar{x}^2(n - 1). \quad (3)$$

At the next iteration, the old sample is dropped from the buffer and the new sample replaces it. This operation is repeated at every new measurement (i.e., overlapping windows), and a new moving variance is computed as shown in Figure 3.

The EMV uses the same overall concept, but it differs in two main points: (1) it does not require the storage of all the samples of a subset, but only the last result and (2) it gives more weighting to new measurements. The EMV computation can be summarized as follows: there are two variables that need to be saved in the memory, the average and the variance of the samples within a subset. With the arrival of a new sample (i.e., moving to a new subset) the old average and variance are weighted and a new average and variance are computed taking in consideration the new sample. This way the samples do not need to be saved in the memory in a subset and consequently all what is needed for memory are two variables, the previous average and the variance. The formulas to calculate the EMV are given by [17]

$$\bar{x}_{\text{EMV}}(n) = \frac{(N - \alpha) \bar{x}_{\text{EMV}}(n - 1) + \alpha x(n)}{N}, \quad (4)$$

$$\sigma_{\text{EMV}}^2(n) = \frac{(N - \alpha) \sigma^2(n - 1) + \alpha (x(n) - \bar{x}_{\text{EMV}}(n))^2}{N}, \quad (5)$$

where α is the weighting factor, whose value depends on many factors as it will be seen in the following sections. In the following, both NMV and EMV of the raw acceleration measurements are computed, and performance comparison of these two methods are provided.

4. Acceleration Measurement Analysis

4.1. Static Mode. Our analysis of the recorded data started with the analysis of the static measurements of the accelerometer. Around 35 hours static measurements were collected in order to find the noise behavior of the accelerometer. Figure 4 shows the power spectral density of each axis where it is clear that the Y-axis is the axis sensing the gravity presenting a higher amplitude component around 0 Hz. It can be seen that the behavior of the 3 axes in a static condition is very similar and can be approximated as an independent white noise with peak-to-peak amplitude around 0.01 mg, and a variance around $5 \times 10^{-6} \text{ g}^2$.

The NMV and EMV for the 3 accelerometer axes are shown in Figure 5. It can be seen that the NMV responses on the 3 axes are similar to the noise variance of the accelerometer and approximately equal to $5 \times 10^{-6} \text{ g}^2$. As for the EMV, it can be noted that for the weighting factors of 1 and 5, the static behavior is very similar to the case of NMV and hence it is consistent with the noise level of the accelerometer. For a weighting factor of 10, we can see that the EMV has a slightly noisier response. This is due to the fact that the EMV gives more weighting to the new samples and in this case for a high weighting factor, the past information of the variance is less weighted than the new information that is added, and therefore, there is a loss in the smoothing effect on the variance computation.

Based on the results shown above, it can be seen that the output of the NMV and EMV in static mode can approximately provide the noise level of the sensor, and this value can be taken as a reference later to decide whether the accelerometer is in a static or dynamic mode.

4.2. Dynamic Mode. As in the static case, the PSD of the acceleration measurements was first computed in order to check if a low quality sensor as the one used for the tests herein could detect small movements and more specifically the vibrations of an airplane's motor. The power spectral densities of the measurement norm of the three accelerator axes during flights onboard different airplanes are shown in Figure 6. It is interesting to notice the existence of frequency peak components between 10 Hz and 30 Hz during all flights' stages. These peaks can be identified as the vibration frequency of the airplane and typically they characterize the vibration frequency of the motors. Analyses of different series with different types of planes show that most of the aircraft vibration frequencies and dynamic frequencies are within the band ranging from 0 Hz up to ~30 Hz. Therefore, the sampling frequency that has been used for the analysis (100 Hz) could be lowered down to 60 Hz if necessary. Thus, decreasing the sampling frequency in between 60 and 70 Hz seems feasible, and within this range all the vibration and dynamic frequencies are preserved.

Regarding the NMV and EMV of the dynamic flight mode, Figure 7 shows the NMV computation for the 3 accelerometer axes of the flight mentioned in Section 2. The existence of peak components with high amplitudes that represent sudden movements of the sensor can be noticed. A

FIGURE 4: Power spectral density of the raw acceleration measurements in a static case.

closer look at the takeoff and landing phases during the flight is shown in Figure 8.

During the takeoff, it can be seen that as soon as the engines are on, the NMV outputs on the 3 axes increase, and also it can be noted the existence of high peaks during the plane acceleration and liftoff. During the landing, it can be seen that the NMV outputs have high amplitude peak components and, as soon as the engines are off, the outputs on the 3 axes decrease momentarily. Results of the EMV are similar to those of NMV and hence are not shown here.

Overall, these graphs show that both the NMV and EMV are able to detect movement of the sensor under different conditions. As a result, it can be concluded that by using one of these two methods, the state of the sensor can be found. Thus, the proposed algorithm for movement detection is based on these analyses and described in the next section.

5. Proposed Algorithm

The proposed algorithm for the detection of static/dynamic condition of the sensor is based on the computation of the moving variance of the raw accelerometer measurements as described before. The overall architecture is illustrated in Figure 9 and was implemented in Matlab in a first step. The algorithm can work in real time directly on measured data coming from the accelerometer, but as the analyses

conducted during this work were limited to offline data processing, the measurements were taken from the database and postprocessed.

The algorithm starts by reading measurements during a specific window size (W_s) of time and computes the moving variance (MV) of the measurements for each accelerometer axis separately:

$$MV_i(n) = f(a_i(n-N), a_i(n-N+1), \dots, a_i(n)), \quad (6)$$

where i represents the X, Y, or Z axis, N is the window size, f stands for a normal (NMV) or exponential (EMV) moving variance computed in (1) or (3), n is the subset's index number where the MV is computed, and a is the acceleration measurement during the specified period of time. Afterwards, two conditions are performed to set the status of the sensor. The first one is to check periodically every W_s second if the MV of each axis is smaller than a predefined threshold, as shown by

$$\text{cond}_\sigma^n = (MV_x(n) < T_\sigma^x) \text{ AND } (MV_y(n) < T_\sigma^y) \text{ AND } (MV_z(n) < T_\sigma^z), \quad (7)$$

where $\sigma_{m,th}^i$ is the MV threshold on the i axis. If this is not fulfilled, the condition is set to "nonstatic" and the next window measurements are read. Otherwise, the condition is

FIGURE 5: NMV and EMV of the 3 accelerometer axes in a static condition for a window size of 1 s or $N = 100$ samples.

set to a “temporary static condition” and another threshold related to time is being checked. This second threshold is performed in order to obtain a more secure and stable response. In fact, once a “temporary static condition” is declared, the algorithm goes backwards for a specific duration of time and examines if the status of the sensor within this duration was also “temporary static condition”

$$\text{cond}_{\text{time}}^n = (\text{cond}_{\sigma}^n) \text{ AND } (\text{cond}_{\sigma}^{n-1}) \text{ AND } \dots (\text{cond}_{\sigma}^{n-T_{\text{th}}}), \quad (8)$$

where T_{th} represents the threshold in time or the duration of time in which the algorithm will look backward for a temporary static condition. If this is not fulfilled, a “nonstatic” condition is declared and the next window measurements are read. If the condition is fulfilled, the algorithm outputs a “static condition” and consequently at this time the module is allowed to transmit a wireless mobile communication signal if needed. T_{th} typically ranges from 1 up to several seconds. For example, if the threshold in time is set to 3 seconds and the MV condition is checked every 1 second, then in case of a “temporary static condition,” the algorithm checks the state of the previous three conditions. If all the three of them were

of “temporary static condition,” then the sensor is considered to be in a static state. Otherwise, the sensor is considered to be in dynamic state.

5.1. Probability Analysis and MV Threshold Determination. Our movement recognition algorithm can be seen as a binary detection problem, that is, deciding between two hypotheses whether the sensor is in a static or a dynamic mode. As a result, a statistical test, more specifically a binary hypothesis-testing problem [18], is set up. The two hypotheses are defined as H_0 where the sensor is in a dynamic mode and H_1 where the sensor is in a static mode. A binary test of H_0 versus H_1 takes the following form:

$$\phi(n) = \begin{cases} 0 \sim H_0, & \text{cond}_{\text{time}}^n \text{ is false,} \\ 1 \sim H_1, & \text{cond}_{\text{time}}^n \text{ is true.} \end{cases} \quad (9)$$

This equation can be read as the test function $\phi(n)$ equals 1, that is hypothesis H_0 is rejected and H_1 is accepted, if $\text{cond}_{\text{time}}^n$ is true (i.e., the MV on each of the accelerometer axes is smaller than the corresponding MV threshold during the duration of time specified by T_{th}). Otherwise (i.e., if at

FIGURE 6: Single-sided amplitude spectrum of the accelerometer measurements norm during flights onboard different airplanes types.

least the MV of one of the accelerometer axes is higher than the MV threshold), $\text{cond}_{\text{time}}^n$ is false and the test function equals zero, that is hypothesis H_1 is rejected and H_0 is accepted. Figure 10 shows the binary hypothesis test and its corresponding probabilities.

If the sensor is static, then the probability of detection P_D is defined as the probability that $\text{cond}_{\text{time}}^n$ is true and H_1 is accepted (i.e., detecting final static condition), and the probability of misdetection P_M is defined as the probability that $\text{cond}_{\text{time}}^n$ is false and H_0 is accepted (i.e., detecting dynamic condition). They can be expressed as

$$\begin{aligned} P_D &= P_{H_1} [\phi(n) = 1], \\ P_M &= 1 - P_D = P_{H_1} [\phi(n) = 0]. \end{aligned} \quad (10)$$

If the sensor is dynamic, then the probability of false alarm P_{FA} is defined as the probability that $\text{cond}_{\text{time}}^n$ is true and H_1 is accepted (i.e., detecting final static condition). It can be expressed as

$$P_{FA} = P_{H_0} [\phi(n) = 1]. \quad (11)$$

Note that our focus is on selecting a threshold that will lead to a sufficiently low P_{FA} that is compliant with the civil aviation regulations. For our particular application, the maximization of P_D (and thus minimization of P_M) is not so critical, as

long as static condition detection can still happen regularly to allow periodic location determination.

In the next two subsections, we seek to determine the probability density function for the NMV and EMV outputs in static and in dynamic mode, respectively, in order to apply the well-known classical detection theory (see, e.g., [18]) to theoretically determine a suitable threshold.

5.1.1. Probability of Misdetection (P_M). As shown in Section 4.1, the behavior of the 3 accelerometer axes while it is static is similar, with the variance on each axis approximately equal to $\sigma^2 = 5 \times 10^{-6} g^2$. The NMV and EMV outputs in this case are also very similar. Therefore, it is possible to define the same MV threshold for the three axes, that is,

$$T_\sigma^x = T_\sigma^y = T_\sigma^z = T_\sigma. \quad (12)$$

This threshold should be chosen as a function of two parameters: the noise behavior of the accelerometer and the probability of misdetection P_M , which is related to the distribution of the accelerometer measurements while it is static. In order to find the MV threshold, the sensor should be set in a static mode, and the subsequent approach can be followed:

- (1) measure the noise variance on each axis of the accelerometer,

FIGURE 7: NMV computation for the 3 accelerometer axes of the flight from Geneva to London.

- (2) derive the distribution of the MV measurements on each axis,
- (3) set the probability of misdetection P_M ,
- (4) knowing the distribution of the MV measurements, then the theoretical formula of PM can be derived, and an MV threshold can be computed in function of the noise variance.

Once an MV threshold is computed, the correspondent P_{FA} should be derived to check if it is below the maximum value defined by the civil aviation regulations. If not, then P_M is increased and a new MV threshold is computed.

In the case of the accelerometer chosen in this paper [16], the output of the NMV or EMV algorithm applied to nonoverlapping datasets of the static accelerometer raw measurements can be approximated as independent and identically distributed (i.i.d.) with a Gaussian distribution, as can be seen in Figure 11 that depicts the probability density function (pdf) and the cumulative density function (cdf) for the Y axis. The measurements for the two other axes and from other static data sets have very similar probability distribution as the one plotted here and are thus not shown.

The probability of misdetection P_M in this case can be written as [18]

$$\begin{aligned}
 P_M &= P_{H_1} [MV \geq T_\sigma] \\
 &= 1 - \theta(T_\sigma) \\
 &= \frac{1}{2} \left[1 - \operatorname{erf} \left(\frac{T_\sigma - \mu}{\sqrt{2\sigma_{MV}^2}} \right) \right], \tag{13}
 \end{aligned}$$

where μ , σ_{MV}^2 , and $\theta(x)$ are the mean, the variance, and the cdf of the MV distribution, and erf is the error function.

TABLE 2: Probability of misdetection versus different MV thresholds.

T_σ	P_M	Time _{Error}
$2\sigma^2$	6×10^{-3}	166 s
$2.5\sigma^2$	8×10^{-6}	35 hours
$3\sigma^2$	2×10^{-11}	1585 years
$4\sigma^2$	6×10^{-19}	5×10^{10} years

Table 2 shows the probability of misdetection for different MV thresholds, where Time_{Error} corresponds to the period of time for the algorithm to missdetect a dynamic condition if the MV is checked every 1 s, that is Time_{Error} is equal to $1/P_M$.

It can be seen that for a threshold of $2.5\sigma^2$ or above, P_M is already very small, yielding sufficiently large values for Time_{Error} (considering our application). We now look at the threshold requirements to yield a sufficiently small probability of false alarm when we are in dynamic mode, starting with the above thresholds as well as an additional one of $5\sigma^2$.

5.1.2. Probability of False Alarm (P_{FA}). To compute P_{FA} , the temporary probability of false alarm (i.e., to detect a temporary static condition) P_{FA}^t should first be derived. P_{FA}^t is defined as the probability that at a specific time the computed MV outputs of the three accelerometer axes during a dynamic condition are lower than T_σ . As the noise on the 3 axes is uncorrelated (i.e., consecutive MV output measurements and measurements from different axes are assumed independent since we assume that they are obtained from nonoverlapping datasets of the static accelerometer raw measurements), then P_{FA}^t and P_{FA} can be expressed as

$$\begin{aligned}
 P_{FA}^t &= P_{H_0} [MV \leq T_\sigma] = P_{FA}^x P_{FA}^y P_{FA}^z, \\
 P_{FA} &= (P_{FA}^t)^{T_{th}/W_s}, \tag{14}
 \end{aligned}$$

where P_{FA}^i is the probability that the computed MV on the “i” axis is lower than T_σ during a dynamic condition. In order to compute these probabilities, the distribution of the NMV and EMV outputs in dynamic conditions should be found. However, by analyzing all the collected measurements, we could neither determine a theoretical nor an empirical model for the dynamic mode (such a model would need to depend on many parameters such as the type of aircraft, the weather, the packaging of the sensor, etc.). Therefore, we could not provide an analytical expression for the MV threshold as a function of the desired probability of false alarm. Thus, we decided to proceed empirically as follows:

- (i) for each data set, only the part of the series where the sensor is subject to a dynamic movement is considered and used to compute the EMV and NMV;
- (ii) then the corresponding pdf and cdf of the EMV and NMV outputs are computed;
- (iii) finally, the probability that the NMV or EMV outputs of each axis are lower than the threshold is computed.

FIGURE 8: NMV computation for the 3 accelerometer axes during landing and takeoff of a flight from Geneva to London.

Following this approach, P_{FA} was empirically computed for different MV thresholds and thresholds in time.

5.2. Threshold in Time. As it was described above, the threshold in time T_{th} stands as a second condition after testing the MV on the three axes. In fact, the threshold in time ensures that the sensor was static for the period of time that is specified, and once this period is exceeded, a final static condition is declared that can be used as an indication for the allowance of transmission of mobile communication signals. The main reason behind this second condition is to decrease P_{FA} in order to meet the specific requirements for the civil aviation regulations, which is the most critical point especially during flight time. Similar to the MV threshold, we analyzed empirically different threshold in times on different measurements from different flights, in order to find a minimum threshold in time that ensures a reliable performance of the algorithm under different conditions for the chosen accelerometer.

5.3. Impact of Window Size (W_s). The window size (W_s) is defined as the duration of time where the algorithm is computing the MV and the period at which cond_σ^n is checked. If the accelerometer measurement rate is 100 Hz and W_s is equal to 1 s, it means that the MV over 100 samples is computed and compared to the MV threshold every 1 s. When decreasing the window size, the MV bandwidth is increased and the noise also increases. Consequently, the threshold should be increased. Table 3 shows the MV average for different window sizes considering static measurements. For 1, 2, or 3 s window size, the moving variance is almost the same; meaning that no significant additional information will be gained when increasing the window size more than 1 s for static data. For 0.5 s the response starts to be slightly noisier and in this case the threshold should be slightly increased. Overall, a small change in window size does not have a great

TABLE 3: MV average for static data for different window sizes.

Window size (W_s)	0.5 s	1 s	2 s	3 s
MV $\cong$	0.6	0.5	0.51	0.51

impact on the moving variance of the static condition and hence the MV threshold is not so affected.

Another aspect that should be taken in consideration is the effect of different window sizes on the MV during dynamic conditions. Figure 12 shows the acceleration measurement on one axis and its corresponding NMV computation. In the graphs on the left, it can be seen that when the window size is increased, the moving variance becomes less sensitive to dynamic change. Also, a smaller window size results in a noisier response and in this case it would be harder to detect a static condition. However, one exception is noticed regarding this point. In fact, it can be seen from the graphs on the right of Figure 12 that there is an opposite behavior when increasing the window size. Normally, for dynamic movements resulting in an acceleration measurement having sudden jumps with a very short duration (a), the NMV output for a 1 s window size (c) is higher than for a 3 s window size. This is expected, because for a window size of 1 s the variance is computed using less information and any short duration movement within this 1 s can have more effect than in the case of a 3 s window size where more information are taken in consideration. In a different case, when a movement occurs resulting in an acceleration having a jump that lasts more than 3 s (b), it can be seen that a window size of 3 s results in a higher variance (d). In general, it has been noticed that this type of movement rarely occurs and even in this case a 1 s window size can still detect it.

Overall, it is observed that varying the window size has a limited impact on the MV threshold and a general value for different window sizes can be used. Regarding sensitivity to

FIGURE 9: Proposed algorithm for detecting the status condition of the sensor.

FIGURE 10: Binary hypotheses for the movement recognition algorithm and its corresponding probabilities.

dynamic movement, it is shown that increasing the window size results in a variance that is less sensitive to dynamic detection and decreasing it results in a variance less sensitive to static detection. As a tradeoff between these two results, we have found that a window size between 1 and 2 s works well.

6. Simulation Results

For testing and optimization, the algorithm is implemented in Matlab where simulation results are obtained using measurements from all the flights datasets. Hereafter, only results from two selected flights are shown for illustration and comparison purposes.

- (i) Flight 1, which is the flight from Geneva to London already described in Section 2, is chosen as a representative of a typical flight.
- (ii) Flight 2, from Helsinki to Zurich onboard a Bae Avro RJ85, is chosen because it was the most challenging flight for detecting dynamic conditions during flight time.

Figures 13 and 14 show the algorithm detection output applied on measurements from both flights using NMV

FIGURE 11: pdf and cdf of NMV (or EMV) output in static condition.

FIGURE 12: NMV behavior under different dynamic conditions.

FIGURE 13: Algorithm detection output using NMV and 5 s threshold in time for flight 1.

method for different MV thresholds and a 5 s threshold in time. The magenta peaks in the bottom of the figures represent the “temporary static condition” and the black peaks represent the final “static condition.” What can be observed is that the second condition (time duration of static period upon which it is decided if the condition is static or dynamic) filters the first condition and as expected improves the performance of the algorithm. For flight 1, it can be noted that the algorithm detects the movement of the sensor with high precision, and during all the flight time it always outputs a dynamic condition up to an MV threshold of $4\sigma^2$. For an MV threshold of $5\sigma^2$, the algorithm outputs a false temporary static condition before landing. For flight 2, the algorithm starts to detect false temporary static condition during flight time for a MV threshold of $4\sigma^2$ or higher, and false final static condition for a MV threshold of $5\sigma^2$. These two graphs show that the maximum MV threshold needed for detecting a temporary static condition during flight time is around $3\sigma^2$ and as soon as the MV threshold reaches $4\sigma^2$ the false temporary static conditions start to appear. At this MV threshold level or higher, the importance of the threshold

in time to detect or not a final static condition becomes crucial. For flight 1, it can be seen that even with increasing the MV threshold up to $5\sigma^2$, no final static condition is detected; however, for flight 2, at $5\sigma^2$ the algorithm starts to detect false final static conditions for a 5 s threshold in time. The significance of T_{th} is better shown in Figure 15 that displays the algorithm detection output for flight 2 for different thresholds in time and a $4\sigma^2$ MV threshold. It can be noted that for a T_{th} smaller than 4 s, the algorithm detects falsely final static conditions during flight time. And for 4 s and higher, no final static condition is declared and hence no transmission is allowed. These results were confirmed by analyzing a huge number of flight hours where similar results were obtained and the MV threshold of $4\sigma^2$ was found to be the maximum threshold allowed for a reliable performance of the algorithm with a corresponding threshold in time of 4 s.

A deeper look of the algorithm behavior using these two thresholds is shown in Figure 16 during boarding time for flight 1. The right graph shows the NMV output of the 3 accelerometer axes and the two dimensions check (i.e., amplitude and time). Once both conditions are satisfied,

FIGURE 14: Algorithm detection output using NMV and 5 s threshold in time for flight 2.

the algorithm outputs a static condition where it can be seen as the black peak in the left graph.

The output of the algorithm using EMV method is similar when using NMV for small weighting; however, when increasing the weighting factor the EMV starts to be more sensitive to static condition detection. In fact when increasing the weight, and consequently increasing the importance of new measurements, the response of the algorithm starts to be more fluctuating. This can be explained by looking at Figure 17, where it is shown the NMV and EMV outputs for a measurement resulting from a dynamic movement. It can be seen that for EMV, with increasing the weighting, the amplitude variation increases; this is because the new sample has more weight, and if it represents a sample coming from a dynamic movement it will increase the output variations. As for the NMV where there is no weighting, the variance is more stable as it results from an average of data having similar weight. Therefore, the effect of increasing the weight too much in the EMV method has a disadvantage. For example, if the flight is very calm, then the MV output could potentially be close to the MV threshold, and if the weight is increased, then the EMV output will be noisier and there is a possibility due to the variation that it crosses the threshold and thus the

algorithm outputs a static detection during the flight. This type of scenario should be avoided, and this is why it is not recommended to increase the weight above 10.

Regarding the probability of false alarm, and following the approach in Section 5.1.2, Tables 4 and 5 show the temporary probability of false alarm and P_{FA} during flight time for flights 1 and 2. It can be seen that the performance of the algorithm varies from one flight to another, and in both cases P_{FA} is considerably low. Also the table shows that if a lower P_{FA} is required, it is sufficient to either decrease the MV threshold or increase the threshold in time. Probability analyses from other flights were similar to the assessment discussed in this paper and the algorithm showed a high fidelity in detecting the state of the accelerometer.

To conclude, this section shows that the proposed algorithm has a very precise detection of static/dynamic conditions of the sensor using EMV or NMV for a MV threshold less than or equal to $4\sigma^2$. It has been also shown that the detection performance is very good especially during flight time where P_{FA} is very low. For EMV it is not recommended to increase the weighting more than 10, as the detection output starts to be biased by the weighting factor on the new measurement and can lead to false detection.

FIGURE 15: Algorithm detection output using NMV and $4\sigma^2$ MV threshold for flight 2.TABLE 4: P_{FA} during flight time for flight 1 using NMV method.

Flight	1					
T_σ	$3\sigma^2$		$4\sigma^2$		$5\sigma^2$	
T_{th} (s)	3	4	3	4	3	4
P_{FA}^x	5×10^{-2}	5×10^{-2}	8×10^{-2}	8×10^{-2}	1×10^{-1}	1×10^{-1}
P_{FA}^y	7×10^{-4}	7×10^{-4}	2×10^{-3}	2×10^{-3}	4×10^{-3}	4×10^{-3}
P_{FA}^z	9×10^{-7}	9×10^{-7}	2×10^{-6}	2×10^{-6}	2×10^{-4}	2×10^{-4}
P_{FA}^t	2×10^{-10}	2×10^{-10}	3×10^{-10}	3×10^{-10}	8×10^{-8}	8×10^{-8}
P_{FA}	6×10^{-33}	1×10^{-42}	3×10^{-29}	1×10^{-38}	5×10^{-22}	4×10^{-29}

7. Implementation Analysis

The implementation of the algorithm is assessed using three criteria: the response time, resources, and practical considerations.

7.1. Response Time. An important point to study is the response time of the algorithm, which we define as the time required for the algorithm to output a static condition at turn-on assuming that the sensor is static. Indeed, minimizing this response time will also extend the battery life.

For the NMV algorithm and assuming a window size of 1 s, it will first take 1 s for the data window to be full

of measurements before the algorithm can output the first temporary condition. Assuming a threshold in time of 4 s, the final static condition will be generated after 4 s.

For the EMV algorithm, the behavior is different and depends on the initialization state of the accelerometer before startup. For example, Figure 18 shows the EMV outputs for different weightings compared to the NMV output (i.e., no weighting) for one axis of an accelerometer in static mode when the algorithm is initialized with an acceleration variance of $10\sigma^2$ (this value is chosen to guarantee that the algorithm will not provide a temporary static condition at startup). The cyan dotted line represents a MV threshold of $4\sigma^2$.

FIGURE 16: Algorithm behavior during boarding. Left side is the algorithm output in comparison to the acceleration on the 3 axes. Right side is the corresponding NMV outputs.

TABLE 5: P_{FA} during flight time for flight 2 using NMV method.

Flight	2					
T_σ	$3\sigma^2$		$4\sigma^2$		$5\sigma^2$	
T_{th} (s)	3	4	3	4	3	4
P_{FA}^x	3×10^{-1}	3×10^{-1}	6×10^{-1}	6×10^{-1}	7×10^{-1}	7×10^{-1}
P_{FA}^y	1×10^{-2}	1×10^{-2}	1×10^{-1}	1×10^{-1}	4×10^{-1}	4×10^{-1}
P_{FA}^z	7×10^{-3}	7×10^{-3}	7×10^{-2}	7×10^{-2}	2×10^{-1}	2×10^{-1}
P_{FA}^f	2×10^{-5}	2×10^{-5}	4×10^{-3}	4×10^{-3}	6×10^{-2}	6×10^{-2}
P_{FA}	9×10^{-15}	2×10^{-19}	7×10^{-8}	3×10^{-10}	2×10^{-4}	1×10^{-5}

It can be noted that with increasing the weighting, the response time decreases as expected and consequently the output of the EMV algorithm reaches faster the MV threshold than the NMV. It is also important to note that increasing the variance of the initial dynamic condition will increase the response time. Therefore this factor should be chosen carefully to guarantee a shorter response time while minimizing the probability of false alarm.

7.2. Implementation Resources. The NMV and EMV algorithms have shown similar results in terms of performance. In this section, we explore the hardware implementation of these two algorithms in order to select the most appropriate one.

7.2.1. NMV Implementation. The basic element of the implementation of the NMV method is a moving averager (MA) shown in Figure 19 and its z -domain transfer function is

$$H(z) = \frac{1}{N} (1 + z^{-1} + z^{-2} + \dots + z^{-N+1}). \quad (15)$$

This element is then used to compute the NMV of a sequence as defined by (1) and depicted in Figure 20.

7.2.2. EMV Implementation. The same procedure is used for the EMV method. The basic element of the implementation

is an exponential averager (EA) which is defined by (4) and depicted in Figure 21. This element is then used to compute the EMV of a sequence as defined by (5) and depicted in Figure 22.

7.2.3. Optimization and Comparison. It can be seen that the implementation of these two algorithms requires only four different operations: (1) delay; (2) addition/subtraction; (3) multiplication by a constant; (4) squaring. When an adder has one input which is the delayed version of its output, as it is the case in the MA and EA, this is equivalent to an integrator. Table 6 summarizes the resources required by these operations in terms of logic cell available in CPLDs and FPGAs, when the input signal of the operation has a resolution of R bits.

The implementation of the multiplication by a constant is highly dependent on the value of the constant. If the constant is a power of two, it corresponds to a simple shift and does not require resource at all. If this rule is too strict, it is also possible to implement the multiplication using two shifts and a subtraction when the constant is in the form $2^L - 2^M$ [17]. Table 7 summarizes the total resources required by the algorithm, when the resolution of the input signal is R bits, the number of samples in a MA or EA is N , and considering that the multiplications are all implemented by shifts.

TABLE 6: Resources required by the basis operations.

Operation	Logic utilization	Register utilization	Number of logic cell
Delay	0	R	R
Addition/subtraction	R	0	R
Integrator	R	R	
Multiplication by a constant	0 or R	0	0 or R
Squaring	$R^2 + 4R$	0	$R^2 + 4R$

TABLE 7: Resources required by the algorithms.

Operation	Number of logic cell
Moving averager	$R(N + 2) + \log_2(N) + 1$
Exponential averager	$2R + 1$
Moving variance network	$2R^2 + R(3N + 16) + 2\log_2(N) + 1$
Exponential moving variance network	$R^2 + 13R + 12$

FIGURE 17: NMV and EMV output during dynamic conditions.

It can be seen that the EMV implementation requires less resources than the NMV. This is due to two points: (1) it does not need to store N samples; (2) it requires only one squaring against two for the NMV implementation. For $N = 64$, Table 8 shows a comparison for implementation of both methods in terms of logical elements, where a block includes all the operations needed to process measurements from one axis of the accelerometer.

7.3. Power Consumption Considerations. Package tracking applications are not necessarily very demanding in terms of positioning availability. In fact, it is typically sufficient to obtain a position fix every few hours in order to track a

FIGURE 18: Comparison of time responses of NMV and EMV with different weights.

package from its starting point to its destination. Therefore, the proposed algorithm does not need to work all the time. Instead, a timer can periodically wake up the tracking module algorithm for limited time duration. If during this time a final static condition is detected, then the module can make a transmission to enable its localization. Otherwise, the module goes back to sleep until the next timer occurs. By doing so, the power consumption of the algorithm can be reduced drastically. In addition, even when the system is awake, it has been shown that the algorithm only requires few logical elements and that the sampling frequency needed is very low (<70 Hz), which will result in a power consumption of less than 4 mW considering, for example, an Altera MAX V CPLD [19]. Also, taking into consideration the fact that MEMS-based accelerometers are not power hungry (e.g., <1 mW for the LIS302DL accelerometer [16]), we can infer that the total power consumption of the detection module will be less than 5 mW while active.

Regarding the power consumption of the transmission module, it will consist of the power to establish the connection, and the power to maintain the transmission. As it was already shown in [20], module communication radios have low connection maintenance energy, but high energy per bit transmission cost and low bandwidth. Therefore, the

TABLE 8: Comparison for implementation resources of different methods for $N = 64$.

R	NMV		EMV	
	Total of LEs without memory	Total of LEs with memory	Total of bits with memory	Total of LEs
	1 block	3 blocks	1 block	3 blocks
8	1807	5421	271	813
10	2295	6885	375	1125
12	2799	8397	495	1485
14	3319	9957	631	1893
16	3855	11565	783	2349
18	4407	13221	951	2853

TABLE 9: Main parameter characterization of the proposed algorithm using NMV or EMV methods.

Algorithm	NMV	EMV
T_σ	$2.5\sigma^2 - 4\sigma^2$	$2.5\sigma^2 - 4\sigma^2$
T_{th}	$4s \leq T_{th} \leq 5s$	$4s \leq T_{th} \leq 5s$
α	$\backslash$	$1/N < \alpha \leq 5/N^*$
W_s	$1s \leq W_s \leq 2s^{**}$	$1s \leq W_s \leq 2s^{**}$
P_M	$\geq 6 \times 10^{-19}$	$\geq 6 \times 10^{-19}$
P_{FA}^t	$\leq 4 \times 10^{-3}$	$\leq 4 \times 10^{-3}$
P_{FA}	$\leq 3 \times 10^{-10}$	$\leq 3 \times 10^{-10}$
Sampling frequency	60 Hz–70 Hz	60 Hz–70 Hz
Implementation resources	$813 \leq \text{LEs} \leq 3405$	$540 \leq \text{LEs} \leq 2016$

*N is the number of samples per window size, and the result of dividing the weighting factor by N should be a multiple of 2 for optimal implementation.

**The number of samples in a window size should be a multiple of 2.

FIGURE 19: Implementation of an N -point moving averager [18].

transmission power will be the dominant factor. Since no data is sent during the sleep mode of the tracking module and the size of the data packets sent during the active mode will only consist of the protocol data units (PDUs), that are small in size (16 bytes), the required power consumption should be typically less than 10 mW, according to [20]. Therefore, if we assume the usage of a standard Li-Ion battery such as the one used in an iPhone holding an energy of 5.3 Wh, then we can conclude that the module can support thousands of transmissions, which roughly means many days of battery lifetime.

8. Summary

Based on the analysis shown in this paper, the parameters characterizations of the proposed algorithm using either NMV or EMV methods are summarized in Table 9.

9. Conclusions

In this paper, we proposed a movement recognition algorithm using a low quality MEMS accelerometer sensor

FIGURE 20: Implementation of a recursive N -point moving variance network [18].

FIGURE 21: Implementation of an exponential averager [18].

integrated in a module that is meant to be used onboard an airplane for flight mode detection. The algorithm successfully detects the state of the sensor and prevents the module to transmit mobile communication signals during the flight. The proposed algorithm is based on computing the normal moving variance (NMV) or exponential moving variance (EMV) during a specific window of time of the acceleration measurements and comparing the result to a threshold. In addition, consecutive results from different time windows are

FIGURE 22: Implementation of an exponential averaging network [18].

further compared to a threshold in time to declare a final static condition. The main characteristics of the algorithm are summarized as follows. Generality, as it is independent of the accelerometer quality or type, where only the noise behavior of the sensor is needed to define all the algorithm parameters. Reliability, as it provides a very low probability of false alarm to meet the requirements needed by the airspace regulations. Flexibility, where it is sufficient to vary the MV threshold and the threshold in time to adapt it for different application. Simplicity, as it requires low resources, and thus, it can be implemented on a small FPGA or a CPLD. Consequently, the detection module including the accelerometer and the hardware where the algorithm is implemented will have very low power consumption, typically less than 5 mW.

References

- [1] S. Wang, J. Min, and B. K. Yi, "Location based services for mobiles: technologies and standards," in *Proceedings of IEEE International Conference on Communication (ICC '08)*, Beijing, China, May 2008.
- [2] "Jiiva Tracing Solutions," August 2012, <http://www.jiiva.com/>.
- [3] "Wireless Devices on Airplane," Federal Communications Commission FCC, May 2011, <http://www.fcc.gov/guides/wireless-devices-airplanes>.
- [4] C. S. Miguelez, "GSM Operation Onboard Aircraft," European Telecommunications Standards Institute (ETSI), Cedex, France, January 2007.
- [5] S. L. Lau and K. David, "Movement recognition using the accelerometer in smartphones," in *Future Network and Mobile Summit*, pp. 1–9, June 2010.
- [6] D. Fuentes, L. Gonzalez-Abril, C. Angulo, and J. A. Ortega, "Online motion recognition using an accelerometer in a mobile device," *Expert Systems with Applications*, vol. 39, no. 3, pp. 2461–2465, 2012.
- [7] T. Brezmes, J. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone," in *Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, vol. 5518 of *Lecture Notes in Computer Science*, pp. 796–799, Springer, Salamanca, Spain, June 2009.
- [8] T. Zhang, J. Wang, P. Liu, and J. Hou, "Fall detection by embedding an accelerometer in cellphone and using KFD algorithm," *International Journal of Computer Science and Network Security*, vol. 6, no. 10, pp. 277–284, 2006.
- [9] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 1, pp. 156–167, 2006.
- [10] T. R. Burchfield and S. Venkatesan, "Accelerometer-based human abnormal movement detection in wireless sensor networks," in *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (HealthNet '07)*, pp. 67–69, San Juan, Puerto Rico, June 2007.
- [11] J. Shin, D. Shin, D. Shin, S. Her, S. Kim, and M. Lee, "Human movement detection algorithm using 3-axis accelerometer sensor based on low-power management scheme for mobile health care system," in *Advances in Grid and Pervasive Computing*, vol. 6104 of *Lecture Notes in Computer Science*, pp. 81–90, Springer, 2010.
- [12] Y. Li, X. Chen, X. Zhang, K. Wang, and Z. J. Wang, "A sign-component-based framework for Chinese sign language recognition using accelerometer and sEMG data," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 10, pp. 2695–2704, 2012.
- [13] J. S. Wang, Y. L. Hsu, and J. N. Liu, "An inertial-measurement-unit-based pen with a trajectory reconstruction algorithm and its applications," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 10, pp. 3508–3521, 2010.
- [14] Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou, "Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information," in *Proceedings of the 6th International Workshop on Wearable and Implantable Body Sensor Networks (BSN '09)*, pp. 138–143, June 2009.
- [15] D. T. W. Fong, J. C. Y. Wong, A. H. F. Lam, R. H. W. Lam, and W. J. Li, "A wireless motion sensing system using ADXL MEMS accelerometers for sports science applications," in *Proceedings of the 5th World Congress on Intelligent Control and Automation, Conference Proceedings (WCICA '04)*, pp. 5635–5640, June 2004.
- [16] STMicroelectronics, "Analog, Sensors and MEMS, Accelerometers, LIS302DL," August 2012, <http://www.st.com/internet/analog/product/152913.jsp>.
- [17] R. G. Lyons, *Understanding Digital Signal Processing*, Prentice Hall, New York, NY, USA, 3rd edition, 2010.
- [18] L. L. Scharf, *Statistical Signal Processing Detection, Estimation, and Time Series Analysis*, Addison Wesley, Reading, Mass, USA, 1991.
- [19] Altera, "PowerPlay Early Power Estimators (EPE) and Power Analyzer," August 2012, <http://www.altera.com/support/devices/estimator/pow-powerplay.jsp>.
- [20] A. Rahmati and L. Zhong, "Context-for-wireless: context-sensitive energy-efficient wireless data transfer," in *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (MobiSys '07)*, pp. 165–178, San Juan, Puerto Rico, June 2007.