Wireless Sensing Applications for Critical Industrial Environments

Fabien Joseph Chraim

Technical Report No. UCB/EECS-2014-215
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-215.html

December 13, 2014

Wireless Sensing Applications for Critical Industrial Environments

by

Fabien Joseph Chraim

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy
in

Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Fall 2014

Abstract

Wireless Sensing Applications for Critical Industrial Environments

by

Fabien Joseph Chraim

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Kristofer S.J. Pister, Chair

The widespread deployment of the World Wide Web over the past few decades has connected people globally. The next logical step in this evolution was to connect people to their environment. As public interest in the aging infrastructure grew, so did the desire to make this infrastructure safer and more environmentally friendly. This required the development of low-power wireless sensor networks to monitor and control this infrastructure, the so-called Internet of Things. With the introduction of low-power communication standards rooted in the Time Slotted Channel Hopping (TSCH) mechanism, the ability to have sensors communicate robustly (99.999%) over long periods of time (10 years) using battery power, appears to have been largely achieved.

This thesis is concerned with exploring three significant applications of these new communication networks. It also introduces two modeling tools useful in designing new applications for this space. A process automation solution centered around rotary valve position monitoring is presented first. The "peel-and-stick" MEMS device shows an accuracy of $\pm5^{\circ}$ for quarter-turn valves, and an accuracy of $\pm10\%$ of a turn was obtained with the multi-turn valves. This cost-effective solution is designed to be densely deployed on most of the plant's rotary valves. Next, the design and implementation of fence line perimeter security application is developed and verified. Combining a MEMS accelerometer with a hypothesis testing algorithm, all 91 known intrusions were detected with no false alarms over a period of approximately two months. Finally, a wireless gas leak detection and localization is evaluated from a sensory swarm perspective. This application highlights the detection challenges when working with noisy sensors. A gas plume detection rate higher than 90% is demonstrated, with 7 false alarms over a period of three days and an average detection delay of 108s. In terms of localization, the system estimated the leak locations to within three meters of the actual leak source.

The design of the new system tools is controlled by the energy trade-off between transmitting and locally processing captured data. This has direct implications on the Quality of Service (robustness, delay, lifetime, etc.). An adaptable energy consumption model for TSCH networks is presented and then generalized to an application energy consumption model. These design-time tools allow developers to understand the feasibility of the new application and to refine its hardware and software design in order to meet predefined specifications. The application energy model is presented by taking a motor vibration application as an example, while combining the sensing and communication hardware with energy scavenging.

The work done in this thesis is experimental in nature and care is taken to bring the proposed ideas as close to real implementations as possible: the hardware is built using commercial off-the-shelf components, the networks are configured for the application at hand, and algorithms are devised and validated in each setting.

To Elias Nabhan, who never made it to engineering school...

Contents

List of Figures

List of Tables

Acknowledgements

For being what many look for in an advisor, I would like to thank Kris Pister. Hands-off enough to allow me to wander in research, but involved enough to infect me with his ambitious visions, Kris made my path through graduate school much more interesting.

I consider Alex Bayen and Steve Glaser as my two other mentors during the last few years. I am grateful for all their useful advice, and inspired by their academic accomplishments.

The Chevron Energy Technology Company and Chevron Information Technology Company funded and supported most of my research, and as such, an acknowledgment is due. Greg LaFramboise and Roy Kutlic have visionary ideas in the oil and gas industry, and I would like to extend a big thank you to them. Paula Lucchini, Beverly Coleman and Julio Cedeno also deserve recognition for driving crucial change in today's corporate world.

The Berkeley Sensor and Actuator Center is, in my humble opinion, a model for research centers across the world. The breadth of its contributions is as impressive as the fact that it is held together by a small team of people. For their invaluable support, professionalism and kindness, I would like to thank John Huggins, Kim Ly, Richard Lossing, Dalene Schwartz and Alain Kesseru.

In many ways, UC Berkeley is a great institution. I believe that it is only as great as its students. During the past five years, I have had the honor and privilege of working alongside some of them. I hope only that I was able to affect them as much as they affected me. For their friendship and support, I would like to thank Branko Kerkez, Mike Lorek and Travis Massey. During my time in the Pister group, I had the privilege of working alongside Thomas Watteyne, Anity Flynn, Ankur Mehta, Andrew Tinka and Kevin Weekly (in the early years), then Xavi Vilajosana, Qin Wang, Borja Martinez and Tengfei Chang (in the later years). Whether they know it or not, they have all helped in shaping me as a researcher.

Even though they were with me in the same program, I rarely discussed engineering or my research with Yusuf Bugra Erol and Daniel Aranki. For this, for their friendship, and for all the late night tea sessions at the I-House Cafe patio, I would like to thank them both.

Sravan Puttagunta had to put some of his ambitions on hold in waiting for me to complete my program. For his patience and commitment, I thank him.

I have yet to see most of this world, but for the time being, I would like to acknowledge a few places which left a large mark on my person: Berkeley, the state of California, and the villages of Faraya and Fakiha in the beautiful country of Lebanon.

My parents' sacrifices, love and complete trust got me to where I am today. I owe them more than I will ever be able to put down in words. I am also privileged and grateful to have two brothers who never stopped encouraging me. Even though I never intended to be a surgeon, Michel and Andre have been with me since the start.

Finally, for my lovely wife Rebecca, an acknowledgment and an apology. You have been with me since the beginning of my journey West. Your support made all the difference, and because of you my experience in my graduate studies was marvelous. Thank you for introducing me to Harry Potter, and I'm sorry for all the time I spent away from you.

Introduction

Our daily lives are increasingly dependent on industries like transportation, food, construction, oil and gas, utilities, communication, and medicine. Therefore, it is essential to make these industries resilient to accidents and sabotage in order to continue improvements to our quality of life. As the environmental impact of our industries becomes more apparent, reducing the emissions and toxic wastes from these plants turns into a priority. Furthermore, it is imperative to keep these industries secure and accident-free, by virtue of the fact that they exist in close proximity to our large cities. Finally, human workers are still at the heart of many of these plants. Consequently, plant managers and operators need to ensure the safety of these workers. Wireless sensing therefore becomes an essential tool in keeping critical industrial environments safe, efficient, and competitive.

The turn of the 21st century witnessed great advances in the field of wireless sensing. With this increased interest in wireless technologies however, came a battle over standardization. Hundreds of interested parties from private firms to government institutions, joined the discussion surrounding the communication protocols that would govern the operation of these resource-constrained, battery-operated wireless sensors. Simplistic communication protocols (CSMA[1]-based) were hastily adopted with the goal of quickly capitalizing on this "new-found" world of wireless. This resulted in a number of failed deployments that led to a mistrust of low-power wireless technology in general, especially within industrial fields. As such, the trillions of sensors envisioned by some, and the billions of dollars expected by many, are yet to happen in this world of wireless sensors.

[1]Carrier Sense Multiple Access

Still, a reasonable amount of notable ideas did find their way into products and standards. These are based mainly on the concepts on time-synchronization and channel diversity (WirelessHART [1], ISA100.11a [2], IEEE802.15.4e [3], SmartMesh IP [4]). The introduction of these new communication methods, seems to have resolved the problems of reliability and energy consumption. Indeed, products on the market today achieve a 99.999% reliability with a 10-year battery lifetime. This has led to the beginning of long-term industrial wireless deployments in the oil and gas industry. However, many chemical plants and refineries worldwide are nearing their centennial anniversary and are in dire need of retrofitting and monitoring. Wireless sensing which decreases the installation cost by orders of magnitude, presents an opportunity to dramatically increase the number of installed sensors. Moreover, removing the wires from harsh environments allows plant operators to extract valuable information from locations which were previously unattainable. However, many of the applications being implemented today are rather simplistic: temperature and pressure sensing, flow measurement, contact switches etc. What does it take to go beyond the simple monitoring applications available today? What are the considerations at the hardware and firmware level that need addressing when developing new sensing methods? What is the best way to apply the mathematical tools to the data generated by the sensors? How do different network, sensing and application-level configurations affect the device lifetime and communication reliability?

This thesis is concerned with exploring the realm of possible applications on the newly-developed industrial wireless infrastructure. The work presented herein is divided into two parts. The first addresses three applications that were developed and verified. The second, describes the tools that were built along the way to simplify the design and evaluation process. The applications in Part I revolve around making plants safer and more secure. As such, a "peel-and-stick" wireless valve position monitoring sensor is studied first, with an emphasis on sensing and computation challenges on the resource-limited devices. The second application is perimeter security solution called SmartFence. In essence, the possibility of adding a first line of defense around the plant is studied, utilizing a wireless low-power network. Analysis of this application also presents a widely applicable detection algorithm based in sequential probability testing. The third application presented is wireless gas leak detection and localization. The aim is to demonstrate a reliable and cost-effective solution for the dangerous leaks that plague many plants worldwide. Focus is given to the swarm aspect of the solution by looking at how information gathered from many sensors can be used to solve crucial problems in real time. The detection and localization algorithm is presented along with the results of a large-scale experiment with many participants.

Energy is known to be a major challenge in the field of wireless industrial sensing. Part II of this thesis revolves around two models for addressing those challenges. An energy modeling tool applicable to Time-Synchronized Channel-Hopping (TSCH) networks is presented first. This model is developed using commercial off-the-shelf hardware and allows the user to evaluate the effect of network configuration on energy. Building upon this model, a generalized energy consumption model for wireless sensing applications is then derived. This second tool takes into account the sensing and computation components taking place on-board the device. With energy scavenging and storage in mind, this model is validated by means of the sample application of machine vibration monitoring.

The method of choice here is an experimental one. In wireless distributed sensing, the most useful research style is often one with a heavy emphasis on experiments and real data. Very stringent but realistic requirements are imposed on the applications presented here: lifetimes of 10 years on regular lithium batteries, quick detections, low false alarms, an 8 or 16 bit microcontroller with complex operations to run, etc. By bringing the research as close to commercial deployments as possible, the hope is to go beyond simple proofs of concept, since many challenges are embedded in the implementation process.

Part I:

Applications

Chapter 1

Valve Position Monitoring: a MEMS Approach[1]

From controlling the flow of water, to controlling machines, valves have been with us for centuries. Monitoring the position of valves is clearly beneficial for plant operators. The capability of detecting incorrect valve positions can help avoid the types of accidents that most often lead to personnel injuries and damaged equipment. Attempts to prevent such accidents commonly involve sending employees to various parts of the plant in order to verify valve positions. However, the use of wireless communication to accomplish this task is more time and cost effective. Additionally, access to plant-wide valve position reports will aid operators in making sense of each process pressure gradients [5–9].

In some market reports, experts claim that up to 85% of valves are left without any monitoring [7]. Such a low penetration rate for valve position sensing may be attributed to the fact that sensing technology has been largely stagnant for the past two decades. Most solutions employ some sort of magnetic or optical encoding while others feature contact switches [7–9]. This makes most products expensive and bulky. As wireless technologies

[1]**This chapter was published in a similar form as follows:**
Chraim, F.; Pister, K., "Wireless Valve Position Monitoring: A MEMS Approach," 39th Annual Conference of the IEEE Industrial Electronics Society, 2013

become more accepted in various industries, the number of vendors offering wireless communication is increasing. However, many of the products on the market today do not take full advantage of low-power wireless technologies, rendering them power-inefficient. Additionally, the installation of these solutions is often problematic: in many cases, a complete halt of the process is required to retrofit an existing valve with new sensors. Individual sensor calibration in some cases is also a significant barrier to speedy installations. The industry as a whole seems to be focused on satisfying ISO standards and making their products explosion-proof, while paying little attention to the sensing technology. The advantages of MEMS sensors with their easy packaging and their intrinsic safety appear to have completely missed this market.

In this chapter, we address the problem of valve position monitoring through the use of MEMS sensors. A cheap "peel and stick" wireless solution is presented that requires no calibration and consumes very little energy. Our main contribution is in addressing the issues described above: cost, calibration, energy consumption, size and ease of installation. Along with proposing the hardware, we present a sensing algorithm described in section 1.3 before concluding this chapter with our results in section 1.4.

1.1 Experimental Setup

This study is focused on rotary valves. In particular, we look at multi-turn valves and quarter-turn ones (though the same work can be extended to any part-turn model). Seen in figure 1.1 is our experimental setup with two valve models: a gate valve and a ball valve. The goal is to develop a way to keep track of the number of turns on a multi-turn valve by using the gate model, while the ball valve will be used to develop the quarter-turn angle tracking. Also seen in the figure is GINA (guidance and inertial navigation assistant) [10], the sensing platform of choice. This board holds the MSP430f2618 from Texas Instruments, along with a 9-axis Inertial Measurement Unit (see table 1.1). Other than the sensors, the microcontroller interfaces with the WirelessHART compliant DN2510 radio by Dust

Networks [1]. This general-purpose wireless sensing platform is utilized throughout the research presented in this thesis (in chapters 2, 4 and 5).

With the setup in place, we ran several experiments where data from all the sensors was collected and transmitted to a basestation. The first step in this study was to select the sampling frequency of the sensors. It is desirable in this case to lower this frequency as much as possible since each sample requires on-board communication with the sensor, some sort of analog-to-digital conversion, and processing at the microcontroller. This has a direct effect on the overall energy consumption of the device, and therefore its lifetime. This topic is explored in more depth in Chapter 5. Initially, the data was collected at 300 Hz. However, after looking at the frequency content of this data, we noted that a sampling frequency of 17 Hz was enough to capture all the motion information. All subsequent experiments were therefore run at that frequency. In each of the experiments, the valves were turned from one extreme to the other repeatedly, and in different valve orientations (horizontal, vertical and oblique).

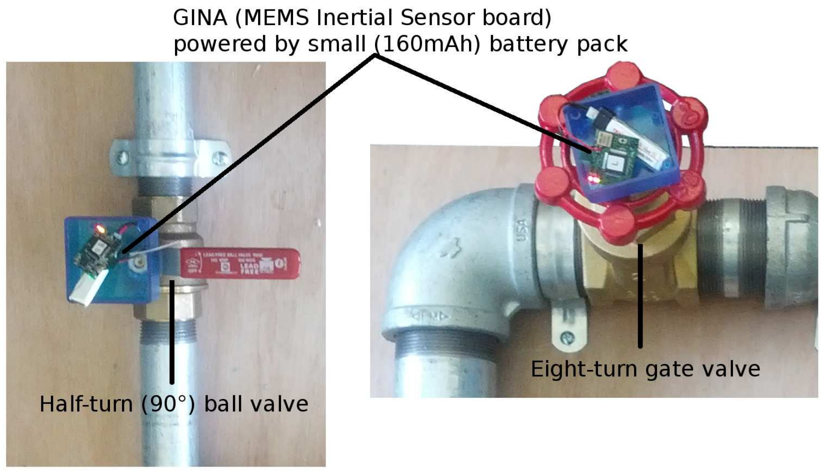

GINA (MEMS Inertial Sensor board)
powered by small (160mAh) battery pack
Half-turn (90°) ball valve
Eight-turn gate valve

Figure 1.1: The experimental setup showing an instrumented ball valve (left) and an instrumented gate valve (right)

Type	**Manufacturer**	**Part Number**	**Features**
Microcontroller	Texas Instruments	MSP430F2618	16-bit, 16MHz, 116kB flash, 8kB RAM
3-axis accelerometer (sensitive)	STMicroelectronics	LIS344ALHTR	+/-2 Gs or +/-6 Gs, 1.8 kHz, 660 mV/G
3-axis accelerometer (large range)	Kionix	KXSD9-1026	+/-8 Gs, 2 kHz
3-axis gyroscope	Invensense	ITG3200	2000 degs/s
3-axis magnetometer	Honeywell	HMC5843	+/- 6 Oe, 116Hz
Temperature sensor	Texas Instruments	TMP20AIDRLT	+/-2.5 C, -55 C to 130 C

Table 1.1: GINA sensing platform hardware description

1.2 Detection with Noisy Sensors

Tracking the valve angle in its plane of rotation is not as straightforward as one may think. Indeed, MEMS sensors are generally noisy or prone to drifting. While it is tempting to task this problem to the processing power of a PC, this would require the device to transmit all of its sensor data back to a server. Given our hardware design, an activity of five seconds at the valve (opening a gate valve for example) would generate about 12 data packets for transmission. This is not only costly in terms energy for the device itself, but it is also costly for the nodes transporting the data to the gateway[2]. Therefore, the detection needs to take place on board the 16-bit microcontroller used here.

The initial plan was to use the magnetometer to compute the orientation of the sensor. However, collecting preliminary data in the vicinity of the pipes revealed that the magnetic field around them is highly non-uniform. Additionally, considering the environment where the devices will ultimately be placed (factories and plants), the presence of metallic machines and large moving vehicles will make the magnetic environment even more unpredictable. Figure 1.2 shows the results of an experiment run on the ball valve. In this time series, the valve angle is computed from the magnetometer data and plotted over time. Although the first 120 seconds were supposed to show an idle device, the magnetometer picked up significant variations. During the motion of the ball valve between the two extremes (a

[2]A more in-depth analysis of the network energy and associated tradeoffs is presented in Chapter 4

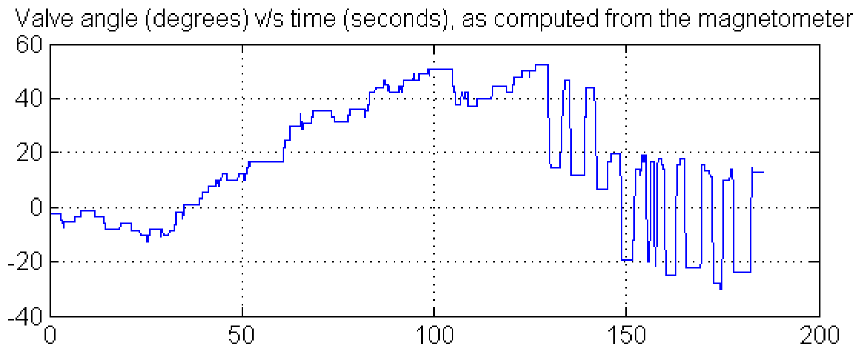


Figure 1.2: Plot of the valve angle over time, as inferred from the magnetometer. The valve was steady for the first 120 seconds (no motion), then it was turned from one end to the other repeatedly. Clearly the magnetometer recorded significant motion when there was none, then showed a repetitive swing of around 40° instead of 90°

range of 90°), the magnetometer picked up a reduced range (about 40°) while constantly drifting.

The magnetometer was ultimately ruled out from the list of possible sensors. Instead, we will present how the gyroscope and accelerometer were used to figure out the valve angle. There are two types of drift in MEMS gyroscopes. One of them is due to the temperature of the sensor, while the other is a result of manufacturing constraints. MEMS accelerometers suffer from noise issues. Next, we will see how each of these issues was tackled.

1.2.1 Gyroscope Temperature Self-Calibration

Consumer-grade MEMS gyroscopes commercially available today are inherently sensitive to temperature due to the relatively large temperature coefficient of single crystal silicon. Process limitations then rule out one-time factory calibration [11]. Luckily, this relationship is linear (in our device) over a wide range, meaning that one can easily calibrate each axis with respect to the on-board temperature reading of the chip. In our study, we imposed the constraint that each sensor cannot be calibrated independently before installation. Instead a method needed to be devised to allow the sensor to self-calibrate. When the sensor is first turned on, the operating of current heats the device. Keeping the sensing

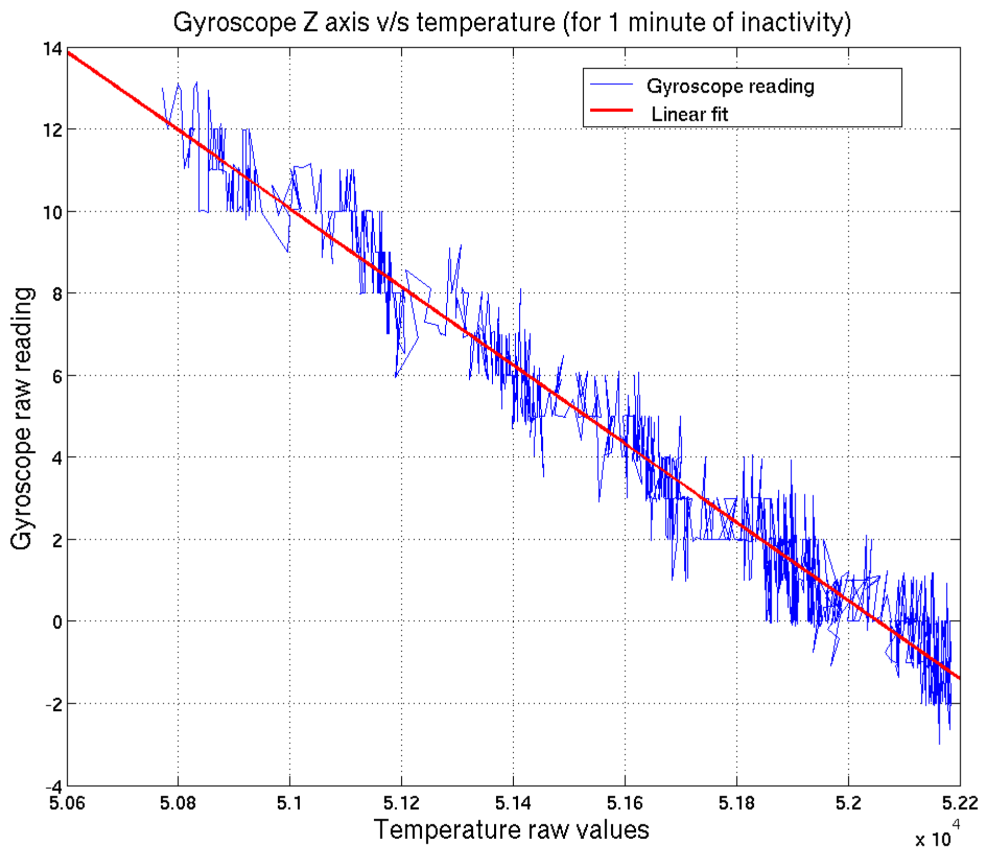


Figure 1.3: The plot of the gyroscope axis reading v/s temperature shows the linear dependency between the two. This data was recorded for one minute of inactivity.

platform stationary in this initial step allows for the self-calibration of the gyroscope. We took advantage of the initial heating of the stationary device to establish the compensation function. Seen in figure 1.3 is the plot of the Z axis reading versus temperature for the first minute of operation, where the sensor is sampled every 3 ms. From this data, one can easily derive the slope and intercept of this line, through an Ordinary Least Square approach.

Given a vector of n temperature readings T_i, padded with a column of ones and a vector of sensor data $\hat{gx}_i$ (say the X axis of the gyroscope when it is stationary),

$$A_{n,2} = \begin{pmatrix} T_1 & 1 \\ T_2 & 1 \\ \vdots & \vdots \\ T_n & 1 \end{pmatrix}, B_{2,1} = \begin{pmatrix} slope \\ intercept \end{pmatrix}, \hat{Gx}_{n,1} = \begin{pmatrix} \hat{gx}_1 \\ \hat{gx}_2 \\ \vdots \\ \hat{gx}_n \end{pmatrix} \tag{1.1}$$

here is the method used in order to find the slope and intercept of the linear relationship.

$$AB = \hat{Gx} \tag{1.2}$$

$$A^T AB = A^T \hat{Gx} \tag{1.3}$$

$$B = (A^T A)^{-1} A^T \hat{Gx} \tag{1.4}$$

The result is then used in the following fashion, where Gx is the temperature compensated gyroscope vector. The same procedure is repeated for all three axes.

$$Gx = \hat{Gx} - AB \tag{1.5}$$

1.2.2 Issues with Rigid-Body Dynamics

A calibrated gyroscope yields the angular rate of a rigid body (valve) along three axes, in the body (moving) reference frame. In order to integrate those rates and find the valve angle with respect to a fixed reference frame, one must rotate the rates to the Earth reference frame. One would be tempted to retrieve the Euler angle rates (roll, pitch and yaw) directly using relations similar to the following.

$$\dot{\phi} = Gx + Gy \cdot sin(\phi)tan(\theta) + Gz \cdot cos(\phi)tan(\theta) \tag{1.6}$$

$$\dot{\theta} = Gy \cdot cos(\phi) + Gz \cdot sin(\phi) \tag{1.7}$$

$$\dot{\psi} = Gy \cdot sin(\phi)/cos(\theta) + Gz \cdot cos(\phi)/cos(\theta) \tag{1.8}$$

This mapping is however not unique and could potentially result in a loss of information (known as the *gimbal lock problem* [12]). A possible workaround involves the introduction of quaternion algebra, which is an extension of the algebra of complex numbers. Quaternions are 4-tuples that can be thought of as a sum of a scalar and a vector, and do not suffer from the same issues as Euler angles. Seen here are the equations deriving the quaternion rates directly from the gyroscope data.

$$\dot{q_0} = -q_1 \cdot Gx/2 - q_2 \cdot Gy/2 - q_3 \cdot Gz/2 \tag{1.9}$$

$$\dot{q_1} = q_0 \cdot Gx/2 + q_2 \cdot Gz/2 - q_3 \cdot Gy/2 \tag{1.10}$$

$$\dot{q_2} = q_0 \cdot Gy/2 - q_1 \cdot Gz/2 + q_3 \cdot Gx/2 \tag{1.11}$$

$$\dot{q_3} = q_0 \cdot Gz/2 + q_1 \cdot Gy/2 - q_2 \cdot Gx/2 \tag{1.12}$$

These quaternion rates can be integrated directly and one can then obtain the Euler angles through the following equations.

$$\theta = arctan(2\frac{q_0 \cdot q_1 + q_2 \cdot q_3}{1 - 2 \cdot ({q_1}^2 + {q_2}^2)}) \tag{1.13}$$

$$\phi = arcsin(2 \cdot (q_0 \cdot q_2 - q_1 \cdot q_3)) \tag{1.14}$$

$$\psi = arctan(2\frac{q_0 \cdot q_3 + q_1 \cdot q_2}{1 - 2 \cdot ({q_2}^2 + {q_3}^2)}) \tag{1.15}$$

1.2.3 Gyroscope Motion Drift

MEMS gyroscopes drift, mainly due to manufacturing constraints (this drift is not related to temperature). This can be seen when integrating the angular rates obtained from a sensor rotating back and forth between two positions. The resulting angle is not equal to zero as one would expect. Instead, a DC bias appears and varies over time. In our experiment in particular, when looking at the ball valve, one can see that moving the valve away from its original position and back results in an offset from the original angle (refer to figure 1.4). Since there was no way of removing the drift from the gyroscope itself, we looked at the accelerometer data. The initial distribution of the gravity vector along the three axes of the accelerometer was captured and saved (when the sensor was not in motion). Any time this same distribution was measured and the valve angle was close to zero, the angle measurement was reset. In an ideal case (with no gyroscope drift), the initial accelerometer distribution would occur exactly when the valve angle was measured to be zero. However,

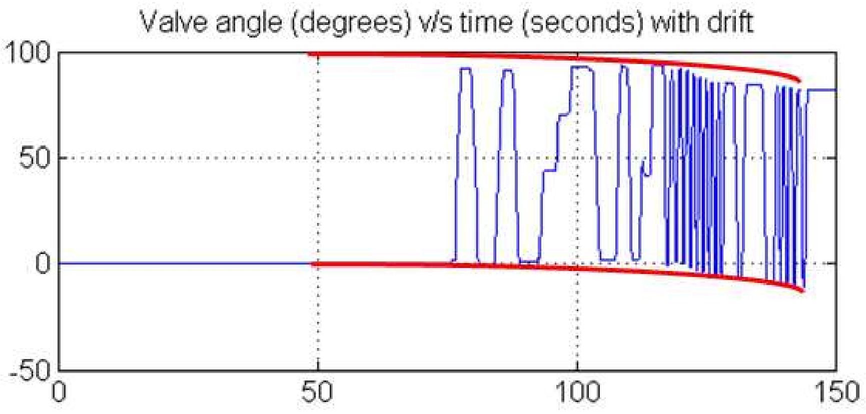


Figure 1.4: Integrating the gyroscope data yields a significant drift. This drift occurs only during motion, as the valve is moved from one end to the other (0° to 90°)

because of this drift, resetting the angle manually becomes necessary. Additionally, the sensing platform was mounted in such a way to have a significant proportion of the gravity vector on each of the axes. This allowed to improve variance of the signal with respect to the noise, on each accelerometer axis. Of course, we needed to detect when the valve was stationary to perform this angle correction as, when the valve moves, the accelerometer also picks up this motion. We will look at the results of this operation in section 1.4.

1.2.4 Filtering Accelerometer Noise

In general, MEMS accelerometers suffer from noise, even when stationary. This can be seen in figure 1.5 which tracks the motion on each of three accelerometer axes as the ball valve moves. In this study, since we only require the accelerometer to capture the stationary gravity vector and not the motion of the valve, only smoothing was necessary. As such, a 1st degree Robust Locally Weighted Regression filter was used [13], with a window size of 20 samples. This LOWESS filter is better than a simple moving average filter, when it comes to filtering outliers (erratic accelerometer noise peaks). This is due to the fact that each sample has an associated weight at any iteration, and due to the regression fit attempted by the filter. Though computationally more intensive compared to the rolling mean approach, the LOWESS filter was necessary in getting rid of those outliers, and yielded overall improved smoothing with a quicker settling time. Finally, seen in figure 1.6 is the time series representation of the same three axes, but at the output of the filter. We note that the first few seconds of accelerometer inactivity are characterized by a settling of the internal Infinite Impulse Response (IIR) filter on each axis, which explains the slight increase or decrease of sensor reading during that time. Clearly, looking at the difference between figures 1.5 and 1.6, we can see that the signal to noise ratio has decreased significantly, allowing us to use the accelerometer reading to extrapolate as to whether the valve has returned to its original position or not, as it moves between the two extremes.

1.3 Algorithm

The detailed pesudocode for the detection algorithm used to compute and report the valve angle is shown in Algorithm 1.

The function *initialize()* contains the operations described in section 1.2.1. This function is responsible for the temperature self-calibration, among other things (such as obtaining the reference accelerometer reading necessary for the drift correction). The *rotate()* function moves the gyroscope data from the rigid body reference frame to the Earth reference frame. The *filter()* function takes acceleromter data and returns a filtered version, as described in the part concerning the LOWESS filter. Finally, the function called *wireless.send()* transmits the data over the WirelessHART network.

The sensing routine shown here is designed with energy consumption in mind. When the valve is not moving (in the *IDLE* state), only the gyroscope is sampled and a very basic compare function used to determine whether the device is in motion or not. This is commonly known as Lebesgue sampling. The set of instructions that occur during the motion of the valve are simple additions and multiplications, which do not constitute a heavy microcontroller load. The heaviest function (computationally) is the *filter()* one since it runs two iterations of additions and multiplications. We note however that this function is only called after the valve returns to the idle position and only on 20 to 40 accelerometer samples. This allows the entire routine to run on a basic microcontroller without presenting a heavy

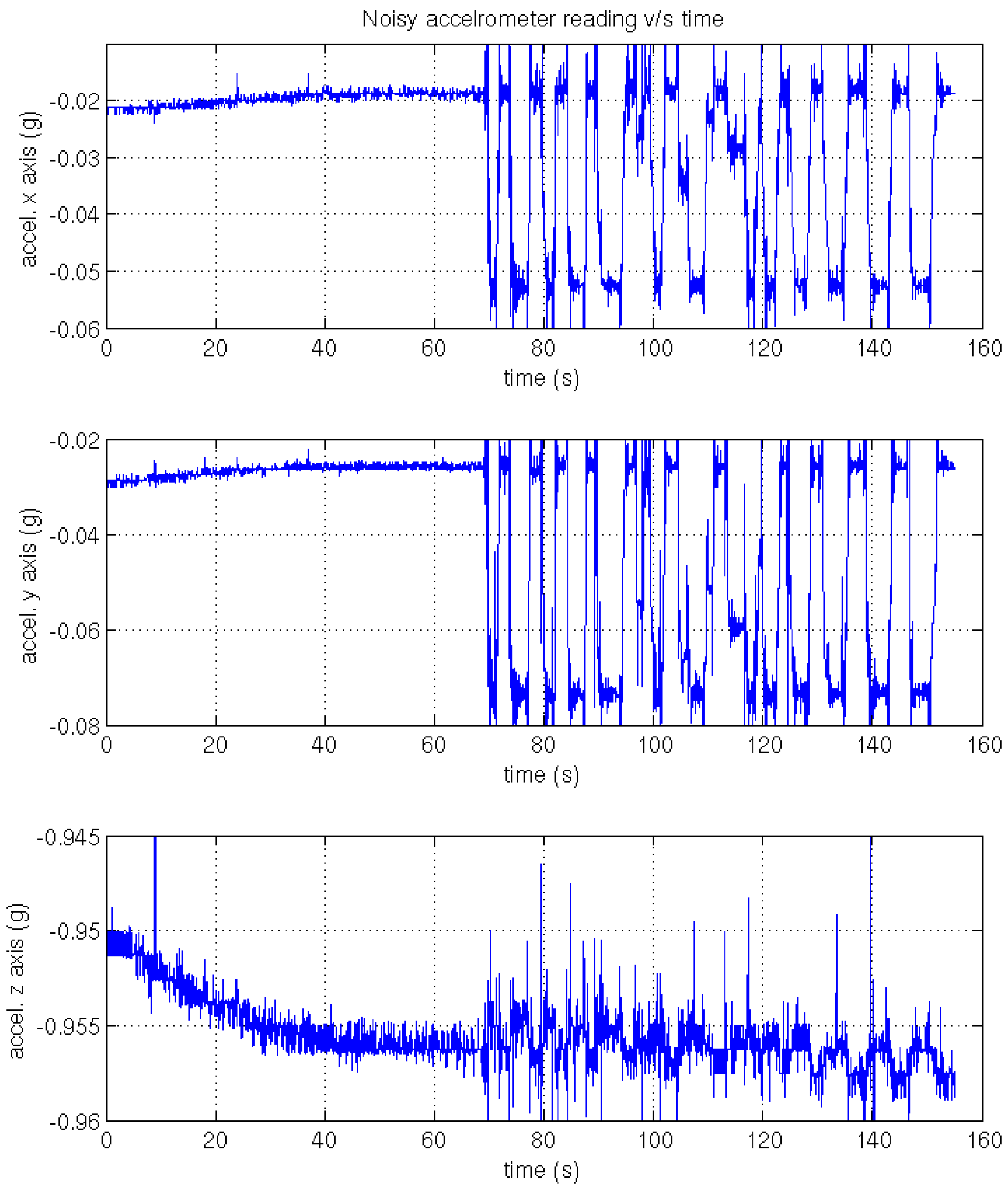


Figure 1.5: Unfiltered Accelerometer readings (in units of g's) on all three axes during the motion of the ball valve. Motion starts 70 seconds in, and the valve is moved between 0° and 90° repeatedly

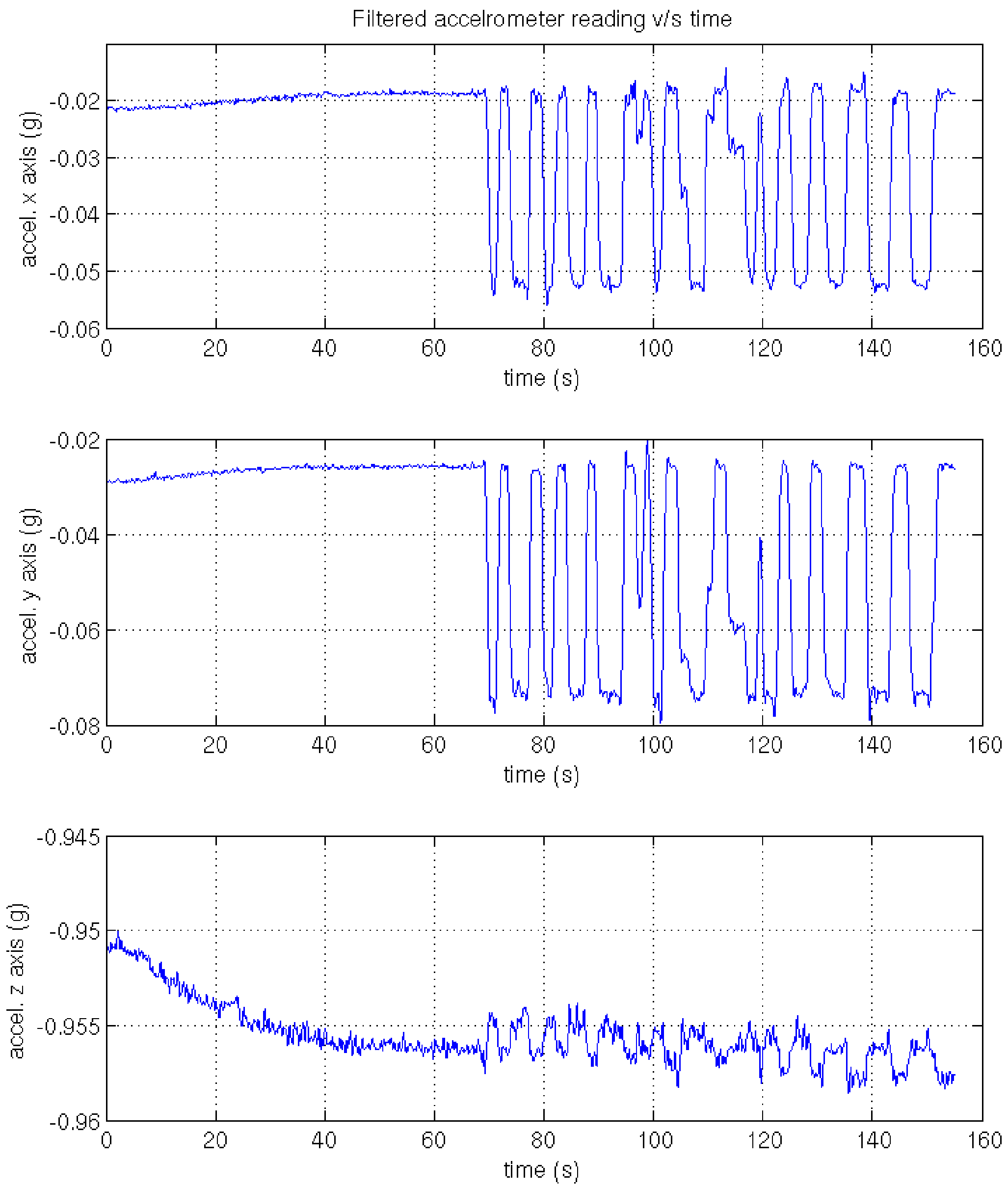


Figure 1.6: Same time series as in figure 1.5, but at the output of the LOWESS smoothing filter

Algorithm 1 Valve position monitoring algorithm

```
1: $initialize()$
2: $state \leftarrow IDLE$;
3: $valveAngle \leftarrow 0$;
4: $idleCounter \leftarrow 0$;
5: while $TRUE$ do
       $sleep(SAMPLING.TIME)$
6:     if $state == IDLE$ then
7:         $gyroData \leftarrow gyro.sample()$
8:         if $gyro.isMoving(gyroData)$ then
9:             $state \leftarrow MOVING$
10:            $rotatedGyroData \leftarrow rotate(gyroData)$
11:            $valveAngle \leftarrow integrate(valveAngle, rotatedGyroData)$
12:        end if
13:    else if $state == MOVING$ then
14:        $gyroData \leftarrow gyro.sample()$
15:        if $gyro.isMoving(gyroData)$ then
16:            $rotatedGyroData \leftarrow rotate(gyroData)$
17:            $valveAngle \leftarrow integrate(valveAngle, rotatedGyroData)$
18:        else if $++idleCounter \bmod SAMPLING.TIME == 0$ then
19:            if $abs(valveAngle) \leq 10^\circ$ then
20:                $accelData \leftarrow accel.sample()$
21:                $filteredAccelData \leftarrow filter(accelData)$
22:            end if
23:        else
24:            if $abs(valveAngle) \leq 10^\circ \& filteredAccelData == originalAccelData \pm \sigma_{noise}$ then
25:                $valveAngle \leftarrow 0$
26:            end if
27:            $state \leftarrow IDLE$
28:            $wireless.send(valveAngle)$
29:        end if
30:    end if
31: end while
```

load on the batteries (to extend device lifetime). A more thorough analysis of application energy requirements is presented in chapter 5.

1.4 Results

In this section, we present the results following a number of experiments run as described in section 1.1. In each of these experiments, the valve is positioned such that its plane of rotation intersects the horizontal plane at an angle. This means that the valve was not always horizontal, and that we positioned it at 20°, 40°, 60° and 90°. In each of these orientations, the valve was moved from one extreme to the other repeatedly, sometimes stopping at points in between and returning to the origin. Motion always started after a minute of inactivity at least, to allow the device to calibrate. Next we present the collected results for the part-turn and multi-turn valves separately.

1.4.1 Quarter-Turn Valve

Figure 1.7 shows the valve angle vs. time for a series of experiments. As expected, the valve angle varies between 0° and 90° (some of the time series were inverted since different starting angles were used). In all five time series, and during the first minute of the experiment, the valve angle remains at 0°, showing that the temperature compensation is working correctly. During the remainder of the experiment, we can observe that when the angle returns to a value close to zero, and as the valve stops moving, if the accelerometer detects a return to the origin, then the valve angle is adjusted accordingly. This is most apparent after the second peak of the last time series: about a second after the valve has returned to its original angle, the accelerometer forces the angle to return to zero, resulting in a small "bump" following the peak. On the figure the '$*$' labels indicates some instances of using the accelerometer to correct the valve angle. The 'P' labels correspond to a partial opening or closing of the valve. The same code was run on all of the time series, demonstrating that the results are repeatable and orientation independent.

Table 1.2 shows the results of 90 experiments in which a valve was turned to nine

Actual valve angle (in degrees)	10	20	30	40	50	60	70	80	90
Minimum measured angle (in degrees)	8.93	18.96	30.36	39.66	48.27	57.48	67.82	78.03	87.24
Maximum measured angle (in degrees)	11.2	21.64	33.18	42.64	52.59	63.74	72.17	84.21	94.22
Maximum recorded error (in degrees)	1.2	1.64	3.18	2.64	2.59	3.74	2.18	4.21	4.22

Table 1.2: Ball valve angle measurement results. For each target angle, the measurement was repeated 10 times. The maximum and minimum measured angles are reported, along with the error in absolute value.

different angles, ten times each. Every time, the angle obtained from the MEMS sensors was recorded and compared to the ground truth (accurate to within 0.5°). The table reports the minimum (and maximum) angle measured for each target angle, as well as the maximum deviation (in absolute value) from the real angle. The accuracy of the estimated angled obtained from this sensing methodology is ±5°.

1.4.2 Multi-Turn Valve

When presenting the results for the multi-turn valve, we decided to use a congruence modulo operator on the angle, so as to let it swing between −180° and +180°. This is a convenience measure because it allows us to easily count the number of turns in each motion. These results can be seen in figure 1.8. Defining a *full cycle* as a movement from one extreme to the other and back, each of the time series depicts two full cycles (8 full revolutions). As with the quarter-turn valve, the temperature compensation is obviously working, and the gyroscope only drifts when in motion. Again, the accelerometer is used to "zero" the angle when the resting position is reached. By design, the valve we installed in our experimental setup has both extreme positions pointing in the same direction. This is apparent in time series where both *fully open* and *fully closed* positions are zeroed. However, when one tightly closes the valve (as can be seen in the second cycle of the third and fifth series), the resulting heading is slightly away from the open position, which justifies the distancing from the original angle that is observed in some experiments. On the figure, a

Actual valve angle (in degrees)	90	450	810	1170	1530	1890	2250	2610
Minimum measured angle (in degrees)	86.902	434.04	797.34	1154.5	1513	1864.6	2227.7	2583.2
Maximum measured angle (in degrees)	94.332	464.41	816.21	1191.3	1557.9	1897.3	2274.3	2637.9s
Maximum recorded error (in degrees)	4.332	15.96	12.66	21.3	27.9	25.4	24.3	27.9

Table 1.3: Gate valve angle measurement results. For each target angle, the measurement was repeated 10 times. The maximum and minimum measured angles are reported, along with the error in absolute value.

'$*$' represents places where the valve returned to its original angle and the accelerometer data used to set the measured angle to zero. A 'T' shows when the valve was tightly closed and the angle therefore deviated from zero. This effect was investigated because a zeroing of the angle measurement was expected but not seen. Accurate angle measurements on the gate valve showed a noticeable deviation between the "loosely" and "tightly" closed states.

As with the ball valve, 80 experiments were performed on eight different target angles, with ten tries each. Table 1.3 lists the maximum and minimum angles as reported by the sensor board for each target. The maximum error is also shown in absolute value. The accuracy of the measurement for the multi-turn valve is of ±10% of a turn (or about ±1% of the entire range of the valve).

1.4.3 System Architecture

Our results show that with our algorithm and sensing platform, we can reliably determine the valve angle with no prior knowledge of the valve orientation or plane of motion, and with no calibration required. In our experiments however, the valve was always initially in either the "fully-open" or "fully-closed" positions. We cannot assume that this will be the case for real-world installations. In fact, in many situations, it is undesirable for the plant operators to change the position of the valve in order to install a new sensor. This means that the valve can be partially open or closed when the platform is installed. A problem then arises of keeping track of the valve position using only relative angles. This

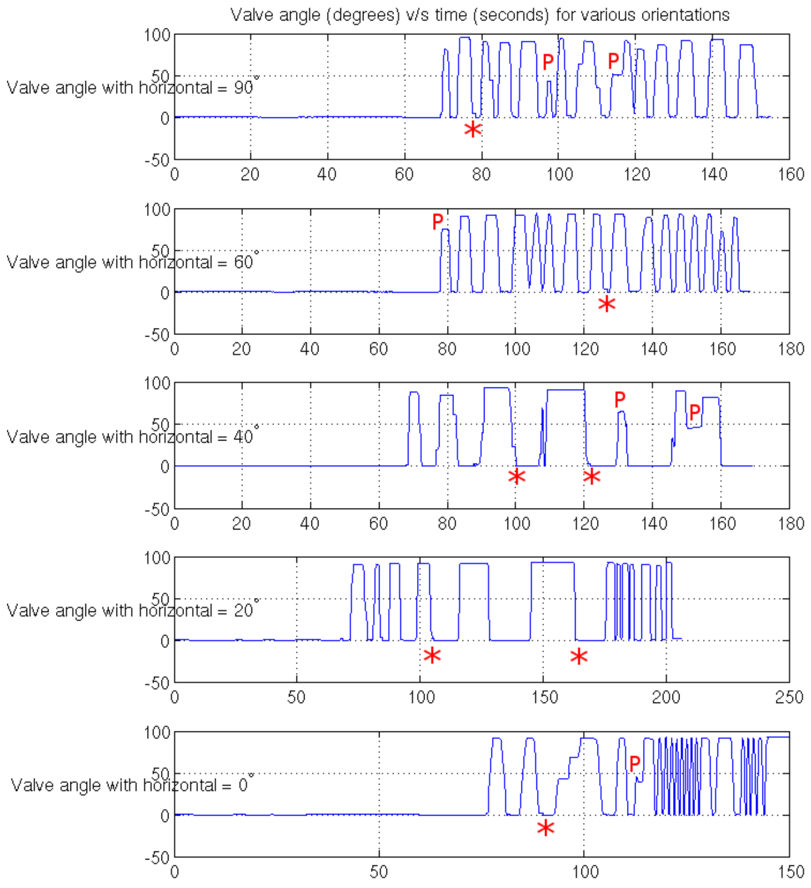


Figure 1.7: Results of five experiments run with the ball valve. In each time series, the first minute (approximately) of inactivity is reserved for temperature calibration. Then the valve is moved from one end to the other, with varying motion types. A '$*$' represents the places at which the accelerometer was used to remove the gyroscope drift.

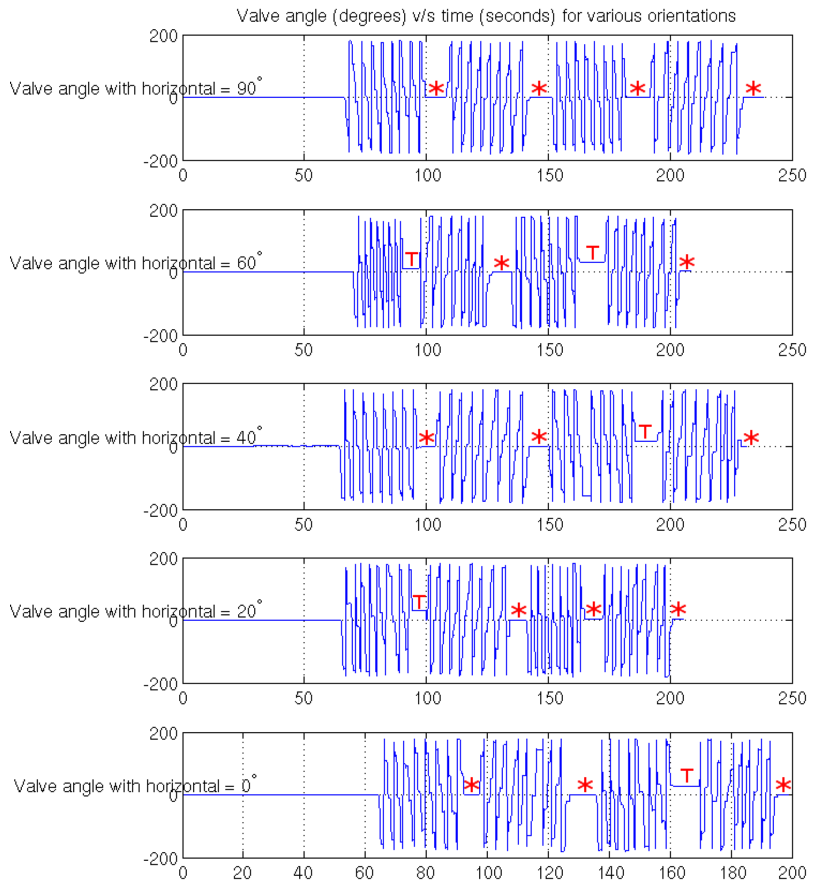


Figure 1.8: Results of five experiments run with the gate valve. In each time series, the valve is closed and opened twice (8 full turns each way). The '∗' sign shows alignment with the origin and a zeroing of the angle, while a 'T' marks a tight close.

can be easily solved however by tracking the evolution of the angle at the server side. As the valve is moved by plant employees, the sensing platform itself does not need to be aware of the position of the valve. The server-side application (or *Plant Management System*) will be responsible for figuring out this absolute position and raising alarms in case this position is problematic. Additionally, and as an improvement to our methodology, the algorithm can be augmented with a feature to dynamically adjust its saved accelerometer values that are used to "zero" the gyroscope drift. Another alternative would be to enable the server application to directly contact the valve and inform it of its absolute position.

1.5 Concluding remarks

In this chapter, we have presented a "peel and stick" solution to the valve position monitoring problem. Our MEMS based sensing device will ultimately be inexpensive to manufacture, as it uses sensors similar to the ones found in mobile phones and other consumer electronics products. In its present form, the board is already small enough for the majority of valves but can be made even smaller. Our solution is also calibration-free.

In terms of energy, the bulk of the consumption occurs at the gyroscope. Considering commercially available MEMS gyroscopes, our algorithm would consume about 2.74mA on average for a valve with moderate utilization (10 seconds of activity every hour). Adding 100μA to maintain a WirelessHART connectivity, and a battery pack of three D-cell industrial Lithium batteries (19Ah of charge each) the lifetime of the device would be around 2.3 years. Though this lifetime falls short of the 10-year goal and the device becomes bulky with a large battery pack, it is important to take note of the continuing advancements in commercially available MEMS gyroscopes. Energy consumption in these devices is decreasing every year, a trend also accompanied by a reduction in device turn-on time. Frequency-modulated gyroscopes [11] show promising energy and performance results, with orders of magnitude of improvement.

Finally, we have demonstrated that valve angle accuracy of $\pm5^\circ$ was recorded for quarter-turn models, while an accuracy of $\pm10\%$ of a turn (or about $\pm1\%$ of the entire range) was obtained with the multi-turn valves. Obviously, it is easy to retrofit existing part-turn and multi-turn valves with our device. However, it is equally simple to integrate with new valve designs. Our accurate angle measurements demonstrate a reliable replacement for today's aging valve monitoring solutions.

Chapter 2

Smart Fence: Decentralized Sequential Hypothesis Testing for Perimeter Security[1]

Since humans first transitioned from nomadic tribes to permanent settlements, we have had to find solutions to the essential problems associated with property: how to mark what is ours and how to protect it. In short, perimeter security has been an ever-evolving priority in human history. At their most basic, fences delineate land and serve as a marker of private property. Factor in a few technological innovations, and they can serve as a deterrent to intruders, or even an alarm system. However, even the strongest barriers have their weaknesses. Fences can be climbed over, dug under, and even cut through.

Many security systems were implemented in an attempt to reinforce the fence structure [14]. The most common fence line intrusion detection systems are taut wire, fiber optic,

[1]**This chapter was published in a similar form as follows:**
Chraim, F.; Pister, K.S.J., "Smart Fence: Decentralized Sequential Hypothesis Testing for Perimeter Security," Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Sensor Systems and Software, Vol 122, p. 65 - 78, 978-3-319-04165-0, 2013

strain-sensitive cable, electric field systems. However, each of those approaches comes with its own set of challenges. For example, taut wire setups are expensive, complicated to install, and require regular tensioning of the fence sections. Fiber optic fence sensors, which use light variations within the communication medium to detect movements, are susceptible to weather variations. Electric field systems, check for variation in the capacitance between two different points to detect intrusions. These installations suffer from Electro-Magnetic interference. Strain-sensitive cables, are not susceptible to such problems, but are instead sensitive to poor fence installations, as well as vegetation ingress onto the fences. All of the described security schemes require laying at least one cable around the facility to be monitored. This becomes quite expensive as the perimeter grows, and can also lead to issues with maintenance.

Our approach to the problem of fence intrusions in perimeter security is one of Wireless Sensor Networks (WSN) combined with cheap Microelectromechanical systems (MEMS) sensors. Essentially, we set up a low-power wireless network along a fence line, equip each fence section with inertial sensors and run detection algorithms to detect intrusions. This idea has previously been explored in [15], [16] and [17]. Yousefi et al [15] implement an accelerometer-based hardware platform which collects samples at 360 Hz. They focus on classifying the event type seen at the fence. They first measure the variance in the signal to determine whether any activity is occurring or not. If enough variance is detected in the signal, a filter-based Gaussian Mixture Models (GMM) classifier then decides between a climbing or rattling events. The results they collected show a good performance in accurately detecting activity at the fence (100% of the time), and classifying between rattling vs climbing (more than 90%). The authors further generalize the classifier to a non-homogeneous Hidden Markov Model in [16]. The improvement allows to further classify kicking, leaning, rattling, climbing and scratching. Both methods they present are however very computationally intensive, and would not be practical in a real, long-term deployment. Wittenburg et al [17] present a solution which is wireless and employs an accelerometer running at 41.6 Hz. The main contribution they present is a feature extraction method followed by a classifier. The authors also describe their training routines. They report a detection accuracy over 80% for leaning, kicking and climbing events. This method is again relatively involved, and could be energy inefficient on a limited microcontroller. The study does not justify the placement of the sensing device on the fence poles (instead of the fence web).

Our goal in this chapter is to detect all types of fence intrusion activities with no false negatives, while limiting false positives to less than (an arbitrarily chosen) 1/mile/month. We start by presenting the theoretical problem of sequential hypothesis testing in section 2.1, both for the centralized and decentralized case. We then describe the experimental setup both in terms of hardware and software and network architecture in section 2.2. Applying the theory to a practical algorithm is shown in section 2.3. Section 2.4 illustrates the results we obtained following several deployments.

2.1 Sequential Hypothesis Testing

Sequential analysis is the branch of statistics that deals with decision-making as the samples are being collected. It differs from classical hypothesis testing in that conclusions are reached more quickly, often before the end of the experiment. Consequently, it is ideal for detection, signal processing, clinical trials and other applications. In general, sequential decision problems involve one or more sensors and a fusion center where the final decision is made. In the centralized setting, all of the information received by the sensors is made available at the fusion center. However, in the decentralized case, the sensors themselves are part of the decision process and relay partial information rather than all of their observations [18–20]. The following two sub-sections go into some detail concerning both approaches. In this part, we will be following the study done in [21].

Depicted in figure 2.1, we see K sensors observing a stochastic process $\{\xi_t^i\}$ with prior P^i. The assumption we make here is that the observed processes are independent. Furthermore, we consider that there are two possible hypotheses in the system at hand

$$H_0 \text{ with probability } P = P_0 \text{ and } H_1 \text{ with } P = P_1 \tag{2.1}$$

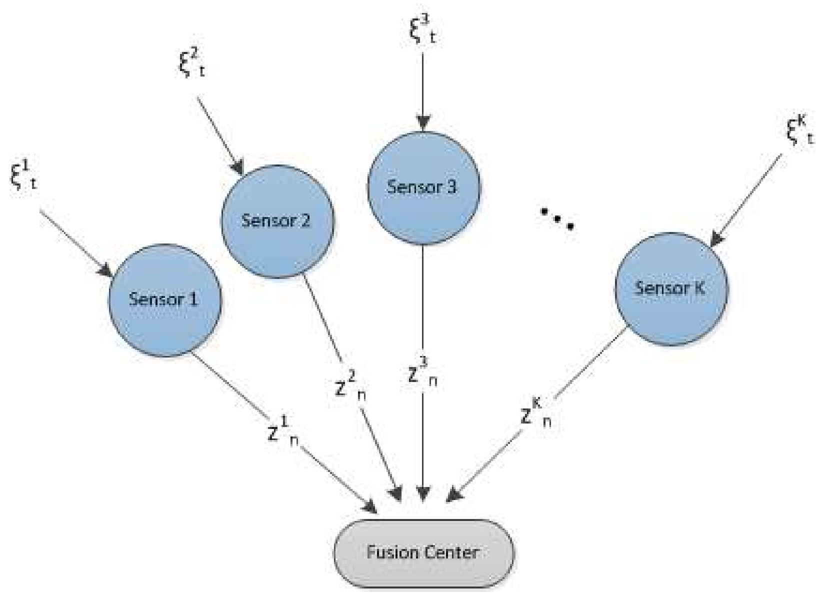


Figure 2.1: A system of K sensors and a fusion center

Additionally, $P_j = \prod_{i=1}^{K} P_j^i$ for $j = 0, 1$. The next step would be to define the local log-likelihood ratio that takes place at each of the sensors.

$$u_t^i = log\frac{dP_1^i}{dP_0^i}(\xi_t^i) \text{ with } u_0^i = 0 \tag{2.2}$$

Independence between the observed processes allows us to write the following log-likelihood ratio that takes place at the fusion center,

$$u_t = log\frac{dP_1}{dP_0}(\xi_t^i) = \sum_{i=1}^{K} u_t^i \text{ for } 0 \leq t < \infty \tag{2.3}$$

The aim is to optimally define the pair (T, d_T) where T is the stopping time and d_T is the decision, which takes values 0 or 1, depending on which hypothesis was picked. We attempt this both in the centralized and decentralized cases.

2.1.1 Centralized Sequential Hypothesis Testing

It has been shown by Wald and Wolfowitz [18] that the Sequential Probability Ratio Test (SPRT) is optimal in solving the centralized case of the following problem.

Given, the type-I and type-II probability levels $\alpha, \beta > 0$ such that $\alpha + \beta < 1$, we want to find $(\mathcal{T}, d_T)$ such that

$$E_j[\mathcal{T}] = inf E_j[T], j = 0, 1 \tag{2.4}$$

The SPRT mentioned above is defined in this case as,

$$\mathcal{T} = inf\ t > 0 : u_t \notin (-A, B) \tag{2.5}$$

$$d_{\mathcal{T}} = \begin{cases} 1 & \text{if } u_{\mathcal{T}} \geq B \\ 0 & \text{if } u_{\mathcal{T}} \leq -A \end{cases} \tag{2.6}$$

$$A = log(\frac{1-\alpha}{\beta}), B = log(\frac{1-\beta}{\alpha}) \tag{2.7}$$

Hence, the procedure for applying this test is to obtain the observations from the sensors, apply the global log-likelihood ratio, add it to the previous value and verify whether this sum left the open interval $(-A, B)$ or not. The first time at which we leave this interval will be time $\mathcal{T}$, and the associated decision $d_{\mathcal{T}}$ will follow the rule in (2.6). This test is optimal under our assumptions. We will see in the next sub-section how to generalize this concept to the decentralized case.

2.1.2 Decentralized Sequential Hypothesis Testing

This decentralized problem is approached from the discrete time case. Though the continuous time case is studied in [21], it is not of major practical use. Rather it provides some intuition on treating its discrete time equivalent. As we have mentioned before, the difference between the centralized and decentralized cases is that, in the latter, the sensors make decisions before relaying some information back to the fusion center. Indeed, each sensor computes a local log-likelihood ratio,

$$l_n^i = log\frac{dF_1^i}{dF_0^i}(\xi_n^i) \tag{2.8}$$

Now, we set two thresholds $-\underline{\Delta}_i, \bar{\Delta}_i$,

$$-\underline{\Delta}_i = log\frac{P_1}{P_0}(\xi_n^i = 0), \bar{\Delta}_i = log\frac{P_1}{P_0}(\xi_n^i = 1) \tag{2.9}$$

This allows us to define an SPRT that occurs at each sensor as follows,

$$z_n^i = \begin{cases} 1 & \text{if } u_{\tau_n^i}^i \geq \bar{\Delta}_i \\ 0 & \text{if } u_{\tau_n^i}^i \leq -\underline{\Delta}_i \end{cases} \tag{2.10}$$

where τ_n^i is the local stopping time at the sensor and z_n^i is the information sent to the fusion center at that time. In order for the fusion center to compute its log-likelihood ratio based on the z_n^i, we could envision that the following two values be precomputed and made available at that fusion center:

$$-\underline{\Lambda}_i = log\frac{P_1}{P_0}(z_n^i = 0), \bar{\Lambda}_i = log\frac{P_1}{P_0}(z_n^i = 1) \tag{2.11}$$

Now, the reason the pair $\underline{\Lambda}_i, \bar{\Lambda}_i$ is defined separately from the pair $\underline{\Delta}_i, \bar{\Delta}_i$ resides in the fact that discrete time sampling gives rise to an *overshoot effect*. One can think of this effect as an uncertainty in the exact time the local log-likelihood ratio crossed the open interval $(-\underline{\Delta}_i, \bar{\Delta}_i)$ for the first time.

In turn, the fusion center will use the information provided by the sensors to update its global log-likelihood ratio.

$$u_n = u_{n-1} + \begin{cases} \bar{\Lambda}_i & \text{if } z_n^i = 1 \\ -\underline{\Lambda}_i & \text{if } z_n^i = 0 \end{cases} \tag{2.12}$$

The main result of Fellouris and Moustakides in [21] is first to define a measure of the D-SPRT thresholds at the fusion center,

$$\tilde{A} \leq |log\beta|, \tilde{B} \leq |log\alpha| \tag{2.13}$$

then to show the following asymptotic optimality on the global log-likelihood ratio:

$$|E_0[u_{\tilde{\mathcal{T}}}] - E_0[u_{\mathcal{T}}]| \leq \frac{O(\theta)}{\Theta(\Delta)}|log\beta| + \Theta(\Delta) \tag{2.14}$$

$$|E_1[u_{\tilde{\mathcal{T}}}] - E_1[u_{\mathcal{T}}]| \leq \frac{O(\theta)}{\Theta(\Delta)}|log\beta| + \Theta(\Delta) \tag{2.15}$$

where θ is the maximum overshoot (formally defined in [21]), Δ is equivalent to any of the two $\underline{\Delta}_i, \bar{\Delta}_i$ assumed to be equal, and $\tilde{\mathcal{T}}$ is the stopping time for D-SPRT at the fusion center, while $\mathcal{T}$ is the optimal stopping time for the centralized case. In fact, one can readily see that because of the loss of information and loss in time resolution between the centralized and decentralized cases, the optimality is lost. Equations (2.13) and (2.14) represent the Kullback-Leibler divergence (which can be thought of as the relative entropy between an optimal distribution and a sub-optimal one) applied to the log-likelihood ratio. The authors in [21] go on to simulate and observe that the D-SPRT derived in this section is useful in most practical implementations.

To apply the presented methodology to our particular application (fence line intrusion detection), only two thresholds need to be set: $\bar{\Delta}_i$ and $\bar{\Lambda}_i$. This is done by placing a sensor on the fence, and running controlled experiments to compute the probabilities of both hypotheses, given the sensor reading. By assuming that the fence construction is uniform (not always true) in a particular site, this "calibration" step does not need repetition. These thresholds can be tuned either at the device side or the fusion center. This process translates to the trade-off between detection delay and false alarm rate.

2.2 Experimental Setup

We now turn our focus to implementing a solution to the perimeter security problem at hand. Tackling fence monitoring was not a straightforward task. In fact, fence models were not readily available, which meant that building any detection infrastructure needed experimentation. We now present the hardware platform of choice along with the underlying software architecture.

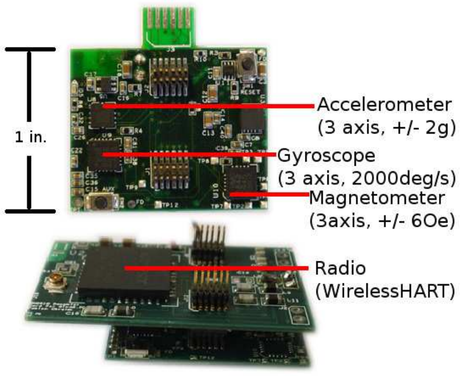


Figure 2.2: The hardware platform: GINA/WHIM

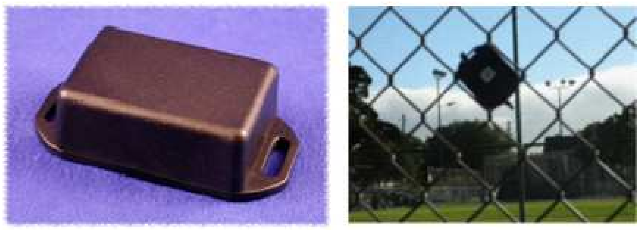


Figure 2.3: The IP-65 enclosure (rain and dust resistant) that houses the hardware platform [courtesy of Hammond Inc] shown next to one sensor as deployed in the middle of a fence section

2.2.1 Hardware

The hardware platform used here is again GINA (presented in chapter 1). As can be seen in figure 2.2, a daughter card is mounted on top of the GINA board. This daughter card is referred to as the WirelessHART Interface Module (WHIM) and carries the DN2510 radio by Dust Networks. This radio is WirelessHART compliant and operates in the 2.4 GHz band. The GINA/WHIM combination consumes a few microwatts in sleep mode and, when it is running at full capacity, currents around 20mA were recorded at 3V. An IP-65 enclosure (shown in figure 2.3) completes the hardware solution fitting nicely in the diamond pattern of the chain-link fence.

2.2.2 System Architecture

As the sensors are placed along the fence line, they join the WirelessHART network formed by the gateway, and start reporting data there. Looking at figure 2.4, we can see that a computation element exists both at the sensor side and at the gateway as well. This means that any detection scheme can implement one of the following three models:

- server-side computations only
- sensor-side computations only
- hybrid model with computation taking place at both ends

Choosing one of these three methods is dictated by the application energy requirements. For the first model, an intrusion of five seconds which triggers three sensors generating two bytes of sensor data at 70 Hz, would result in about 5 packets per second for a WirelessHART network. If one were to use the second model however, the sensors would have to be programmed to communicate between each other to take full advantage of the distributed sensing aspect. Finally, the hybrid model would require the wireless devices to *summarize* the data before transmitting only a minimum amount of packets to the server. As will be seen in chapter 5, it is often preferable (in terms of energy) to perform simpler operations locally on the microcontroller rather than transmitting the raw data. This makes the first model difficult to justify, especially that it could also suffer from congestion: if three people attempt to climb the fence line simultaneously at three different locations, a surge of about 15 packets per second could be problematic in certain deployments.

In terms of routing, it is often easier to collect data at the sink in low-power mesh networks, as opposed to having peer-to-peer communication. Additionally, the sheer presence of two sensors in close proximity does not guarantee that they will be only one hop away within the network. As a result, the third (hybrid) model is preferred over the second

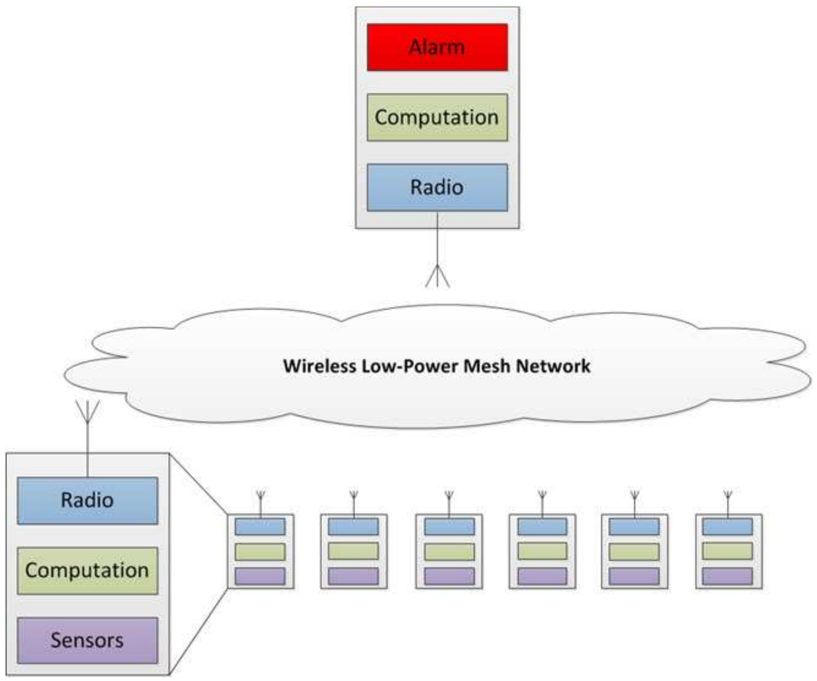


Figure 2.4: System architecture for the Smart Fence. The detection algorithm can run only on the motes, or only on the server, or using a hybrid approach where some computations happen in the motes and their output is relayed to the server to make the final decision.

(purely distributed). In our particular topic, the hybrid model also works well with the detection algorithm.

2.3 Detection Algorithm

In this section we explore the procedure we followed to develop the detection algorithm for fence monitoring.

2.3.1 Preliminary Testing

Some of the primary questions to answer were the following: Which of the sensor data generated by GINA is relevant? How fast should we sample our sensors? To come up with a solution, we strapped three of the platforms we developed on three contiguous fence sections. All of the sensors (accelerometer, gyroscope, magnetometer...) were then sampled at 300

Hz and the raw data transmitted to a gateway. We then ran controlled shaking, kicking and climbing tests. A singular value decomposition of the sampled sensor data containing all axes enabled us to single out only three axes of interest, as can be seen in figure 2.5. The data is normalized against the noise and, after the singular value decomposition is performed, the data is projected along the axes of highest variance. Not surprisingly, it turned out that most of the information is contained in the z-axis of the accelerometer (pointing out of the plane of the fence), and the x-axis and y-axis of the gyroscope (in the plane of the fence). The result is intuitive, and the reader is invited to imagine a chain-link fence vibrating under the influence of shaking for example. Clearly, the accelerometer will see accelerations dominantly in the z-direction, while the angular rates observed by the gyroscope will mainly be along the plane of the fence itself. For the purpose of this application, the data generated by the acceleromter z-axis was sufficient to detect intrusions. The next step in the analysis was to determine the sampling frequency. A quick look in the frequency domain revealed that most events of interest happened below 35 Hz, justifying our sampling frequency of 70 Hz. An additional tradeoff can be observed here. Increasing the sampling frequency obviously yields better results, in terms of capturing all of the information contained in the the waveform. However, the added data generated by this oversampling has to be processed either at the sensor or server side. This means that energy will be spent either on the on-board microcontroller or during transmission.

The preliminary testing phase also showed that strong gusts of wind were problematic at the fence line. As a matter of fact, the accelerations recorded during those events were comparable to those recorded after a person gently shook the fence. For this reason, we made the decision to apply Lebesgue sampling and set a threshold under which, the entire platform enters sleep mode.

2.3.2 Applying D-SPRT to Fence Monitoring

The process of applying D-SPRT in our case consists of selecting the probability priors both at the sensor side and server side. This was done by running controlled tests along the fence line. Similar to the previous section, intrusion traces were compared with "background

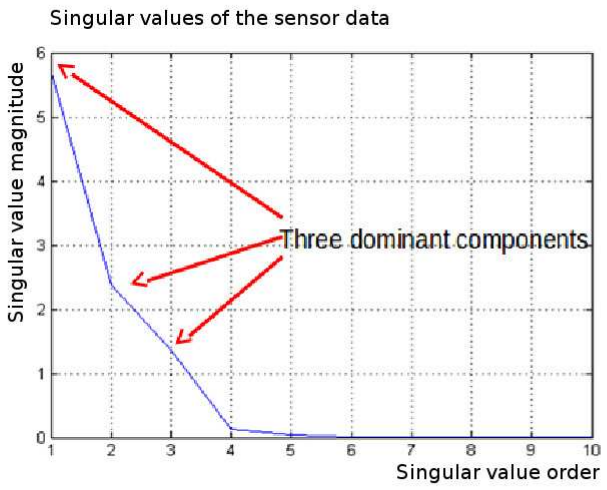


Figure 2.5: Singular value decomposition of the sensor data including all axes. This plot shows that only three axes contain the majority of the information. Namely, the z-axis of the accelerometer into the fence and the x-axis and y-axis of the gyroscope in the plane of the fence.

noise" traces. With enough repetitions, the probabilities could be estimated. This process was necessary every time a deployment at a new location took place. Furthermore, even within the same fence line, not all fence sections were similar in their response to intrusions. Because of construction limitations, certain sections were sturdier, while others oscillated more vigorously when shaken. However, it was not necessary to repeat this "calibration" process for each fence section.

Figure 2.6 shows log-likelihood ratios recorded at the sensors during experiments we performed on fences. As can be seen in the left-most figure, every time the upper limit of the interval is crossed, a value of H_1 is transmitted to the server and the ratio gets reset. The middle graph shows the case where the sensor records activity but deems it regular noise. A value of H_0 gets transmitted and the ratio is reset as well. The right-most figure shows the case where the sensor starts recording data, but the traces end before the interval is left and no packets are transmitted to the server. This corresponds to very small disturbances for example. Figure 2.7 shows the progression of the log-likelihood ratio over time for a

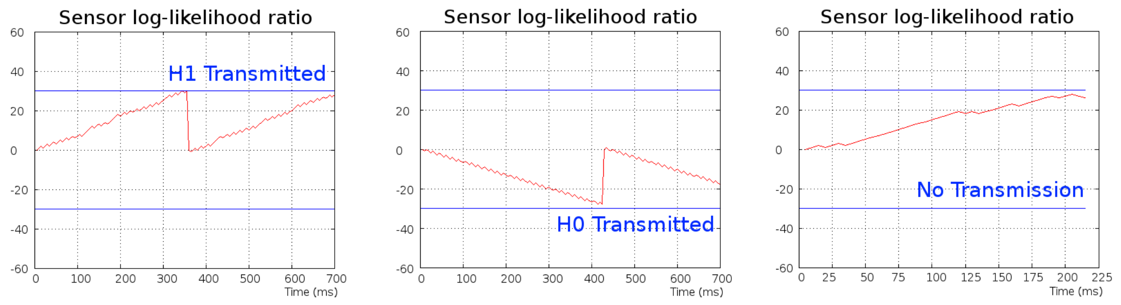


Figure 2.6: Sensor log-likelihood ratios for various disturbances, in the short run. When the ratio crosses the upper limit, an H_1 is transmitted to the server. This is then added to the global log-likelihood ratio that can trigger the alarm or ignore the sensor output

climb signal vs a kick signal. It can be readily seen that the climb signal generates several H_1 packets sent to the server side, while the kick signal only generates a couple before the ratio decays and the sensor stops recording. At the server side, a decision is made based on the number of H_1 and H_0 packets received. Similarly to the likelihood ratio computer at the device side, the server increments/decrements a counter each time an H_1/H_0 packet is received. It is important to know that for this to work, information about the location of the sensors needs to maintained at the server. If two sensors separated by a large distance are reporting information, they should each get a different probability ratio test since they would not necessarily be observing the same event.

2.4 Deployments and Test Results

Several deployments were made to put the Smart Fence system to the test. They varied in longevity, environmental conditions and fence construction. In all of them, different individuals were asked to climb the fence. Shown in figure 2.8 are the server side log-likelihood ratios for various climbers under varying conditions. Clearly we can see that the output of our detection algorithm is not the same in all of the time series. The main reason

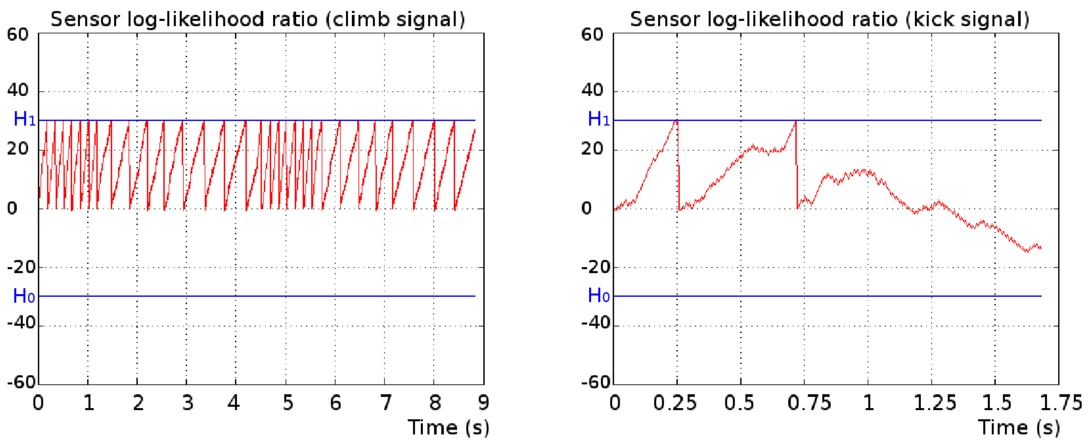


Figure 2.7: Sensor log-likelihood ratios for a climb and kick signal, in the long run. In reaction to climbing the fence, the sensor reports a series of H_1 consecutively as it seeing a lot of activity. For the kick stimulus however, the sensor may report a couple of H_1 packets for example, before its log-likelihood ratio decreases.

is, as stated before, not all climbing styles are equivalent. While some people tend to be aggressive, and generate a lot of activity at the fence line, others choose a calmer approach and climb more methodically.

Table 2.1 lists the parameters and results of four deployments. The first two deployments were short in length (<1 hour), while the other two lasted for days. The longest deployment was performed at the Chevron refinery in Richmond, CA. During that test, four sensors were deployed along the North-East fence line of the Technology Center for a period of about six weeks. Shown in figure 2.9 is the network manager and Linux box running the detection algorithm overlooking the sensors from one of the offices in that building. Some refinery employees were asked to disturb the fence line during the deployment period and record the time and date of that activity.

In each of the experiments listed in table 2.1, the dimensions and construction of the fence sections were different. Out of the 91 known incursions, 100% were detected with no false alarms. The average detection time listed in the table takes into account the sensor sampling, device-side algorithm, the transport delay in the network, and the time to raise

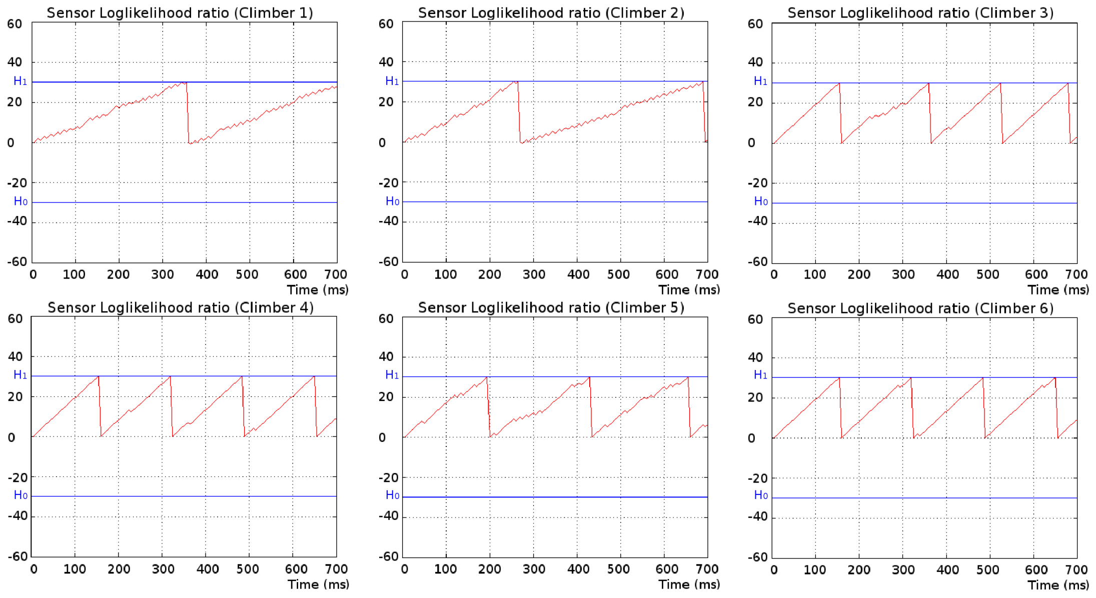


Figure 2.8: Log-likelihood ratios for different climbers and conditions. This figure shows that even with different climbing styles, our detection algorithm reported the intrusion within a few milliseconds of its start.

the alarm at the fusion center. On average, alarms were raised within two seconds of the incursion.

In terms of energy, the bulk of the consumption went to the network in this application. Since the hardware platform was in sleep mode for the majority of the time, the microcontroller and sensors were consuming about 6μA. The networking component used about 100μA on average, meaning that with a small 160mAh lithium battery, the application lifetime was around 60 days. However, using a standard industrial lithium C-cell battery of 9Ah, the lifetime could be extended to about 8.5 years.

Deployment Number	1	2	3	4	**Total**
Deployment duration	<1 hour	<1 hour	5 days	44 days	59 days
Number of fence sections/sensors	3	3	6	4	16
Fence dimensions (HxW in m)	2.1 x 3	1.6 x 1.6	1.8 x 2.1	2.1 x 3	n/a
Known incursions	18	21	39	13	91
True positives	18	21	39	13	91
False positives	0	0	0	0	0
First alarm delay (average)	1.1 s	0.92 s	2.3 s	1.8 s	1.7 s

Table 2.1: Test results from four deployments of varying parameters. A detection rate of 100% was achieved with no false alarms for the 91 known incursions. A detection delay of about 2 seconds on average was achieved.

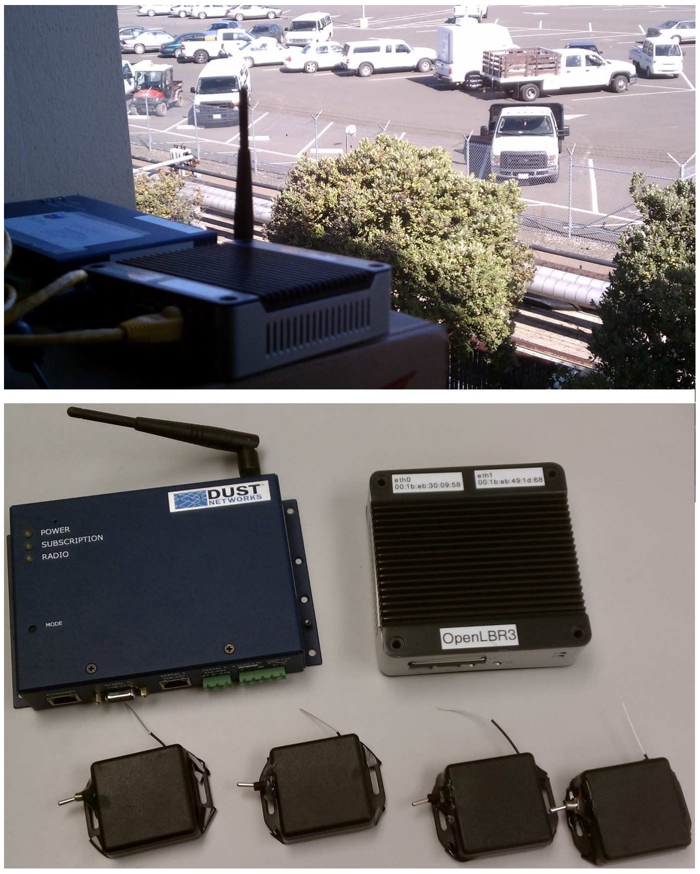


Figure 2.9: Long-term deployment setup at the Chevron-Richmond refinery. The result of this test was a detection rate of 100% with no false alarms. The sensors withstood strong winds and rainy weather.

2.5 Concluding Remarks

In this chapter, we have demonstrated that the Smart Fence is a viable solution to the perimeter security problem of chain-link fences. We applied a decentralized sequential hypothesis testing methodology to detect intrusions at the fence line. We also presented a low-power sensing platform which communicates over a reliable mesh network (WirelessHART). Following short-term and long-term deployments, we recorded a 100% detection rate for the 91 known intrusions, and no false alarms over a period of about 2 months. Though these results are inline with our initial objectives (no false negatives, and less than 1 false positive/mile/month), more testing would be beneficial for additional validation. In terms of energy consumption, and using the most up-to-date hardware, a lifetime of more than 10 years can be achieved using a standard lithium C-cell battery of 9Ah (8.5 years could be achieved with the presented hardware). By utilizing cheap off-the-shelf microcontrollers, MEMS sensors which are present in most cell phones, and an open-source implementation of the mesh network (namely OpenWSN [22]), a hardware cost of about $20 per unit can be achieved. Finally, it is worth mentioning the solution proposed in [23] which is similar in terms of hardware to the implementation presented in this chapter. In addition to the accelerometer, the commercial solution also features a passive infrared (PIR) presence sensor which is used to increase the detection confidence (at an energy cost of course).

Chapter 3

Wireless Gas Leak Detection and Localization

The number of gas leaks that occur every year on industrial plants is unknown. Most of these leaks, even if detected, go unreported when they don't directly lead to tangible accidents. Environmental Protection Agency (EPA) reports estimate that, in the United States alone, these plants emit close to one billion cubic meters of methane (not taking any other gas into consideration). Most of these losses (around 80%) seem to come from leaky compressors, valves, seals and connectors [24]. In 2012, approximately 2,200 million metric tons of CO_2 equivalent were accidentally released from petroleum systems and other chemical processes necessary for the production of plastics, cement, iron and steel [25]. It is estimated that around 800,000 to 900,000 leaks are investigated each year on refineries, with between 200 and 300 of them having directly resulted in loss of life, injuries, damaged equipment, or operational losses [26]. In short, industrial gas leaks present a major challenge in the quest for safe, environmentally-friendly, and cost-effective plants.

In this chapter, we present a distributed wireless sensor approach to the problem of gas leaks in large industrial spaces (chemical plants, refineries, oil rigs, etc.). The objective is to detect and localize "refinery-like" gas leaks within seconds of their occurrence. With many corporations upgrading their facilities with a low-power wireless infrastructure (WirelessHART) [27], a leak detection system that simply connects to the wireless umbrella would be a desired addition to the existing safety framework. Our goal is to study the feasibility of such an approach, while carefully reviewing some of the implementation challenges in the hope for opening the door for widespread commercial adoption.

The remainder of this chapter is organized as follows. In section 3.1 we will review some of the available solutions to the problem at hand, both at the academic level and commercially. Our approach is then presented in section 3.2, where we look at the system architecture, the hardware and the detection/localization algorithms. Our experimental results are shown in section 3.3, in which we validate our approach. Future directions and recommendations are left for section 3.4 where we conclude.

3.1 Literature Review and State of the Art

The review presented in this section is divided into a survey of commercial methods for leak detection, and some of the ideas coming out of academia.

3.1.1 Gas Leak Detection Systems

Conventional leak detection methods fall under two categories: fixed instrumentation and mobile sensing. In the former, a sensor is affixed in the general vicinity of equipment suspected of leaking (valves, compressors, ...). These instruments are usually connected to a constant power source and generate alarms based on their sampled data. These alarms can be visual or audible, or can feed directly into a plant management system. Mobile sensors are usually hand-held devices that a worker has to point at the suspected leak source and evaluate the readings on the spot. Reports of the measurements are relayed in real-time either through a wireless connection, or by direct communication between the worker and other plant employees. Both these methods have their advantages and disadvantages and most often, a hybrid system of fixed and mobile sensors is implemented. In particular, a fixed sensor is able to continuously monitor an area, as opposed to a worker who samples the same region for a few seconds perhaps before moving on. Fixed sensors have better instruments by virtue of the fact that they are less constrained, but mobile sensors allow the operator to trace a leak to its origin. It is obvious that mobile sensors put the worker at risk during the sampling process, while the fixed sensors enable safer operation [28].

In this study, we are only interested in fixed instruments because our proposed solution is static in nature. We now look at some of the commercially available solutions for comparison. Many solutions have been proposed for the problem of leak detection in pipelines. This topic, though relevant, is not of direct interest here since leak detection near pipelines can be accomplished by deploying a series of sensors in a linear sequence. The interested reader is invited to glance at [29]. The solution presented in this chapter would not be very practical for long pipeline installations due to the high number of sensors it would require.

Perhaps, the most prevalent leak detection methodology is by concentration measurement. Pellistor, electrochemical, semiconductor and infrared sensors are all used to sample the ambient gas for particular species. By means of preset threshold detection, alarms are raised alerting workers and plant operators [30]. These widely adopted sensors however suffer from one or more of the following: low sensitivity, short lifetime, high energy consumption, sensitivity to ambient conditions, high costs, drift... Typically, these sensors are operated independently, meaning that no information about the source of the leak is given. Due to the fact that they consume a considerable amount of energy, installing them becomes an issue as virtually always, the cost of laying cables outstrips the cost of the device itself.

Pipeline diagnosis systems gave rise to ultrasonic sensors which have recently been adopted in some plants. The principle of operation relies on the fact that gas leaks sometimes come from punctured pipes which emit acoustic "tone" signals in the ultrasonic band [31]. These sensors, though unaffected by environmental conditions, do not measure the intensity of the leak, and are still unable to determine its origin. They are designed to work with gases under pressure, and do not represent a general solution to industrial gas leaks.

In recent years, camera systems have found their way to the gas leak detection and localization market. These devices tend to be mounted on elevated towers, often rotating to cover the entirety of the plant. They operate by taking snapshots of the environment, then analyzing the sampled images to detect gas leaks [32]. Most of the available solutions on the market operate in the infrared band, but recently more versatile snapshot hyper-spectral instruments are being utilized.

3.1.2 Detection and Localization Using Multiple Sensors

Academically, the problem of detecting and localizing leaks has been addressed in many fields. Under different names, similar methodologies have been applied to detecting the location of a speaker using many microphones, localizing objects using multiple radar streams, etc. We now list of few relevant examples.

Glenn et al. propose using inverse diffusion modeling to localize leaks. By assuming a fickian diffusion model, they consider a large network of sensors surrounding the source of the leak. Coupling diffusion with Ensemble Kalman filtering allows them to estimate the location of the source of the leaks. Their simulated system reports plume origins as numerous hypotheses each having likelihoods [33].

Huseynov et al. [34] propose a distributed network of MEMS ultrasonic sensors for gas leak localization. In their study, a comparison of energy-decay (ED) and time-difference of arrival (TDOA) methods for localization is presented. With a distributed network of four devices, they attempt to localize a nitrogen leak from a small orifice. They employ maximum likelihood (ML) and the least squares (LS) techniques to find closed form solutions for the diffusion differential equations. In their deployment, a 20 ft 20 ft room is instrumented with 4 MEMS microphones (running at 200 kHz). Nitrogen gas was released at 150 psi at four different locations. They successfully localize the nitrogen leak with an accuracy lower than 1 ft.

Weimer et al. consider gas leaks in wide and dense wireless sensor networks. The problem being addressed is one of large scale leaks, with high concentration of gases (such as harmful gases in a metropole). In their model therefore, they take diffusion and air currents into account. An interesting idea is presented concerning the subsampling of sensors which are in close proximity, in order to reduce the network-wide energy consumption. A wake-up process ensures that all the devices are running when needed. Their method combines binary hypothesis testing with Kalman filtering [35].

The methods presented in these articles (and more on the topic of gas leak detection and localization) have merit as they present valid and interesting theoretical ideas and simulations. However, they all leave much to be desired in the space of experimental validation. In this study, we attempt to solve the problem of gas leak detection and localization in the most applied manner possible. We employ some mathematical and statistical tools, and apply them to real gas concentration measurements recorded during a series of intentional releases performed in a typical industrial setting.

3.2 A Wireless Distributed Sensing Approach

In this chapter, we study the problem of gas leak detection and localization by means of a wireless, distributed network of sensors. Though such an approach could be viewed as a standalone system of sensors, it could also benefit from further integration. The gas sensors utilized here could be tacked on to other industrial process control instruments. For example, the wireless valve positioning solution presented in [36], could be augmented with a gas leak sensor, as both of these problems are often linked. Furthermore, a wireless perimeter security solution [37] could also assist in detecting and localizing plant leaks as soon as suspicious concentrations leave the enclave of the plant. In the remainder of the chapter, we consider the leak detection solution as a stand-alone system.

3.2.1 Hardware

As is customary with most wireless sensing applications, the hardware we developed in this study includes a radio, a microcontroller, a sensor and a power source. Recently, the push for higher integration in the industry resulted in system-on-chip (SoC) solutions for the microcontroller and radio. In this project, the Linear Technology LTP5902 SmartMesh WirelessHART Mote Modules were utilized. They feature a 32 bit ARM Cortex M3 microcontroller, along with a 2.4 GHz, IEEE 802.15.4, WirelessHART (IEC62591) compliant radio. Combining the time-synchronized channel hopping protocol with extremely low transmission and reception power levels (on the order of 5mA at 3V), these modules achieve a 99.999% network reliability with a sub-50μA average currents.

In this project, the focus is on explosive gases. For this reason, all of the studies and releases were performed around propane, which is a by-product of natural gas processing and appears in the process of refining petroleum. This gas is representative of the family of explosive gases, and was made available to us for use in the various experimental stages. Propane is commonly used in commercial and residential applications, and is in the liquefied petroleum (LP) gases family. As such, a propane sensor needed to be integrated with the LTP5902 SoC. The Dynament Premier Infrared Hydrocarbon (propane) Sensor (MSH-P/HC/NC/5/V/P) was selected, with a 0-2% volume measurement range (or 0 - 20,000 part per million (ppm)) and a resolution of 0.01% volume (100 ppm). Our integrated device is shown in figure 3.1. Though this propane sensor is, at the time of writing, one of the best on the market, its performance leaves a lot of room for improvement. For starters, it consumes an average of 80mA of current (at 3V), and possesses a start-up time of 1 minute. This means that for the purpose of gas leak detection, this sensor cannot be duty-cycled and needs to remain on continuously. This certainly represents a challenge in battery operated devices. Additionally, the sensor response seemed highly dependent on the ambient temperature. As gas leak detection applications are mostly outdoors (the indoor counterpart is usually simpler especially in smaller monitoring areas), any small wind current or gust altered the measured concentrations by hundreds of ppms. Figure 3.2 shows the response of the gas sensor to a 100ppm propane calibration gas. The sensor was sampled every second and the ambient temperature was measured to be 23°C. Shown in figure 3.3 however, is the response of two identical sensors deployed outdoors in the vicinity of a leak source. It is clear that state-of-the-art propane sensors still present many challenges, and we hope to demonstrate that they are the missing link in the productization of this solution.

Considering the average power levels of the communication module, and with careful application design and network configuration, it is often possible to implement wireless sensing applications where lifetimes extend beyond the shelf life of batteries (around 10 years). However, in this wireless gas leak application, the sensor remains as the limiting factor and

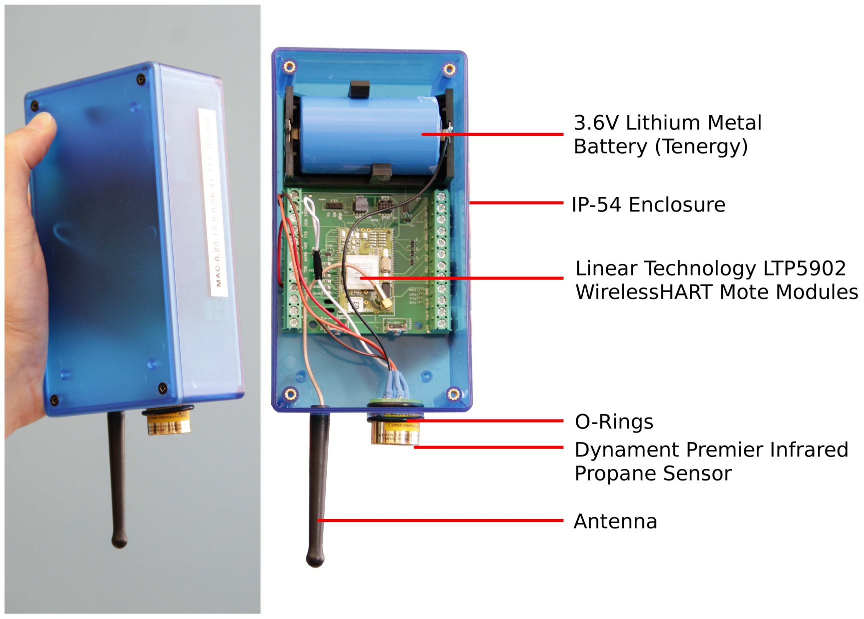

3.6V Lithium Metal Battery (Tenergy)
IP-54 Enclosure
Linear Technology LTP5902 WirelessHART Mote Modules
O-Rings
Dynament Premier Infrared Propane Sensor
Antenna

Figure 3.1: The hardware platform used in this study: a wireless, battery-powered propane sensor.

a burden on the energy budget. Finally, it is worth noting that the hardware was powered by an industrial D-size Lithium metal battery, with 19 Ah of charge (with a duty-cycling of 25%, this battery would hold enough charge for about 40 days). The device was enclosed in an ABS plastic IP-54 enclosure by Hammond.

3.2.2 System Architecture

The system architecture for the leak detection solution is perhaps best explained with a picture (see figure 3.4). Taking a refinery for the sake of example, gas leak detection sensors would be deployed throughout a refinery. Though a grid distribution is often easier to manage, it is not a required feature. Indeed, these easy to install sensors can be mounted in seconds to the existing buildings and poles, and as long as their location is recorded, the concentration data will be easily processed by the algorithms presented later. The path from data to decision starts with concentration measurements at the node side, which are filtered and transmitted to the gateway (when needed), using the wireless infrastructure. The gateway algorithms then collect the concentration data from the many sensor devices

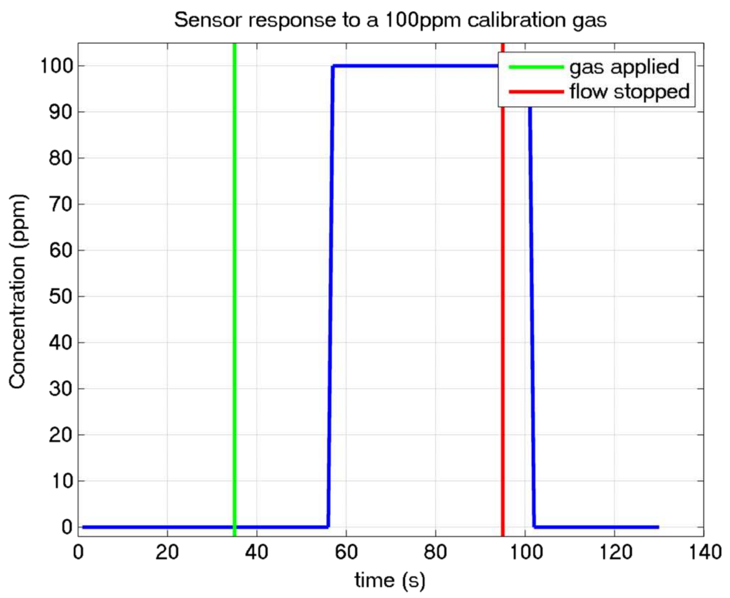
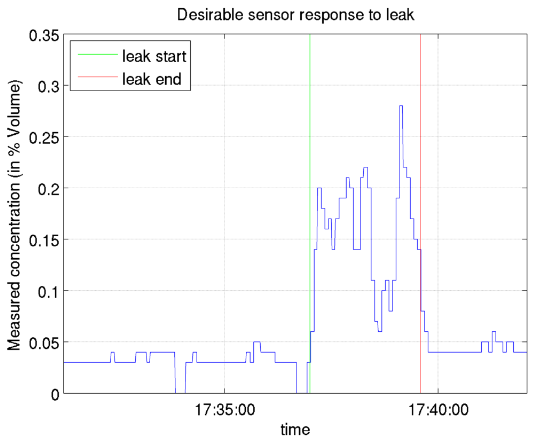


Figure 3.2: Calibration plot showing the sensor response to a calibration gas of 100ppm of Propane. The test data was collected indoors at a temperature of 23°C.

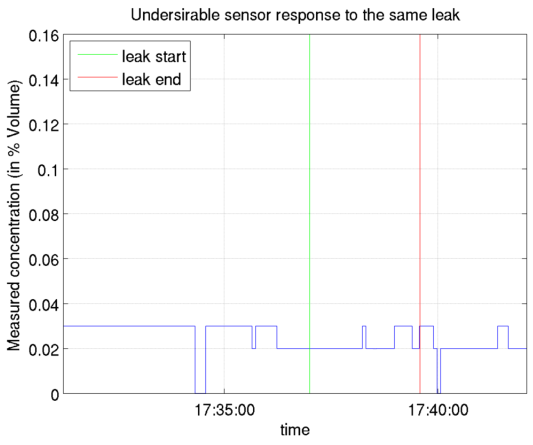
on the grounds and generate alarms whenever a leak is detected. Additionally, periodic reports concerning the concentration gradients of gases on the refinery can be generated and sent to concerned parties.

In terms of sensor placement, these devices would be spread throughout the plant. However, it is beneficial to increase the density in zones susceptible to leaks. For example, an increase in the number of compressors and valves raises the risk of leaks. As such, a typical deployment would have a minimum density necessary for detections, and then certain areas would be characterized with clusters of sensors (increased density) based on the risk. To minimize the overall application energy consumption (for the entire system of sensors), neighboring devices might take turn in "guarding" a zone, then alerting the other devices in case of suspicious increases in measured concentrations.

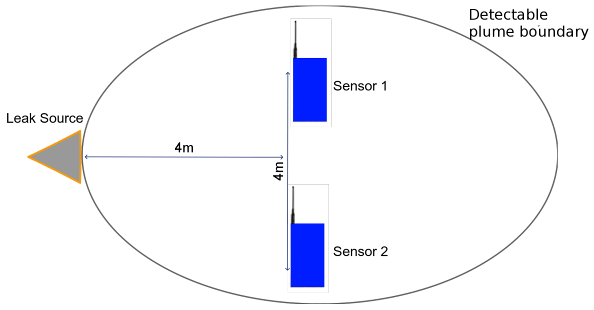
Similar to having an adaptive spatial sampling of concentration, a temporal one would also be beneficial. This means that the mote can decide to increase its sampling frequency when it detects a sudden surge in gas concentration. The reporting rate of data to the



(a) Sensor 1 response: even with a high noise floor, this sensor was responsive to the leak.



(b) Sensor 2 response: elevated noise and no apparent response to the same leak.



(c) Experimental setup diagram showing both sensors within the plume boundary during the leak.

Figure 3.3: Sensor SNR and Response challenges: in this experiment, two sensors were positioned 4m apart, and 4m away from the source of the leak. These sensors were observed with a FLIR Camera and validated to be present in a detectable plume.

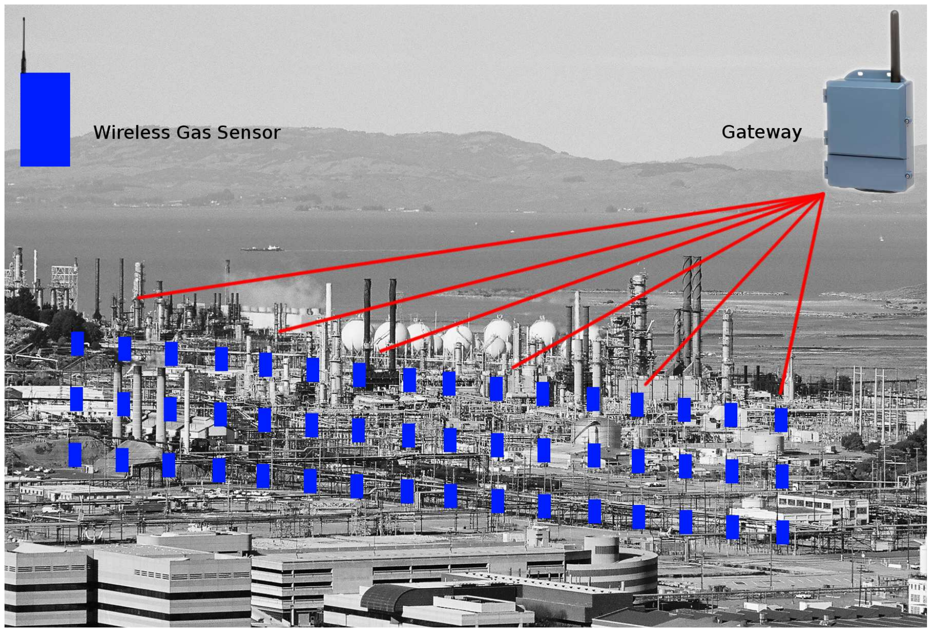


Figure 3.4: Proposed system architecture: gas leak detection sensors are deployed extensively accross a sensitive industrial area (a refinery in this case); data travels through the mesh network towards a single collection point (gateway) where the detection and localization algorithms are applied. The sensors can be duty-cycled spatially and temporally based on the measured concentrations.

gateway, which is not very frequent regularly (on the order of one average reading every 10 minutes for example) can also be increased when the device records an unusual concentration of gas. Augmenting this method with a statistical routine can also get rid of many unnecessary alarms and minimize energy consumption.

3.2.3 Detection Algorithm

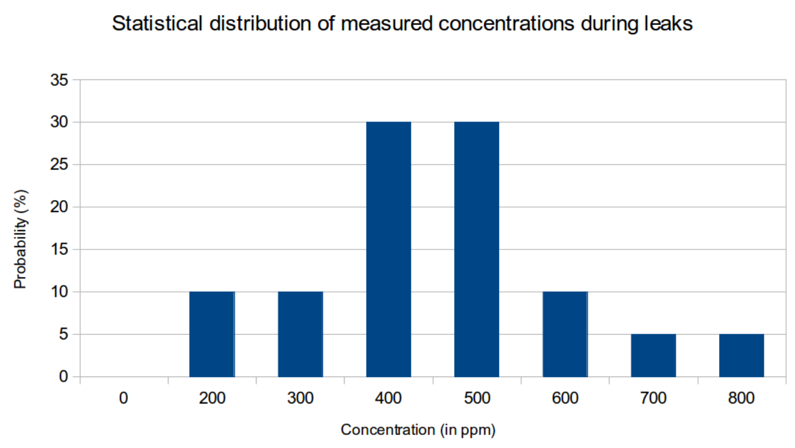
Considering the system architecture above, where each sensor adaptively reports its gas concentration measurements to one location (through one or more gateways), we now look at a method for detecting the occurrences of leaks. Our framework is a probabilistic one, where each sensory observation is represented probabilistically, then the total likelihood of a leak is computed at every time step. To get there, we model each sensor observation independently as $p(s_t(i) \mid \theta_t)$ where $i \in \{1, \ldots, N\}$, N is the total number of sensors per area under consideration, and θ_t is an indicator variable which represents the existence of a leak at time step t. We have derived semiheuristic models for our sensors both in the presence of a leak (ON) and when just measuring leak-free environments (OFF). The measurements and experiments leading to these models will be defined in section 3.3. The histograms corresponding to the derived models are illustrated in Figure 3.5.

Using these models, the likelihood at each time step is computed as follows.

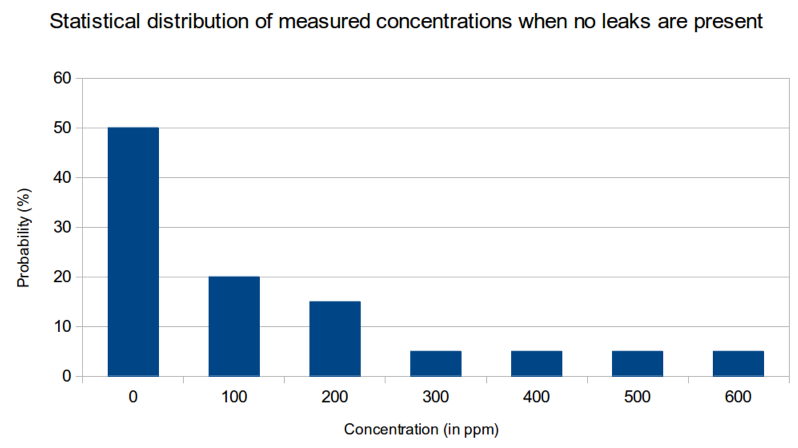
$$L_t(\theta) = \prod_{i=1}^{N} p(s_t(i) \mid \theta) \tag{3.1}$$

When $\theta = 1$, we are essentially computing the likelihood of a leak being present (ON state). $L_t(\theta = 0)$ is the likelihood in the OFF state, or when no leaks are present. This is done at each time step, using all of the available sensory information $s_t(i)$ for $i \in \{1, \ldots, N\}$. Intuitively, one would expect $L_t(\theta = 1)$ to increase in the presence of a leak (i.e. ON state), while expecting $L_t(\theta = 0)$ to decrease during the same time. Such a behavior is observed in our experiments and figure 3.6 is one sample case.

In the quest for a completely automatic leak detection system, we would like to record these changes in the likelihood signal autonomously. We therefore treat the problem as a signal segmentation one [38]. In essence, we are interested in the segments of the likelihood timeseries which differ from the normal. Such signal segmentation techniques have been successfully applied in ECG (electrocardiogram) and EEG (electroencephalogram) applications [39, 40]. We are using the algorithm presented in [39] to detect changes in our constructed likelihood signal $L_t(\theta)$, in both states. The method is detailed in algorithm 2. The main intuition behind this is the following: disturbance free segments of a signal should have *similar* Auto-Correlation Functions (ACFs). However, if there is a clear change in the signal, there should be a corresponding *dissimilarity* in the ACFs as well. Computing the pairwise cosine similarities between each ACF one can visualize the similarity between the segments. By summing the columns of the resultant similarity matrix, the weights are computed. In our application, when there are no leaks, these weights are high. During a leak however, we expect the weights to decrease as they will be *dissimilar* to the regular



(a) Sensor model for concentration measurments during a leak (ON Model).



(b) Sensor model for concentration measurments during a leak (OFF Model).

Figure 3.5: Semiheuristic sensor model derived from experimental data. Non-complementary probabilities accompany each state: ON (leak occurring) or OFF (no leak). This model was obtained by observing various sensor behaviors during leaks and in their absence (similar to the one shown in figure 3.3). The concentration counts were then performed and adjusted heuristically as shown in these histograms. As will be apparent, our detection method is based on the variations of probabilities in a particular period of time.

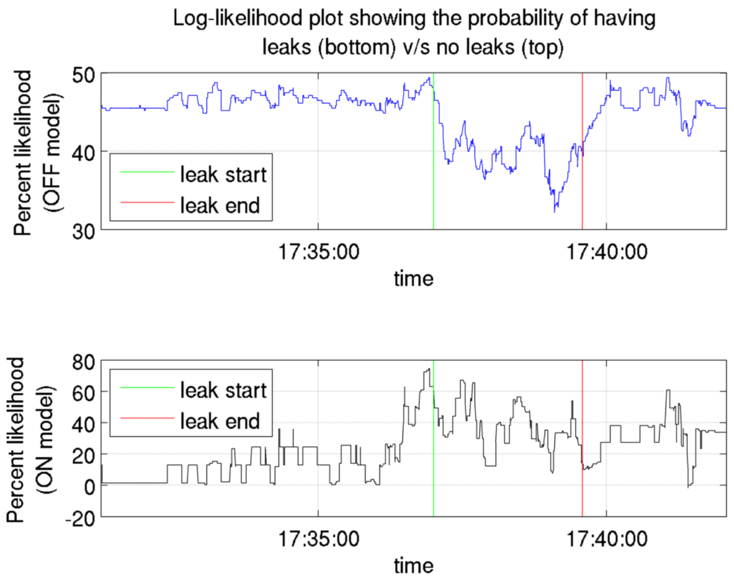


Figure 3.6: Observation likelihood at each time step, for both states: as concentration measurements are received, the probability of having a leak v/s no leak are computed. In the event of a gas release, one would expect to see the likelihood of having no leaks drop, while that of having leaks increase.

"no-leak" segments. The change in the segments is then detected simply by applying a threshold set at the P-th percentile value.

Algorithm 2 Phase-1 change detection algorithm

```
1: Input: likelihood timeseries, window size N, percentile value P
2: Output: indices of the segments corresponding to the irregularities
3: Segment the signal into epochs of length N
4: Compute the Auto-Correlation Functions (ACFs) of each segment, $A_i$
5: Compute the cosine similarity between each pair of ACFs
       $$\cos\theta_{ij} = \frac{A_i^T A_j}{\|A_i\| \, \|A_j\|} \quad (3.2)$$
   and form the similarity matrix
6: Compute the weights by summing the columns of the similarity matrix
7: Calculate the $P$-th percentile of the weights
8: Label the segments with weights outside the $P$-th percentile
```

The algorithm presented here then returns a number of time indices which correspond to the various regions of the likelihood series, which were unusual. We will name these time indices as stage-1 detections. Figure 3.7 shows the same likelihood plots of figure 3.6, but with overlayed segments representing the stage-1 detections.

One of the important parameters of this algorithm is the percentile parameter. Depending on the percentile threshold which is set, one can reduce the rate of false positives. These false positives arise from the fact that the sensor noise floors are elevated (as explained in part A), and therefore reducing our signal-to-noise ratio. Increasing the percentile threshold in our algorithm reduces the false alarm rate, but this however comes at the price of an reduction in true positives. This parameter should be therefore set in such a way that *alarm fatigue* and a "tolerable" leak miss rate are balanced.

Another effect of the sensor noise is the fact that some of the stage-1 detections actually correspond to fluctuations of concentration measurements triggered by temperature variations around the sensing element. This explanation was obtained from the sensor man-

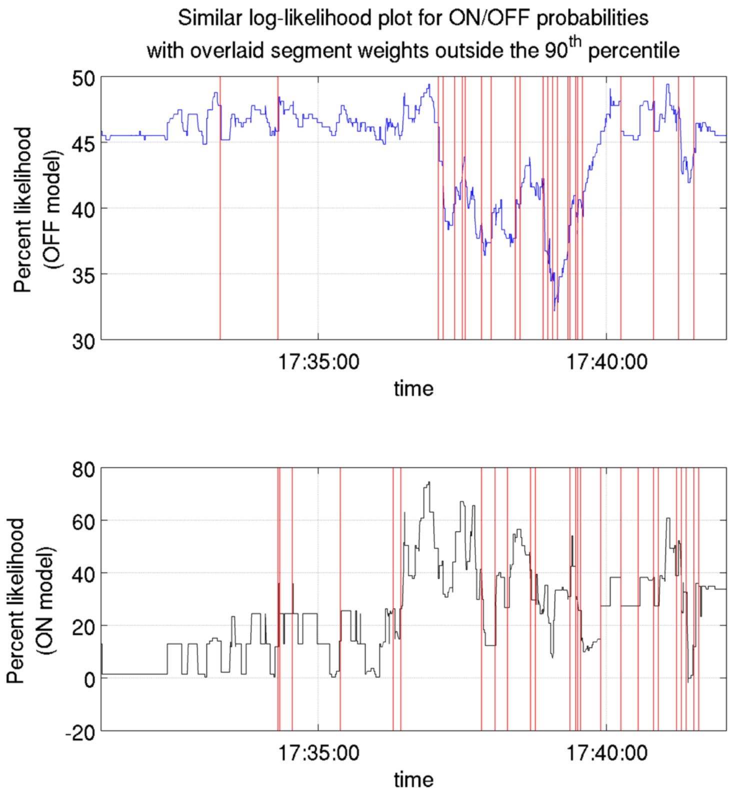


Figure 3.7: The algorithm described here takes the likelihood timeseries as input and returns the instants in time where that plot went beyond the norm (by a certain percentile threshold).

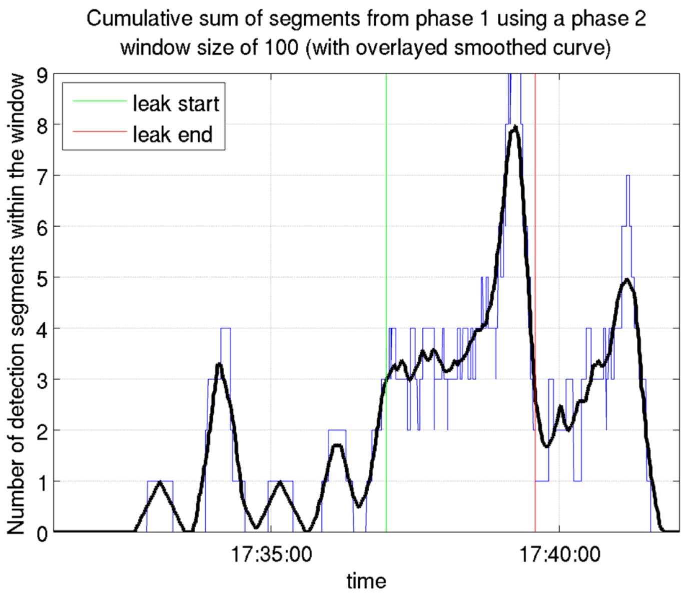


Figure 3.8: Summing the detections in stage-2 allow for an easier identification of leaks. Depending on the window size, the amount of detections can cross a detection threshold for a considerable time. Shown here is the summation plot, overlayed with a smoothed version for clarity.

ufacturer. This effect means that we are not able to immediately consider the output of the first stage and have to perform additional steps before making a decision. To reduce *false positives*, we look at the total number of stage-1 detections in a sliding window. If this cumulative count of stage-1 detections exceeds some pre-determined threshold level for a preset period of time, we then output a detection, which we name as a stage-2 detection. The length of the sliding window, the threshold level, and the duration of crossing, can all be used to control the false positive and true positive rates, similarly to stage-1. The overall performance of stage-2 detections (and therefore that of the entire algorithm) is analyzed with respect to user-set parameters (percentile threshold, stage-1 window size, and stage-2 window size). The results are presented in section 3.3.

3.2.4 Localization Algorithm

Following a successful detection of a leak by both stages of the algorithm, a localization routine is called. We utilized a simple center of mass approach. Upon finding a detection, we calculate the 2-dimensional mean of the concentration measurements in the in the X-Y plane.

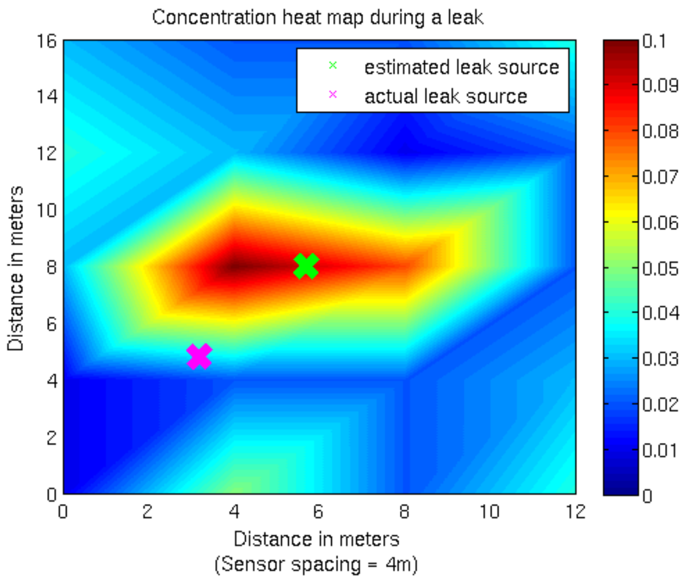
$$\hat{x} = \frac{\sum_{i=1}^{N} s_t(i) x_i}{\sum_{i=1}^{N} s_t(i)} \tag{3.3}$$

$$\hat{y} = \frac{\sum_{i=1}^{N} s_t(i) y_i}{\sum_{i=1}^{N} s_t(i)} \tag{3.4}$$

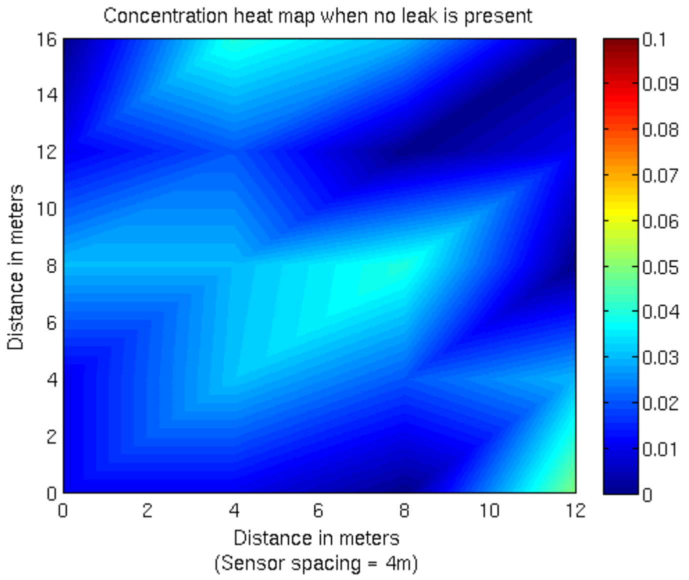
In the above equations, x_i and y_i represent the coordinates of each sensor, while $s_t(i)$ is the sensor concentration reading. The resulting point $(\hat{x}, \hat{y})$ is defined as the estimate of the leak source detection. This would correspond to finding the point of maximum concentration on a heat map. Figure 3.9 depicts the localization result on a particular heat map.

3.2.5 Possible Improvements to our Method

In our analysis, we assumed that the sensory readings are conditioned only on the state of the leak (i.e. ON vs OFF). Now, as the deployment grows in size, a far away sensor



(a) Concentration heat map during a leak. The sidebar denotes concentrations in % volume.



(b) Baseline concentration heat map (no gas is present). The sidebar denotes concentrations in % volume.

Figure 3.9: Concentration heat maps of a 5x4 grid of sensors in two states: during a leak and when no gas is present.

from the leak source will probably not be able to detect any change. Still, considering a reduced spatial sampling, the response of the sensor to the plume of gas will depend on its location with respect to this plume. To take this into account, the sensor models would be augmented in the following form.

$$p(s_t(i) \mid \theta, Location) \tag{3.5}$$

However, developing sensor models based on plumes is well beyond the scope of this work, and it would somehow complicate the study. Nevertheless, it is worth mentioning that a location-dependent model can be worked out similarly to the previous section. The likelihood methods will then be directly applied, with some tuning.

One other important improvement that can be introduced is time dependence. With our current approach, we do not exploit the fact that if a leak exists at time t, it is then likely for this leak to remain at time $t+1$. Considering this feature means that we could use a general state space model to describe the leak phenomenon. Our objective would then be to perform state estimation, for which one may resort to Kalman filtering, Unscented Kalman Filtering or Particle Filtering, etc.

3.3 Experimental Validation

To validate our architecture, hardware and algorithms, we took part in an experiment at the Texas A&M Engineering Extension Service facility, in College Station, TX. Over the period of three days, more than 60 propane leaks of two minutes each were released. These leaks were monitored by 3 different detection systems (including the one presented here), and controlled by a team of engineers and a team of firefighters. The site of the releases is shown in figure 3.10. Twenty wireless propane sensors were used to monitor an area of about $200m^2$ surrounding the two release points (at $0.5m$ and $5.5m$). The sensors were placed in a 4×5 grid configuration, with a separation of about $4m$. All of them were mounted on an elevation of about $2.25m$. These devices measured propane concentrations at a rate of 1 measurement every 5 seconds. The measurements were collected in a data packet and transmitted to a nearby gateway. We now present the results of our algorithm

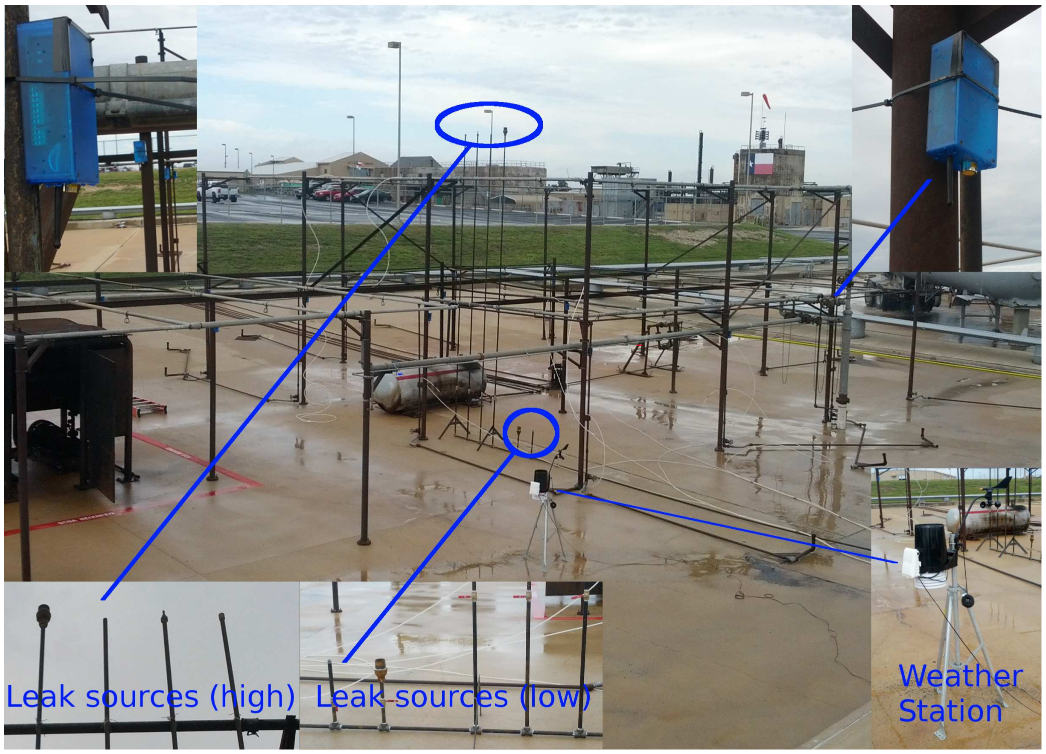


Figure 3.10: Site of the experiment in College Station, TX. This figure shows the two release points, and the placement of the 20 sensor grid (5x4) at an elevation of about 2.25m.

applied to the 60 releases of different source heights, source nozzle sizes ($2mm$, $6.35mm$, $19mm$, and $63.5mm$), and flow rates (ranging between $1.35lb/hr$ and $1020lb/hr$).

3.3.1 Detection Results

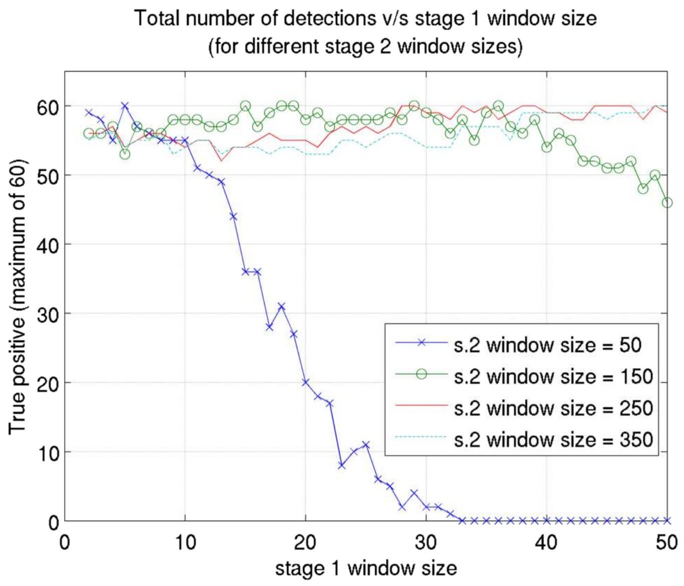
The algorithms described in section 3.2 were applied to the collected concetration measurements, with different parameters modified for every pass. First we look at the number of correct detections and false alarms as the stage-1 window size is varied. The results are shown in figure 3.11. The general trend observed shows that increasing the window size of the first stage does not affect the number of detections greatly, except when it grows to a point where the actual variations associated with the leaks are no longer beyond the selected percentile threshold. This effect is accelerated when the window size of stage-2 is reduced. The effect in question is readily visible in figure 3.11 (a) for the stage-2 window size of 50. Looking at false positives, we notice that the trend peaks at a particular stage-1 window size before starting to roll off. However, it is to note that this roll-off is sometimes associated with a reduction in detections as well.

The impact of changing the stage-2 window size on the number of true and false positives is shown in figure 3.12. Concerning the number of detections, increasing the size of the stage-2 window shows an increase in these detections, which tends to settle (with a slight dip) beyond a particular point (around 125 samples). At the same time, the number of false alarms increases sharply before it peaks and decreases with an increasing window. This validates the conjecture made before: increasing the stage-2 window size allows us to have a better detection methodology, as it decreases the number of false alarms, while not diminishing the true positives by much.

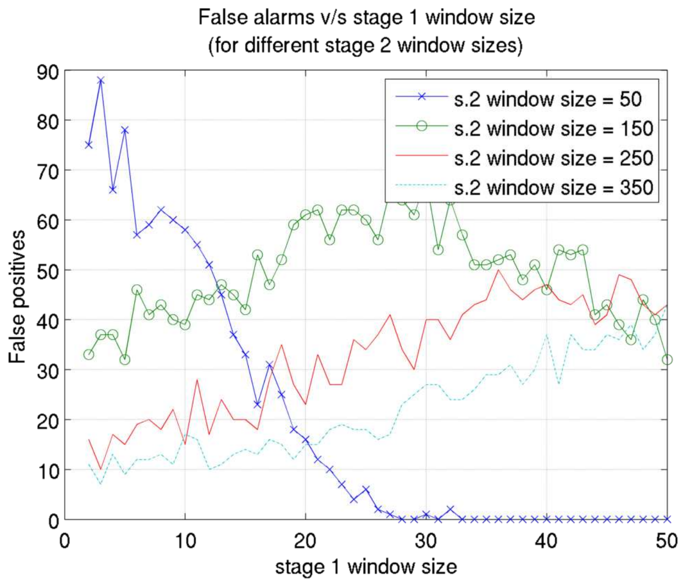
However, and as expected, increasing the stage-2 window size has a direct effect on delays. This is readily observed in figure 3.13. Though the delay starts by being elevated with short window sizes, this can be explained by the fact that the number of leaks detected during with that configuration is very low, and the data is therefore not representative. However, once the delay reaches a minimum value, it starts to climb back up with increased stage-2 window size. Indeed, the confidence in the detection increases, but the consequence is that the algorithm has to wait for more and more samples before making the decision.

As explained before, the percentile parameter is an important one that determines the performance of our detection method. Looking at figure 3.14, we notice that increasing this parameter has a similar effect to increasing the stage-2 window size: the number of detections rises quickly before starting to dip slightly with increase percentile threshold, while the number of false alarms rises quickly in the beginning, before dropping as the percentile is increased.

At this point, it is worth noting that the most balanced configuration of this algorithm returned a detection rate of 55/60, with 7 false alarms, and an average delay of 108 seconds. Though these results seem promising, they leave a lot of room for improvement. The

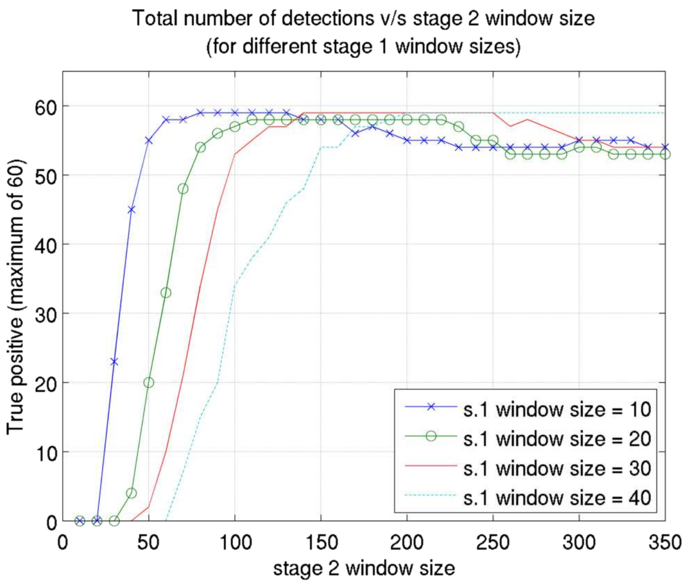


(a) An increase in the window size generally leads to a decrease in the number of detections.

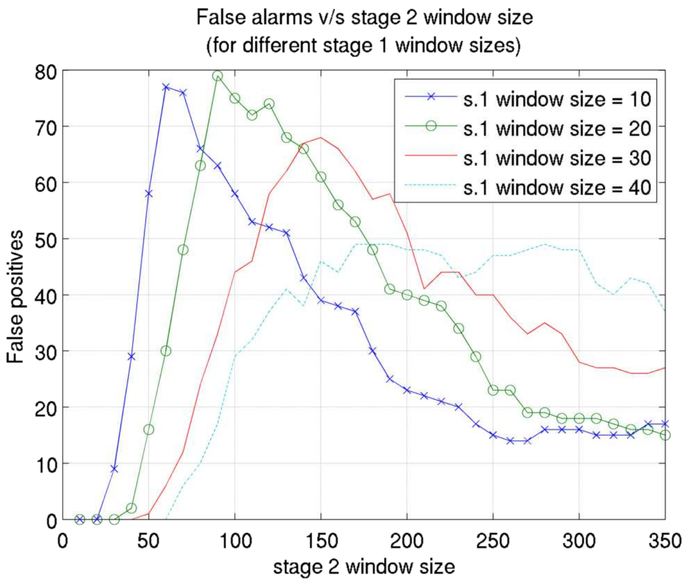


(b) False alarms also increase with the window size.

Figure 3.11: Impact of varying the stage-1 window size on the number of detections and the false alarms.



(a) The increase in window size beyond a certain point reduces the number of detections by a small number.



(b) False alarms are reduced the larger the stage-2 window size is.

Figure 3.12: At the second stage, a larger window size tends to have a better performance overall.

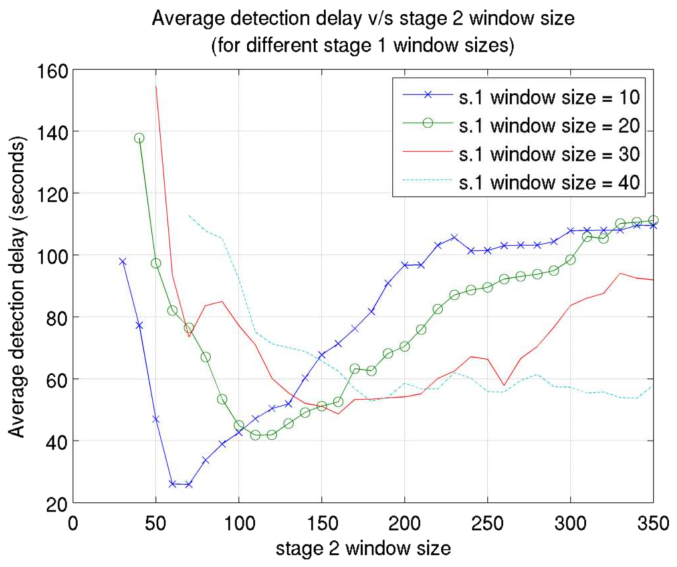
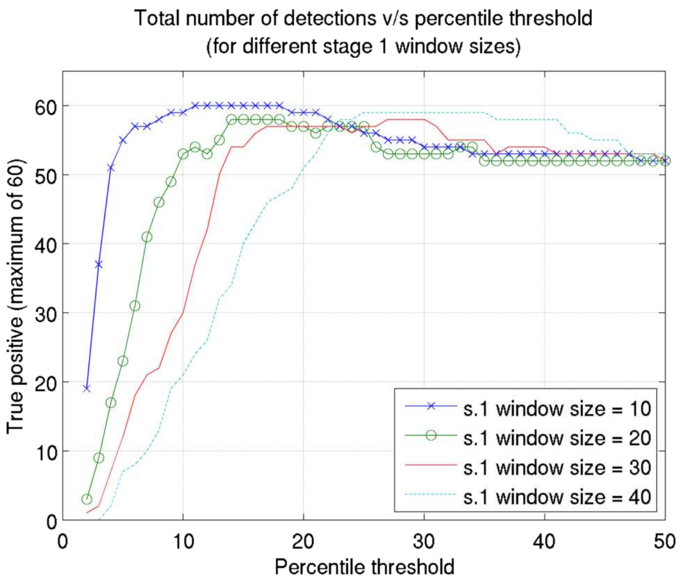


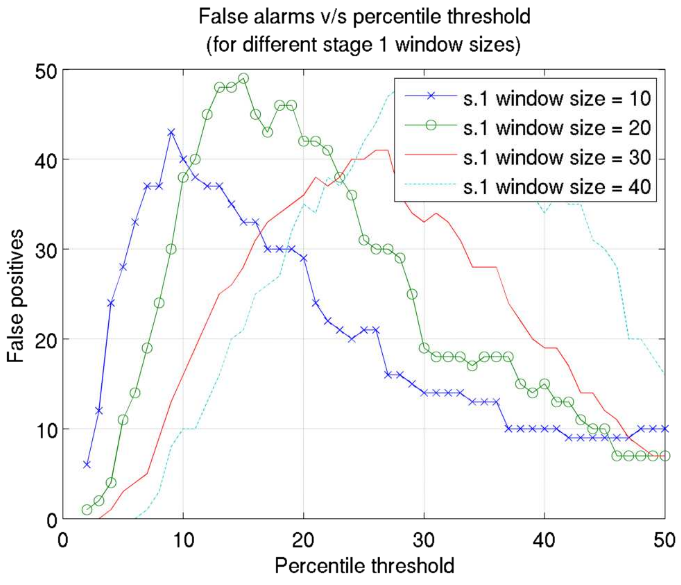
Figure 3.13: The stage-2 window size increase leads to a decrease in false alarms, but that comes at the expense of an increase in detection delay.

detection rate is satisfactory, especially when compared to the absence of widespread reliable detection methods on the market today, but a 100% rate would be desirable of course. A delay of more than 100s, could represent some challenges for a refinery workers in responding to an alarm, so reducing it to below 1 minute is also desired. Finally, a false alarm rate of 7 over a period of three days seems excessive, even if these alarms were short-lived. Still, a false positive rate of 1 per plant per month (or per year) would be highly desirable, especially with an elevated detection rate to accompany it. To reach these desirable results, we would require an improvement in sensor technology of at least one order of magnitude in SNR. Concerning the lifetime of the sensors, and to enable a 10-year deployment without the need to change batteries, a reduction of power consumption of two orders of magnitude is required for reliable detection.

With this in mind, we “massage” our experimental data to reduce its SNR as proposed and run the algorithm again. The results are shown in figure 3.15. It is clear that certain configurations of our detection methodology would give us a 100% detection rate (while others give a slightly reduced rate). Most importantly however, is the false alarm rate, which stays at zero across all configurations considered here.

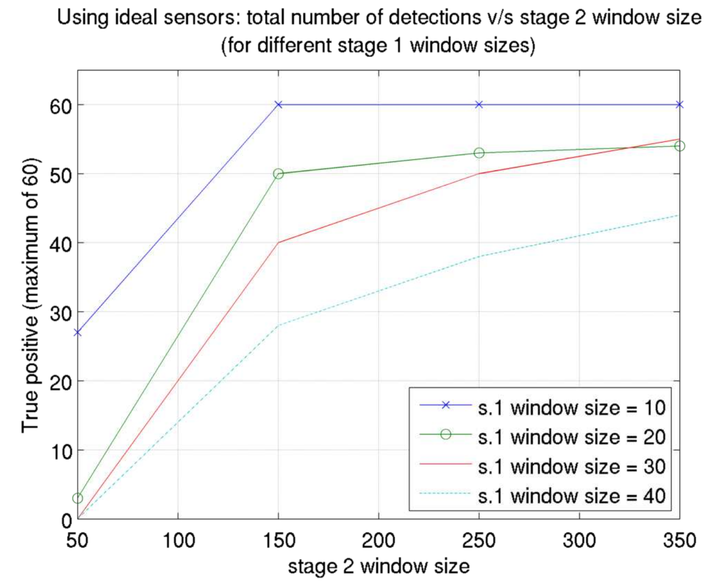


(a) A percentile threshold in the 10-20% seems to lead to a higher number of detections.

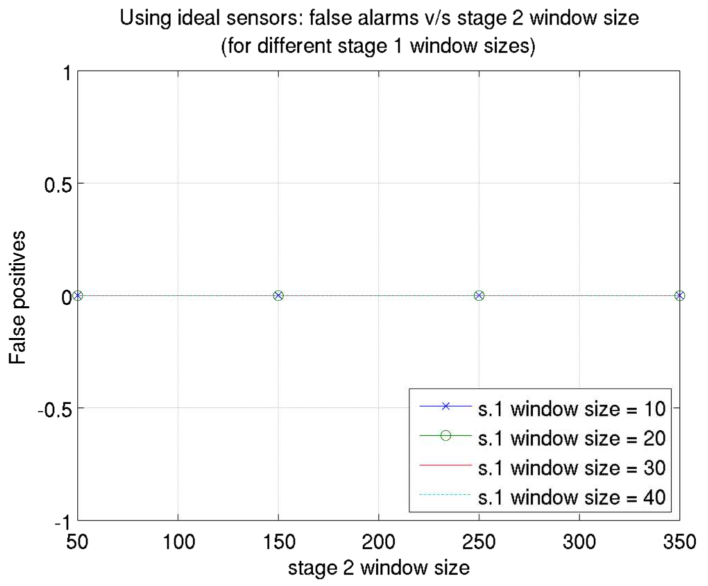


(b) The number of false alarms rises quickly with the percentile, before it rolls off at a slower rate.

Figure 3.14: The percentile threshold parameter is very important in determining the performance of these routines.



(a) In some configurations, detecting all of the releases becomes possible.



(b) The rate of false alarms is zero across most configurations.

Figure 3.15: Given ideal sensors (with high SNR), the detection problem would become much easier. This plot shows the result of the two stage detection algorithm presented here applied to "ideal" data generated from the experimental one.

3.3.2 Localization Results

We now look at the localization results given our noisy sensors covering the test site. Upon finding a detection, the center of mass routine is called upon the concentration data, and the point of highest mean in both the X and Y directions is identified as the leak source estimate. Figure 3.16 shows a scatter of these detection estimates (for the different parameters described above) along with the true source of the leaks. Most of the detections seem clustered in the middle of the sensor grid, but with a slight skew towards the actual source. One potential explanation to this phenomenon is the absence of sensors directly above the leak release point. This means that the algorithm is converging towards the closest sensor(s) in the vicinity of this source.

Looking at the distribution of distance between the estimated leak source and the actual one in figure 3.17, we can see that it follows an inverse Gaussian trend. Most detections actually occurred less than three meters away from the leak source, and a striking majority were localized less than 5 meters away. Also shown in figure 3.18, are the localizations of the 55 “best” detections which were obtained as described above (with the 7 false alarms and the delay of 108 seconds). The results shown confirm the reliability of the localization method, with no estimates found in unusual locations (near the edge of the network for example). In a realistic deployment, the localization figures presented here figures would be very helpful for workers who are familiar with the equipment present in the vicinity of the sensors, and who should quickly be able to identify the source of the leak.

3.4 Final Thoughts

Though we can achieve the results presented here today, there is still some work to be done to get this idea into production. Essentially, as the wireless communication reaches new boundaries in reliability, and as efficient microcontrollers become cheaper and less power hungry, the only component left to be improved is the sensor. Certainly, improvements on the detection algorithm could help reduce the false alarm rate and increase the

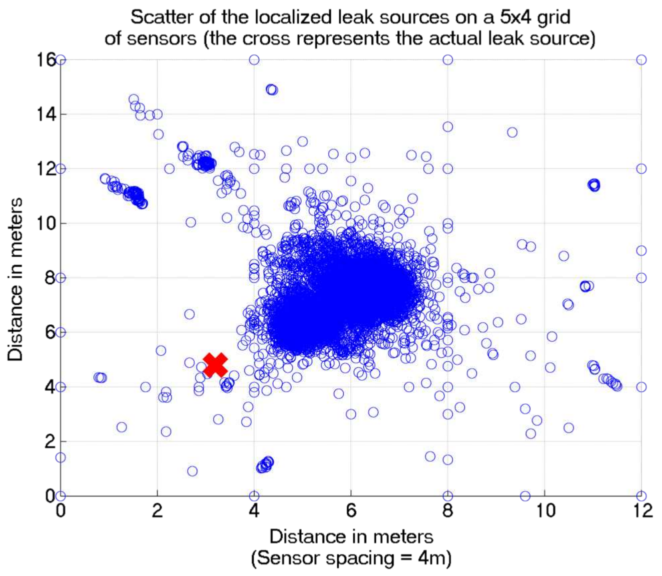


Figure 3.16: Scatter plot of all of the detections (during many different configurations). The offset seen between the real source of the leak and the conglomeration of detections could be explained by the fact that no sensors were present directly above the source.

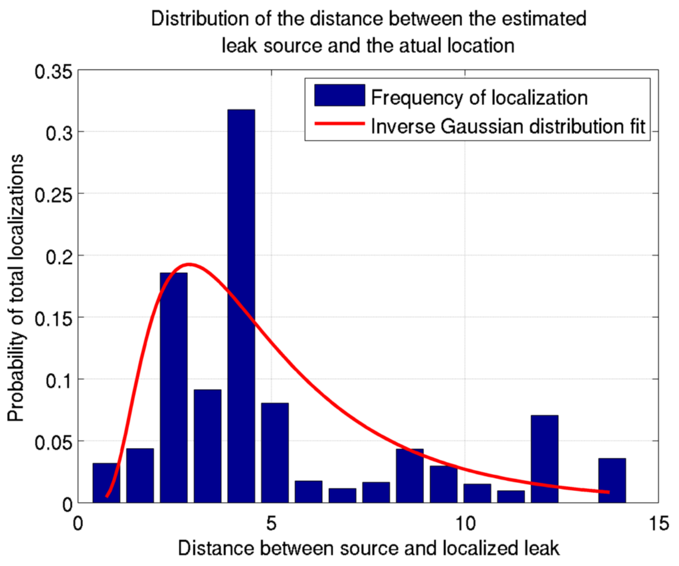


Figure 3.17: A histogram of the distances between the detections and the real source is leak is shown in this figure. More than 50% of all detections are within 3 meters of the source.

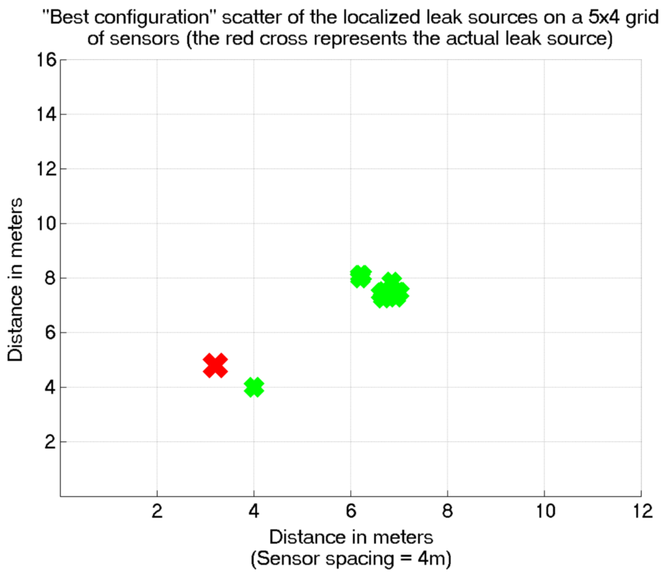


Figure 3.18: In the preferred configuration (resulting in 55 detections out of 60, with 7 false alarms), the localization results appear even closer to the actual source of the leaks.

detection rate, but the sensing hardware would be a better place to start. An order of magnitude improvement in signal-to-noise needs to be accomplished in explosive gas sensors (to reduce false alarms). A quicker wake-up time and two orders of magnitude of improvement in energy consumption are needed to extend the lifetime of the device to 5 years. As a concluding note, and though this study was done with industrial plants in mind, we feel that similar approaches can be accomplished in cities of the future, where a gas detection and localization system can help address the problems of leaks in urban gas pipelines.

Part II:

Tools

Chapter 4

A Realistic Energy Consumption Model for TSCH Networks[1]

Time Slotted Channel Hopping (TSCH) mesh networks are becoming central for wireless industrial deployments as they are able to achieve 99.999% reliability [41] with minimal power consumption. Standards such as WirelessHART [1], ISA100.11a [2] and IEEE802.15.4e [3] are rooted in the TSCH medium access technique. In a TSCH network, nodes are synchronized, and time is split into time slots, each typically $10ms$ long. Time slots are grouped into a slotframe which continuously repeats over time.

The network's communication schedule instructs each node what to do in each time slot: send to a particular neighbor, receive from a particular neighbor, or sleep [42, 43]. Channel diversity is obtained by specifying, for each send and receive slot, a channel offset. The same channel offset is translated into a different frequency on which to communicate at each iteration of the slotframe. The resulting channel hopping communication reduces the impact of external interference and multipath fading, thereby increasing the reliability of the network [44].

[1]**This chapter was published in a similar form as follows:**
Vilajosana, X.; Wang, Q.; Chraim, F.; Watteyne, T.; Chang, T.; Pister, K.S.J., "A Realistic Energy Consumption Model for TSCH Networks," Sensors Journal, IEEE, vol.14, no.2, pp.482,489, Feb. 2014 doi: 10.1109/JSEN.2013.2285411

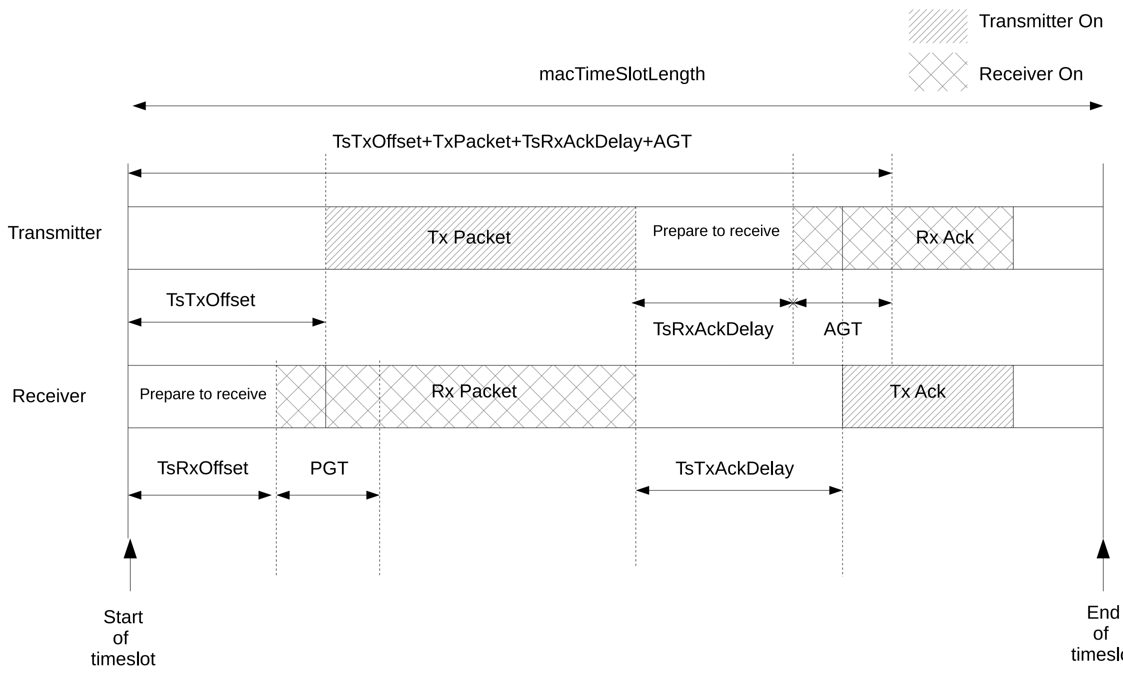


Figure 4.1: Timeslot template for a transmitter (top) and receiver node (bottom). This figure shows the timing breakdown for these two types of slot, highlighting where transmission occurs [3]

All nodes in the TSCH network are synchronized. Because communication occurs at a well-defined times within a time slot, the sender nodes know exactly when to transmit. If the sender and receiver nodes were perfectly synchronized, the receiver node would turn its radio on at exactly the instant when the transmitter starts emitting. The sender's and receiver's radio would be on only for the duration of the packet being transmitted. After the transmission of a packet, both nodes can switch their radios off to save energy or sleep a few milliseconds before repeating the same process in order to receive/transmit an acknowledgment (ACK). This simplistic scenario is the optimal solution in terms of energy consumption that can be achieved in a communication between two nodes, since it minimizes the time during which the transceivers are on.

However, as clocks between neighbor nodes in a network drift ($30ppm$ relative drift is a typical value [45]), a small "guard time" is required at the receiver end to account for relative desynchronization [46]. Acknowledgments follow a similar scheme: a guard time is introduced around the ideal reception moment.

Although the wireless medium is lossy in nature, TSCH networks are deterministic in their scheduling. The energy consumed by a node can be modeled precisely, by profiling the actions that are carried out during each slot. The aim of this chapter is to present an energy consumption model for TSCH networks, and to analyze the impact of control signaling and the communication schedule on this energy consumption.

Kohvakka et. al. [47] model the energy consumption of the legacy IEEE802.15.4-2006 MAC protocol operating in slotted (beaconed) mode. A similar model is later used to predict the energy consumption per received bit as a function of traffic load and packet size [48]. Wang et. al. [49] present a general energy consumption model for WSN devices based on their hardware architecture. This model reflects the energy consumption in various functioning states, and during transitions between states of the devices leading to a very accurate energy modeling. More recently, Casilari et. al. [50] present a model for non-slotted CSMA/CA based IEEE802.15.4/ZigBee networks.

Khader and Willig [51] present an energy consumption model for WirelessHART TDMA networks, which addresses objectives different from the ones presented here. The authors focus on analyzing the WirelessHART protocol and identifying the main aspects that contribute to the energy consumption it induces. They focus is on exploiting different sleep modes on the CC2420 transceiver in order to reduce this consumption. In contrast, our goal is to develop an energy consumption model based on a fine-grained energy analysis of each type of slot in such networks, in the hope of providing a tool for consumption estimation prior to real network deployments.

4.1 Energy Model

As a device joins a TSCH network, it obtains information about the duration of each time slot, and the number of slots in a slotframe. In each slot, the node can either transmit, receive, or keep its radio off. A scheduling entity is responsible for building the schedule which will satisfy the bandwidth and latency needs of the different flows in the network. Throughout the lifetime of the network, this entity modifies this schedule to adapt to changes in the topology and changes of the traffic requirements. The schedule allows for a fine-grained trade-off between latency, bandwidth, redundancy and power consumption. Several scheduling approaches have been proposed, both centralized [52] and distributed [53, 54], and are currently being standardized [55]. The energy consumption of a mote is the sum of the energy consumed in each slot. To build the energy consumption model, we start by investigating the energy consumed in each type of slot, both through analysis and experimentation.

4.1.1 TSCH Slot Template

This chapter focuses on IEEE802.15.4e networks, but the same principle can be applied directly to other TSCH standards such as WirelessHART [1].

In IEEE802.15.4e, there are 6 different types of time slots:

- `TxDataRxAck:` A timeslot during which the node sends some data frame, and receives an acknowledgment (ACK) indicating successful reception.
- `TxData:` Similar to the previous, but no ACK is expected. This is typically used when the data packet is broadcast.
- `RxDataTxAck:` A timeslot during which the node receives some data frame, and sends back an ACK to indicate successful reception.
- `RxData:` Similar to the previous, but no ACK is exchanged.
- `Idle:` Time slot during which a node listens for data, but receives none.
- `Sleep:` Time slot during which the node's radio stays off.

Fig. 4.1 presents a detailed breakdown of the activity of a node in a `TxDataRxAck` slot at the transmitter, and a `RxDataTxAck` slot at the receiver [3]. The transmitter node starts by waiting for *macTsTxOffset*, during which it prepares the data to send, and configures the radio according to the frequency inferred from the schedule. The radio then turns on and transmits the packet exactly *macTsTxOffset* from the beginning of the slot. After the last byte of the packet has left the radio, the transmitter gives the receiver some time to prepare the acknowledgment packet by waiting for *macTsRxAckDelay*. If no acknowledgment is received after the *Acknowledgment Guard Time (AGT)* period, the device turns off the radio and considers the transmission failed.

On the receiver's side, the device waits for *macTsRxOffset*, then turns its radio on, listening for a packet. If after the *Packet Guard Time (PGT)*, no packet is received, the device turns its radio off for the remainder of the slot. If a valid packet is received, the node waits *macTsTxAckDelay* after the reception of the last byte, before turning its radio on and sending the acknowledgment.

4.1.2 Slot Energy Consumption Modeling

In a typical node, the two components which consume the most are the micro-controller and the radio. These components can either be two separate chips interconnected by some digital bus on a board, or grouped in a single System-on-Chip. To accurately model the energy consumption in this node, one must look at a detailed breakdown of the various states each module enters, for each type of slot. As the micro-controller and radio change state, their consumption varies. Table 4.1 shows their state during the different phases of a `TxDataRxAck` slot. The charge (in Coulombs) drawn from a battery during the execution of this slot is the sum of the charge in each of these steps.

$$Q_{TxDataRxAck} = \int_{0}^{TsSlotDuration} I_{TxDataRxAck}(t)dt \tag{4.1}$$

The same analysis holds when computing the energy consumption of the other 5 slot types, and extract $Q_{RxDataTxAck}$, Q_{TxData}, Q_{RxData}, Q_{Idle}, and Q_{Sleep}.

This chapter focuses on two hardware platforms made with commercial off-the-shelf

Period in Template	State of motes	μP state	Radio state
StartOfTimeslot	NewSlot	Active	Sleep
TsTxOffset	TxDataOffset	Active	Sleep
	PostTxDataOffset	Sleep	Sleep
	TxDataPrepare	Active	ToReady
	PostTxDataPrepare	Sleep	Ready
TxPacket	TxDtataStart	Active	ToTx
	TxData	Active	Tx
	PostTxData	Sleep	Tx
TsRxAckDelay	TxRxAckOffset	Active	Sleep
	PostTxRxAckOffset	Sleep	Sleep
AGT	RxAckPrepare	Active	ToListen
	RxAckReady	Sleep	Listen
RxAck	RxAckStart	Active	Rx
	RxAck	Sleep	Rx
	PostRxAck	Active	Sleep
BeforeEnd	Sleep	Sleep	Sleep
EndOfTimeslot	EndSlot	Active	Sleep

Table 4.1: Mapping from periods in template to states of mote modules.

(COTS) components, which run the OpenWSN reference TSCH implementation [22]. These platforms are representative of spectrum of nodes available. OpenMoteSTM is the "high-end" node, which features a STMicroelectronic STM32F103RB 32-bit microcontroller at the Atmel AT86RF231 IEEE802.15.4-compliant radio. GINA [10] is the "low-end" node, with a Texas Instruments MSP430F2618 16-bit microcontroller, and the same Atmel AT86RF231 radio. Table 4.2 and Table 4.3 list the the current draw of those components in the OpenWSN implementation.

4.1.3 Experimental Verification

To verify the validity of the energy model, the energy consumption for three different types of slots is measured on the GINA and OpenMote-STM32 platforms, and compared to the value computed using the model. In the experiments on both GINA and OpemMote-STM32, the same slot timings are used as shown in Table 4.4. Fig. 4.2a shows the current drawn by a GINA mote during an idle slot. At $T_{TsTxoffset} - T_{RxGT}/2 - T_{RxDataPrepare}$, the

GenericMode	AT86RF231 Mode	Current	Measured
Sleep	TRX_OFF	0.4mA	0.49mA
ToReady	TRX_OFF $\Rightarrow PLL\_ON$	5.6mA	N/A
Ready	PLL_ON	5.6mA	5.4mA
Tx	$BUSY\_TX$	11.6mA (0dBm)	13.7mA (0dBm)
ToListen	TRX_OFF $\Rightarrow PLL\_ON$ $\Rightarrow RX\_ON$	12.3mA	N/A
Listen	RX_ON	12.3mA	11.6mA
Rx	RX_ON	12.3mA	11.6mA

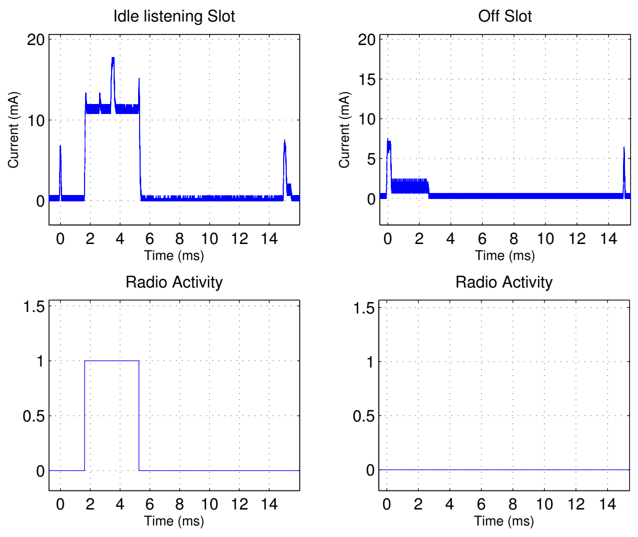
Table 4.2: Current drawn by the Atmel AT86RF231 radio chip for different states (theoretical and measured).

Generic Mode	MSP430 DS	MSP430 Exp	STM32 DS	STM32 Exp
Active	7.4mA@16MHz 3.7mA@8MHz	7.54mA@16Mhz N/A	27mA@72MHz 7.4mA@16MHz 4.6mA@8MHz	32mA@72MHz N/A N/A
Sleep	$1.1\mu A$	N/A	$14\mu A$	N/A

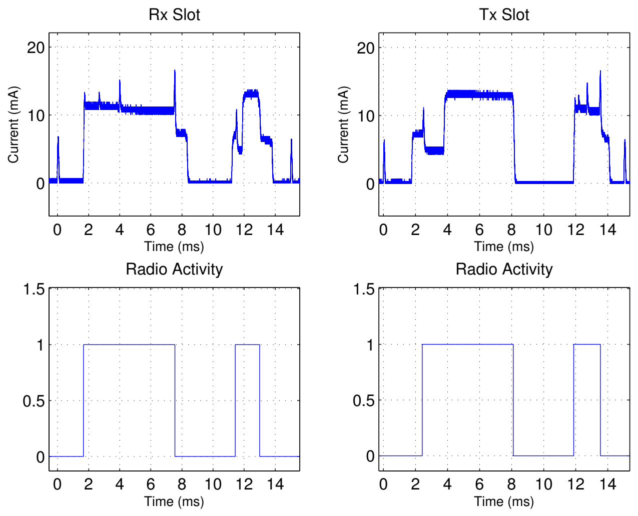
Table 4.3: Current drawn by the MSP430 and STM32 microcontrollers for different states (theoretical as defined by the DataSheet (DS) and measured experimentally (Exp)).

Timing Constant	**Value**
TsSlotDuration	$15000\mu s$
TsTxOffset	$4000\mu s$
TsRxOffset	$305\mu s$
PGT	$2600\mu s$
AGT	$1000\mu s$
TsTxAckDelay	$4000\mu s$
TsRxAckDelay	$TsTxAckDelay - AGT/2\ \mu s$

Table 4.4: Slot timing, as implemented in OpenWSN.



(a) Idle listen and off slots.



(b) Transmission and reception slots.

Figure 4.2: Measured current draw on a GINA mote.

	Measured		Simulated	
State	GINA	OM-STM32	GINA	OM-STM32
Idle	47.9	101.1	54.1	85.2
Sleep	4.9	37.8	8.2	9.2
TxDataRxAck	92.6	161.9	103.3	151.2
TxData	69.6	119.2	76.7	123.1
RxDataTxAck	96.3	217.0	105.2	175.9
RxData	72.1	154.8	78.0	125.0

Table 4.5: Measured and Simulated charge drawn for each types of slot, in μC.

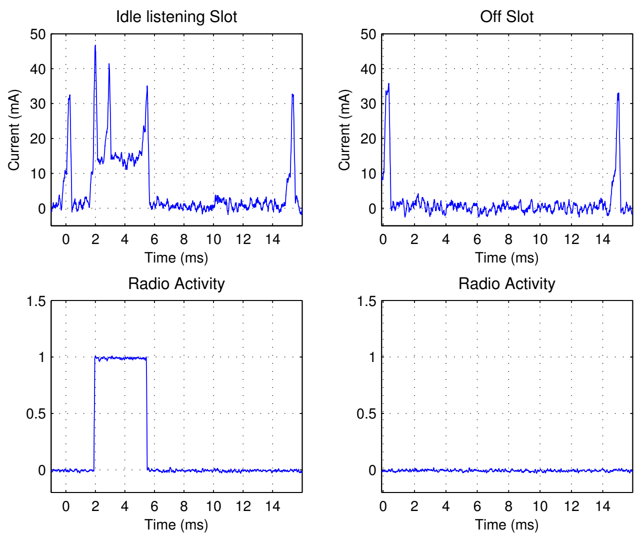
mote's microcontroller turns on to prepare the radio and turns it on at $T_{TsTxoffset} - T_{RxGT}/2$ for reception. After T_{RxGT} the radio is switched off as no packet is received.

Fig. 4.2b shows the current drawn by a GINA mote during a transmission slot. The micro-controller is turned on at $T_{TsTxoffset} - T_{TxDataPrepare}$, allowing enough time to prepare the data. At that time, the radio is turned on and the packet is loaded into the radio. The first current spike in the figure at $0.047ms$ is measured when the radio is being turned on. The bytes of the packet are then loaded into the radio's transmit buffer. At $T_{TsTxoffset}$, the packet is sent. Once the radio is done transmitting, the radio is switched off for the $T_{TsRxAckDelay}$ period. A little before the ACK is expected, the radio is turned on again to listen. After the reception of the ACK, the radio is switched off.

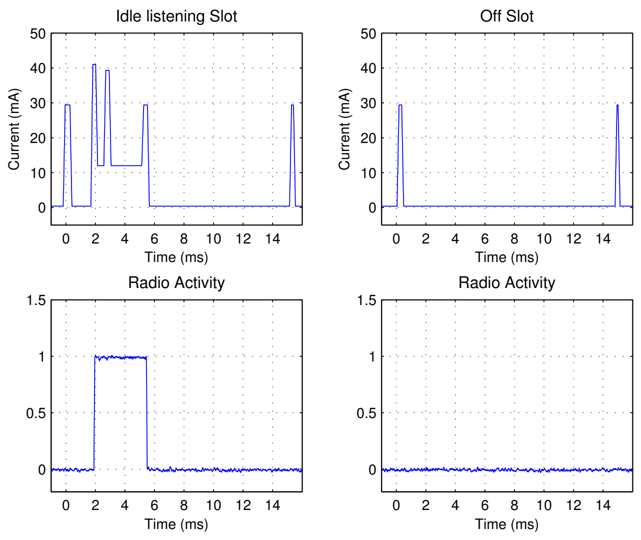
Similarly, Fig. 4.2b presents the current drawn by a GINA mote during a reception slot. The mote sleeps until $T_{TsTxoffset} - T_{RxGT} - T_{RxDataPrepare}$. The micro-controller then switches on to configure the radio to the right frequency. At $T_{TsTxoffset} - T_{RxGT}/2$, the radio starts listening. After listening and receiving for a total of $4.6ms$, the packet is completely received and the micro-controller and radio are turned off during the $T_{TsTxAckDelay} - T_{TxAckPrepare}$ period. The ACK is then loaded into the radio and transmitted at $T_{TsTxAckDelay}$, after which the micro-controller and radio are switched off.

The energy consumption for the three remaining types of slots is presented in Figs. 4.3a and 4.4a.

These experimental results are compared to the results calculated by implementing the energy model in Matlab. The values corresponding to GINA and OpenMote-STM32

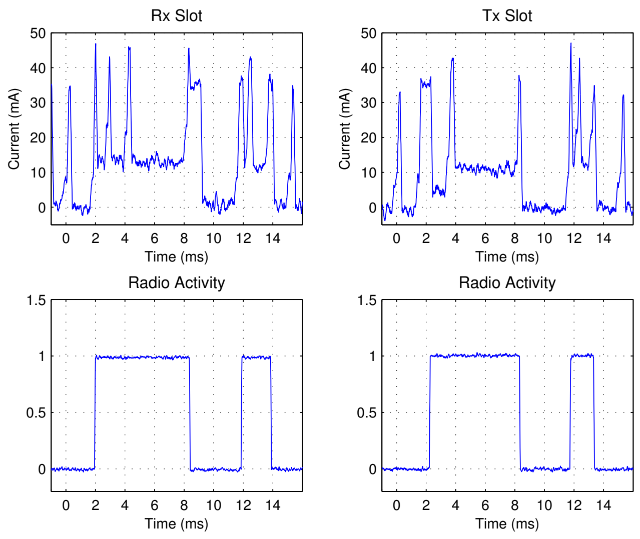


(a) Measured.

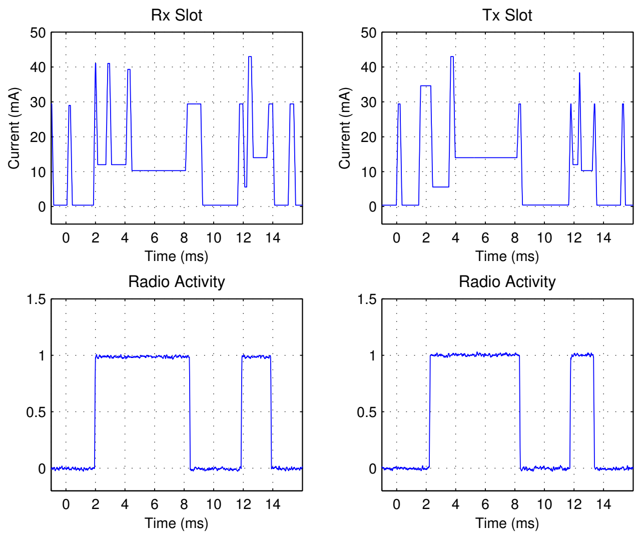


(b) Computed using the model presented in this chapter.

Figure 4.3: Current draw during idle listening and off slots, on an OpenMote-STM32.



(a) Measured.



(b) Computed using the model presented in this chapter.

Figure 4.4: Current draw during transmission and reception slots, on an OpenMote-STM32.

have been selected from Table 4.2 and Table 4.3, and introduced in the calculation. The calculated results are shown in Figs. 4.3b and 4.4b. The comparison between measured and calculated values shows a good match between the model and the experimental values. In addition, Table 4.5 shows the charge (in μC) drawn by the device, both by experiment and through model-based calculation. The difference between the measurement and the calculation comes from multiple sources, e.g. the measurement error (equipment imprecision when measuring currents of the order of μA), little differences on timing within the slot and the difference between typical current consumption values reported by the manufacturer as compared to the ones actually consumed during execution (see Table 4.2 and Table 4.3).

4.1.4 Slot-Frame Energy Consumption Modeling

We can use the energy model of the different slots to determine how much energy is used during each slotframe, and compute the expected battery lifetime of the network. Equations (4.2) through (4.7) define how much energy is consumed in a slotframe for the different types of slots. Subscripts a refers to available slots while subscript u refers to used slots. Consider PDR as Packet Delivery Ratio defined as the number of packets being acknowledged divided by the number of packets being sent by a node. In the following equations, N stands for *number of* and the subscripts define the type of slot being considered (e.g N_{aRxTx} stands for *the number of available slots where the node will be listening for a packet and will send and acknowledgment (ACK).*)

$$Q_{FIdle} = \frac{((N_{aRxTx} - N_{uRxTx}) + N_{aRx} - N_{uRx})) \times Q_{idle}}{PDR} \tag{4.2}$$

Eq. 4.2 defines the contribution of idle slots to the total charge drawn in a slotframe.

The number of slots that are in the idle state are those that have been configured on the schedule as *RxDataTxAck* or *RxData*, but during which no data is received.

$$\begin{aligned} Q_{FSleep} = (N_{sleep} + (N_{aTxRx} - N_{uTxRx}) \\ + (N_{aTx} - N_{uTx})) \times Q_{sleep} \end{aligned} \tag{4.3}$$

Eq. 4.3 defines the contribution of *Sleep* slots. As in (4.2), the *TxDataRxAck* slots (N_{aTxRx}) that are not used N_{uTxRx} are in *Sleep* state. In this case, during a *TxDataRxAck* slot, the transmitter has no data to send and the radio is therefore never turned on.

$$Q_{FTxRx} = \frac{N_{uTxRx} \times (\frac{NBSent}{MaxPktSz} \times Q_{Tx} + (Q_{TxRx} - Q_{Tx}))}{PDR} \tag{4.4}$$

Eq. (4.4) defines the contribution of the N_{uTxRx} *TxDataRxAck* slots which are used. In that case two considerations are taken, the first one involves the number of bytes being sent, $NBSent$ which is considered with respect to the maximum packet size as our energy consumption measurements have been done with $MaxPktSz$[2] packets. In addition the Probability Delivery Ratio (PDR) expected by the network is considered. Analogously, Eq. (4.5) describes the energy consumed in slots that are of type *TxData*.

$$Q_{FTx} = N_{uTx} \times (\frac{NBSent}{MaxPktSz} \times Q_{Tx})PDR \tag{4.5}$$

Eq. (4.6) computes the contribution of *RxDataTxAck* slots. The number of used slots is defined as N_{uRxTx}, in addition the number of bytes being sent is also considered.

$$Q_{FRxTx} = N_{uRxTx} \times (\frac{NBSent}{MaxPktSz} \times Q_{Rx} + (Q_{RxTx} - Q_{Rx})) \tag{4.6}$$

[2] 127 bytes in IEEE802.15.4e.

Eq. (4.7) computes the contribution of *RxData* slots.

$$Q_{FRx} = N_{uRx} \times \frac{NBSent}{MaxPktSz} \times Q_{Rx} \tag{4.7}$$

Eq. (4.8) defines the total charge drawn during a slotframe, and is the sum of the contributions by the different types of slots.

$$\begin{aligned} Q_{slotframe} = {} & Q_{FIdle} + Q_{FSleep} \\ & + Q_{FTxRx} + Q_{FTx} + Q_{FRxTx} + Q_{FRx} \end{aligned} \tag{4.8}$$

Finally, (4.9) defines the battery lifetime of a node, in days, assuming it runs from a 3.6V power supply.

$$lf = \frac{B_{capacity} \times 3.6}{Q_{slotframe}} \times \frac{Length_{slot} \times Length_{slotframe}}{3600 \times 24} \tag{4.9}$$

4.1.5 Relay and Leaf Node evaluation

In order to support our realistic energy consumption model a small network composed of five GINA motes running the OpenWSN [22] protocol stack has been built. A topology is forced so that one of the motes is a relay to the base station and the rest of the motes are configured to be leaf nodes as depicted by Fig. 4.5. Leaf motes are configured to send a packet every 4 seconds leading to at most four packets in a slotframe at the relay mote. The setup has been built in our laboratory, all motes have been installed in a well known position and with a clear line of sight between them. Static schedules have been pre-configured at each node according to Fig 4.5, no overprovisioning have been configured although the data rate in leaf nodes has been configured to be 1 packet every 4 seconds, considerably lower than the supported data rate. Note that Tx slots that are not used are considered to be Sleep slots having the minimal energy consumption. No other IEEE802.15.4 devices were operating in the laboratory. Channel hopping has been used to mitigate the effect

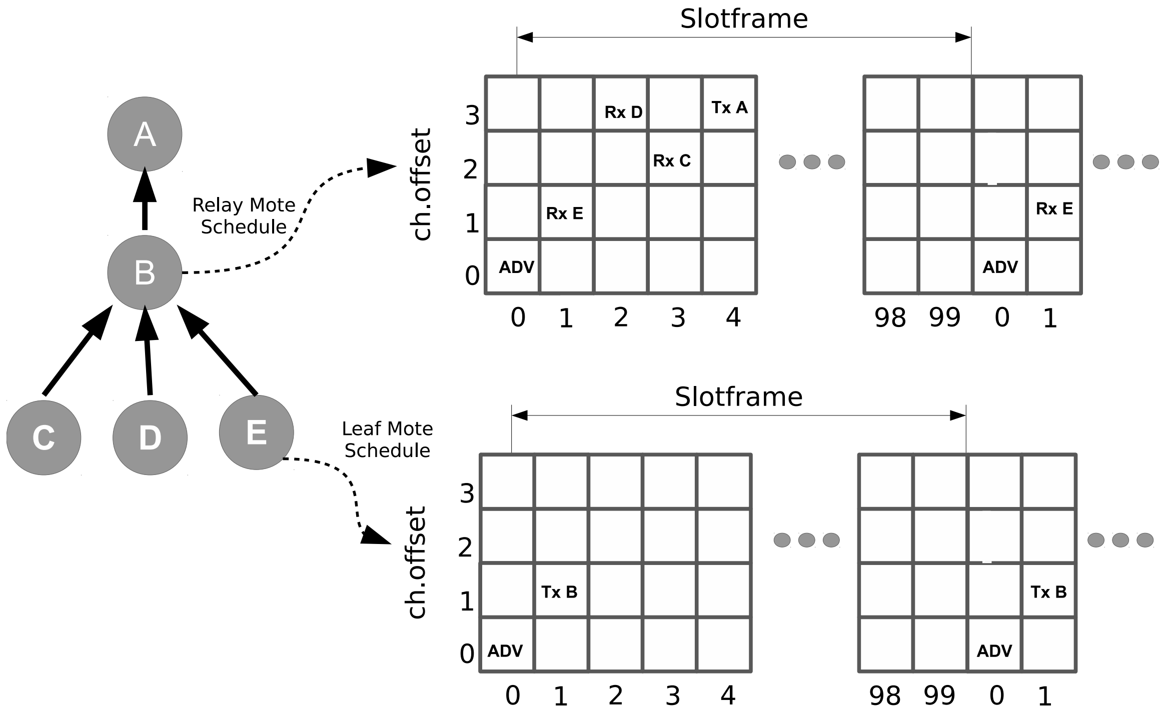


Figure 4.5: Experimental setting diagram. The schedule of Mote B is configured so it can relay information of the leaf nodes. Leaf nodes are configured to be able to send a packet once every 4 slotframes. Empty slots are Sleep slots. Slotframe is composed of 100 slots of 15ms each.

of multipath and external interference [44] and we have considered the expected channel errors to be negligible.

The experiment measures the charge drawn by both relay and leaf motes, as well as the radio duty cycle during a complete slotframe. The results are presented in Table 4.6. Simulation and experimental results match with an average current consumption of 581.9 μA for the relay mote in the experimental setup and 569.8 μA for the simulated setup. The leaf mote consumes less energy (455.0 μA and 415.4 μA respectively) due to less active slots in its schedule. The radio duty-cycle is computed in both cases (experimental and simulated) showing a clear match between our model and the experimental setup.

	Measured		Simulated	
Node Type	Relay	Leaf	Relay	Leaf
Slotframe Charge Drawn (μC)	872.8	682.5	854.7	623.1
Energy Consumed (μA)	581.9	455.0	569.8	415.4
Radio Duty-Cycle in %	1.14	0.47	1.12	0.47

Table 4.6: Charge by slotframe. Slotframes are composed of 100 slots of 15ms.

4.2 TSCH optimizations

The energy consumption model proposed in this chapter can be used to guide TSCH network configuration. In this section, we present two example cases. The GINA platform is assumed in this evaluation.

4.2.1 Synchronization Policy

IEEE802.15.4e defines two schemes to maintain one-hop synchronization. One is advertisement (ADV)-based, and another one is keep-alive (KA)-based. By applying the energy consumption model defined in Section 4.1, we can compare these approaches to understand the impact of each method in terms of energy consumption.

Considering a guard time of $2600\mu s$ and a clock drift of $\pm 30ppm$, (i.e. the clock difference between two nodes will be $60ppm$ in worst case), in order to maintain synchronization, one of the two above actions need to occur at least every $43s$. Leaving some margin, we consider $40s$ as the interval between two synchronization events. In the KA-based synchronization, a keep-alive frame and an ACK frame are both used, each containing 12 bytes. For ADV-based synchronization, only an Enhanced Beacon frame (EB) is needed. Usually, an EB contains time and channel information for synchronization, and initial link and slotframe information for new nodes to join the network. The length of the EB is therefore configurable (26 to 127 bytes), depending on the number of links and options encoded as Information Elements (IE) in the advertisement.

Fig. 4.6 shows the energy consumed on both the transmitter and receiver nodes by using either KA-based or ADV-based synchronization. Keep-alive packets have a fixed

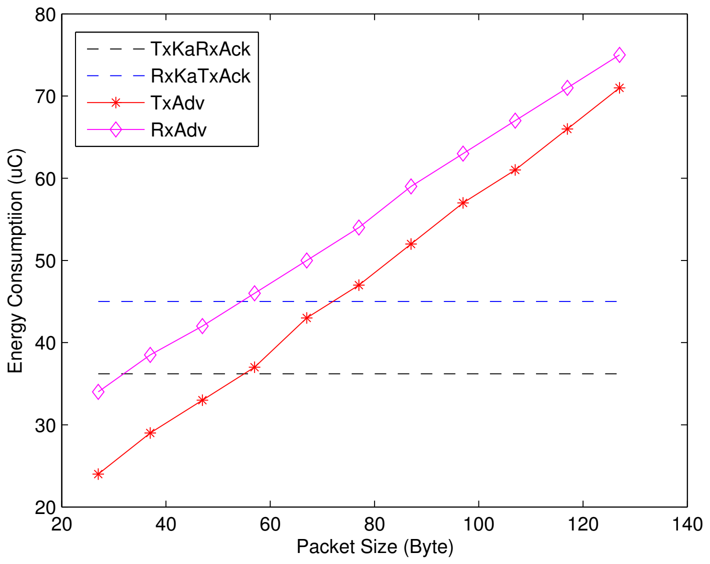


Figure 4.6: Energy consumption by Keep-alive and Advertisement for GINA.

length, while the length of EB packets can vary according to the number of links being announced. We assume that the interval between two EBs (which is defined as a trade-off between energy consumption and network joining requirements) is longer than the maximum interval between two synchronization actions (i.e 40 seconds in our example). In order to minimize the energy consumed during synchronization, it is required that EBs have a length shorter than 57 bytes (this is particular to the GINA platform as other devices will show other energy consumption numbers). Therefore, in the case where EBs cannot be shorter, keep-alive based synchronization is preferable.

Besides the impact on energy consumption discussed above, other factors have to be taken into consideration while scheduling EB and KA messages. For example, EB-based synchronization may have a big advantage for a node which has many children; and KA-based synchronization can lead to power advantages, when using more advanced synchronization mechanisms such as adaptive frame-based synchronization as described by Stanislowski *et. al.* [56].

4.2.2 The Cost of Overprovisioning

Overprovisioning happens when the scheduling entity allocates more links to account for the losses due to the wireless transmission, in order to meet the QoS requirements. Overprovisioning consists in scheduling extra slots for possible re-transmissions. The lower the packet delivery ratio of a link between two neighbors, the more overprovisioned slots are needed. This is present in the equations in Section 4.1.4. It is necessary to say that the queue length of node and the latency in a track are very strong functions of PDR and overprovisioning, as PDR is dependent on the environment, queue length and level of overprovision become the two main factors to vary in order to dimension the network. Overprovisioning impacts the energy consumption of the network while queue length impacts the memory requirements.

With regards to overprovisioning and from the transmitter's side, having *TxDataRxAck* or *TxData* slots in the schedule and not using them is equivalent to considering these slots to be sleep slots. When the packet delivery ratio of a link is better than expected, less re-transmissions happen, and some overprovisioned slots are unused. From the transmitter's point of view then, unused overprovisioned slots do not incur any extra energy consumption, since the radio stays off.

Fig. 4.7 presents the results of a simulation considering a slotframe with different available TxRx slots and different levels of usage of these slots. When a slot is used, the energy consumed by the slot frame is the same, regardless of the number of available TxRx slots, this is due to the fact that non-used TxRx slots are equivalent to sleep slots. Note also that the presented results are centered on the schedule configuration, i.e used slots cannot be more than available slots although logically the data-rate requirement in a node can be higher than the provided and in that case the consumption will be aligned with the maximum available data rate.

Overprovisioning impacts the energy consumption of the receiver side. That is, having *RxDataTxAck* or *RxData* slots on the schedule and not using them has an impact on the energy consumption of the network. The reason for that is that nodes in *RxDataTxAck*

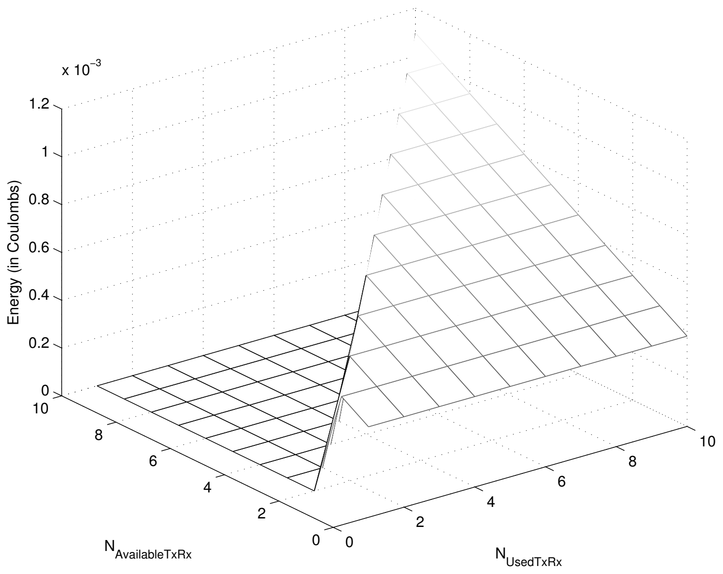


Figure 4.7: Impact of overprovisioning on the transmitter side in a slotframe.

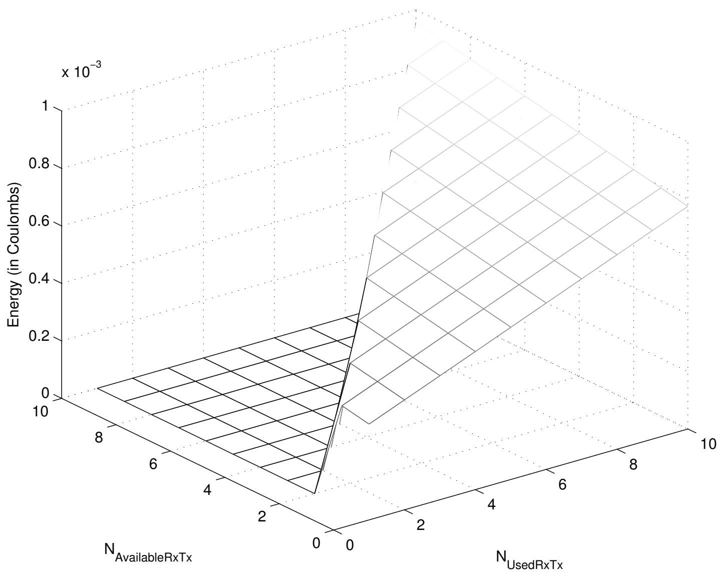


Figure 4.8: Impact of overprovisioning on the receiver side in a slotframe.

or *RxData* slot always listen whether a packet is received or not. In case of not getting a packet, the *RxDataTxAck* or *RxData* slot is equivalent to an idle listen slot because nodes only listen during the guard time period.

Fig. 4.8 presents the result of the simulation considering a slotframe with 10 configured RxTx slots and with different levels of usage. As it can be seen, for example for 1 used *RxDataTxAck* slot, the energy consumption increases depending on the number of available *RxDataTxAck* slots. The amount of that increment is in fact the difference of the energy consumption between an sleep slot and an idle listen slot.

Chapter 5

When Scavengers Meet Industrial Wireless[1]

The consolidation of industrial wireless communication into standards is leading to an increase in deployments throughout various industries today. Despite the technology being considered mature however, plant operators are reluctant to introduce mesh networks into their process, despite their very low energy profiles. One of the major obstacles of course is the power source of various wireless sensors and actuators. In many situations, batteries are too limited and difficult to replace. On top of that, certain devices cannot be powered by wires due to mechanical constraints, or because the cost of installing wires cannot be justified [57]. This is one area where scavenging technologies can be taken advantage of, especially where the industrial setting becomes an interesting harvesting source (momentum, temperature gradients, etc. [58]).

Vibrational energy scavenging [59] is well studied when combined with motors and rotational objects [60, 61], which are omnipresent in heavy industries. Peltier effect devices [62] have also proven to obtain interesting amounts of energy when temperature gradients

[1]**This chapter was published in a similar form as follows:**
Martinez, B.; Vilajosana X.; Chraim F.; Vilajosana I.; Pister K.S.J., "When Scavengers meet Industrial Wireless," Industrial Electronics, IEEE Transactions on, vol.PP, no.99, October 2014 doi: 10.1109/TIE.2014.2362891

are non-negligible, e.g in cold and hot pipes also common in many industries. Recently, RF scavenging is emerging as a very promising source of energy. Sometimes a combination of harvesting sources is even possible [63]. However, augmenting wireless sensing and actuation devices with energy scavenging requires a clear understanding of the power dynamics of each application since harvested energy constitutes a new source of variability to the device.

In this chapter, we present a methodology for precisely quantifying the power consumption on a per application basis, which allows the designers to appropriately size their scavenging and storage devices. The presented work goes one step further than previous research by closing the loop and modeling the energy inputs and outputs in a per subsystem basis. The model enables the adjustment of the application parameters and balancing the cost of communication, processing and data acquisition subsystems according to the amount of harvested energy. Here, we use the model presented in chapter 4 and augment it to cover all the sources of energy consumption in a device.

The chapter is organized as follows. Section 5.1 reviews the state of the art and related work. In Section 5.2 we present our model, broken down for data acquisition, processing and communication. Then, in Section 5.3, we apply our model to the case of motor vibration sensing, as means of an example. In Section 5.4 we discuss the implications of our model for energy scavenging, before concluding in Section 5.5.

5.1 Related Work

Recent approaches to model the use of scavenging techniques in industrial wireless application can be found in the literature. Wang et al. [64] present a model for the power consumption of a sensor network by analyzing the consumption of individual components that are involved in the communication. The provided analysis is suitable to dimension the energy spent during the communication process but neither scavenging techniques nor the cost of the application itself are considered. Torah et al. [60] assumes a dependence between the harvesting pattern and the applications needs. In their approach, they build an application that wakes the microcontroller up and transmits a packet only when enough energy has been harvested. The approach draws a best effort system that cannot provide any sort of determinism, which is often required for industrial applications, method clearly does not allow for the use of TSCH-type layers. On top, this work does not provide a clear modeling of the application energetic requirements nor a detailed network energy consumption analysis. Recently, Liu et al. [65] presented a completely autonomous system using energy scavenging proving the suitability of self-powered networked applications. This first demonstrator, however, is still far from industrial applications and yet, its energy consumption profile is not well defined. Waterbury et al. [61] presented a self-powered industrial application based on vibrational harvesting. Their work is focused on characterizing the amount of energy that can be obtained from an electric motor, complementing our work, but they do not outline how this energy is distributed among the application subsystems and therefore how this energy spending can be optimized. Gungor et al. [66] already pointed out the need for energy-harvesting techniques in industrial wireless sensor network and presented an initial study on how different energy-harvesting techniques influence the device design in general terms. Nasiri et al. [67] presented a deep study on how photovoltaic cells (PV) in indoor scenarios can be dimensioned to power low-power electronic devices. Nevertheless, a fixed application setup is used, and the problem of adapting the application parameters to the available energy is not addressed. Tan et al. [68] presented an analogous approach focusing the scavenging techniques on capturing the energy from thermal variations and photovoltaic cells. Recently, Magno et al. [69] present a wireless structural health monitoring system which incorporates a multi-source energy scavenging subsystem, the article analyzes the energy input obtained by the harvesters but does not take into account in their model the specific low power wireless communications protocols and bandwidth requirements to achieve deterministic industrial communications. Other energy sources such as RF scavenging have also been studied recently by Musseta and Gandelli et al. [70, 71] but the studies in question lack system-wide modeling, and do not point out the impact of various application components to dimension the RF scavengers. Piezoelectric vibrational scavenging modeling has been addressed by D'hulst et al. [72]. Their work focuses on the analysis and modeling of the vibrational scavengers that can be used to harvest energy from vibrating sources such as motors, but the paper lacks a system-wide view and a much-needed discussion of whether the obtained energy is enough to power a device.

In general, a systematic study of how energy is distributed in the whole system have not been addressed in the literature. The present chapter improves current state of the art by defining a general methodology introducing a new perspective that can be applied to different industrial applications. A model is presented as a tool that aims to facilitate the application dimensioning and scavenger selection at pre-deployment phase. The model takes into account the three components that play a fundamental role in a realistic industrial application: standard networking technologies, sensing and processing. This approach has not been tackled by the current literature, and results demonstrate that the contribution to the energy consumption of the different components is of the same order of magnitude and consequently, all of them must be taken into account in order to explore the feasible options and fine-tune the application parameters.

5.2 A Model for Early Power Estimation

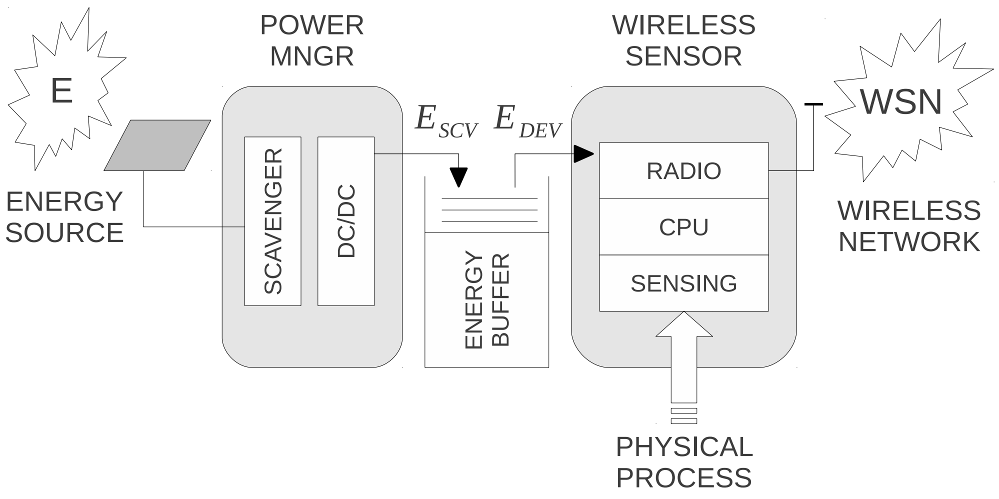


Figure 5.1: Self-Powered Wireless Sensor Device Diagram.

A *self-powered* wireless sensor device is characterized by a sustainable provision of energy. In a general model, as depicted in Fig. 5.1, the energy scavenged from the medium

E_{SCV} must be greater than the energy required by the device E_{DEV} along its operational lifetime. Formally:

$$E_{SCV} = \int_0^\infty P_{SCV}(t)dt \geq \int_0^\infty P_{DEV}(t)dt = E_{DEV} \tag{5.1}$$

Despite this being a necessary condition during the operation of a device, the instantaneous power supplied from the harvester $P_{SCV}(t)$ (conditioned by the environment), is in general independent of the energy consumption rate $P_{DEV}(t)$. The latter is governed by the specific application. Then, a properly sized energy buffer is required to balance the generation and demand asymmetries as shown by Jiang et. al. [73]. With that assumption in mind, dimensioning the energy generation and the energy consumption can be addressed independently. Thus, Eq. (5.1) can be reduced to a simpler form $\bar{P}_{SCV} \geq \bar{P}_{DEV}$ in terms of average behavior on the characteristic cycle of operation of the application. The problem of sustainability of a wireless sensor network is then described by a clear understanding of the energy generating pattern, the characterization of the operational energy demand and a dimensioning of the energy transfer buffer.

As shown in Eq. (5.2), the power required to operate a wireless sensor device can be broken down into three main blocks: energy required for data sensing (acquisition), for application (data processing), and communication (networking). Our model therefore is based on an atomic breakdown of each building block, where the instantaneous power consumption is integrated over the duration of the corresponding task, and averaged out its characteristic temporal scale (period of repetition). The next subsections go into a detailed analysis of each energy building block.

$$P_{DEV} = P_{NET} + P_{ACQ} + P_{PRC} \tag{5.2}$$

5.2.1 A Model for TSCH Networks Energy

TSCH networks show an ultra low power consumption profile due to the low power nature of IEEE802.15.4 compliant radios and due to the fact that nodes are synchronized and actions occur at specific moments in time, enabling nodes to optimize the usage of their resources. Since the actions that occur at each time slot are well known, the energy consumption can be modeled on a slot per slot basis. With precise modeling the task of precisely calculating the energy consumption of a network according to its traffic requirements becomes straightforward. We now apply the model derived in chapter 4 as a tool to estimate the energy consumption of the addressed scenarios. Note that our aim in this case is to introduce the use of the previously defined model into the presented methodology.

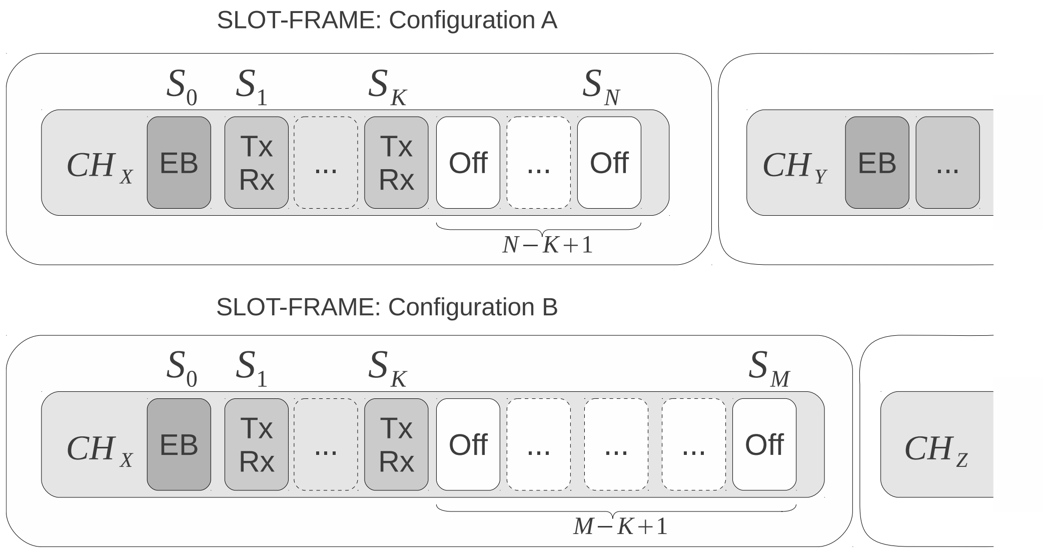


Figure 5.2: Example of two TSCH slotframe configurations with a different number of slots N and M, and assuming $M > N$. The first slot is used for network discovery by means of *Enhaced Beacons*. Then K *Data* slots for transmission and reception are common in both configurations. Configuration A has *N-K Sleep* slots (unused), while Configuration B has *M-K Sleep* slots, meaning that a node running in this configuration will be idle for longer periods. This *Slotframe* repeats over time.

In a TSCH network the slotframe length N_{SLOT} determines how often actions repeat, which usually depends on application requirements. The amount of scheduled cells (i.e *Transmit* or *Receive*) depends on the traffic requirements of the application and, as the transceivers often consume higher amounts of energy, the communication cost dominates as the traffic on the network increases. Latency, robustness and energy consumption are compromised.

Energy consumption can be reduced by increasing the length of the slotframe, i.e by inserting more *Sleep* slots or by disabling some *Active* slots so they become *Sleep* slots. A node only is active in certain of the timeslots in the slotframe, which are used to send or receive information. In the rest of non-active slots the node remains switched off. If the number of active slots remains constant and the slotframe size increases, the ratio of sleep slots increase. This means that the average energy spent by the node is smaller as it is less active. The same effect is obtained changing active per sleep slots. However, the reduction of activity comes at the cost of less bandwidth and increased latency. Reliability is also compromised, as less redundant links to neighbors are expected.

5.2.2 Modeling Data Acquisition Energy

It would be impossible to come up with a model that captures all of the sensing techniques. However it is reasonable to say that most applications fall under the following two categories: regular sensing (with a fixed interval) and event-driven sensing. In regular sensing, a sensor is woken up at regular intervals to collect one or more samples, and then sent back to sleep. Of course, when energy is freely available, the sensor can be left on permanently, but that is not the case of most battery-operated devices, where duty-cycling the sensor becomes necessary. In event-driven sensing, a random event triggers a series of samples from the sensor. This event can be internal to the sensor (e.g. crossing a pre-set threshold) or external (e.g. a request of sensor data coming from the plant operator).

To quantify the energy consumption of the sensing component in the wireless device, we must first look at the energy required to capture one sample. Then, the model is generalized to account for more than one regular sensing intervals (with different periods and sampling requirements) and various event types. Eq. (5.3) models the energy consumption of the acquisition component.

$$E_{ACQ} = E_{SMP} \cdot [\sum_{k=1}^{K} N_{S,REG}^{(k)} + \sum_{l=1}^{L} P_{S,EVT}^{(l)} \cdot N_{S,EVT}^{(l)}] \tag{5.3}$$

In Eq. (5.3) K is the number of sensor sampling in a regular basis, whilst L represents the number of sensors triggered by events. The following variables are used:

- E_{SMP} is the energy of one sample as presented in Fig. 5.5.
- $N_{S,REG}$ is the number of samples taken during one regular sensing interval
- P_{EVT} is the probability of an event occurring in one sensing interval
- $N_{S,EVT}$ is the number of samples taken following the occurrence of an event

5.2.3 Modeling Local Data Processing Energy

The sensing and networking components are common to all applications running on the same platform and thus, the same profiling can be reused for further modeling. In contrast, application developers need to be able to estimate the energy consumption of specific hardware at design time, without having to provide the actual hardware implementation. This enables the exploration of different alternatives while minimizing the risk.

To estimate the energy drained from the battery by an application task, we use a method proposed and validated originally in [74]. Starting from a high level description of the algorithm (e.g. Matlab/Octave), the number of operations to process the original sensed signal is recorded, accounting basically for the number of arithmetic operations: additions, multiplications, divisions and comparisons, the main actors in signal processing loops. Thus, depending on selected hardware architecture, we map these counters into the corresponding number of microcontroller clock cycles, and subsequently the latter is mapped into the corresponding energy expenditure.

This method offers accurate results as long as the CPU tasks rely mainly on arithmetic instructions (as digital signal processing algorithms do). However, it is no longer applicable when the microcontroller is involved in non-arithmetic based tasks, like dealing with a protocol stack. In that case, an alternative method is presented through the use of ISS simulators.

5.3 Application Oriented Model Validation

Rotary machines of all sizes are the propelling force behind many industries (refining, chemical, energy production...). In the constant quest for better efficiency and lower energy consumption, vibration monitoring of rotating machines becomes necessary. In short, plant operators aim at increasing the load on the machines and their running time, while reducing their down-time and energy consumption. This is accomplished by installing vibration sensors (accelerometers for example) and performing an harmonic analysis of those vibrations to monitor the health of the machinery [75]. For cost and logistical purposes, it is often desired that the sensing equipment be self-powered and wireless, thereby forcing the system designer to carefully plan the energy consumption [66]. In this section, we illustrate the model presented in 5.2 by taking the vibration monitoring application as our case study.

As described in section 5.2 and for each subsystem, the atomic energy contributors are identified. As a general rule, the current consumption is obtained from the device's datasheet. This consumption is then multiplied by the duration of each task (estimated according to some selected set of Network/Application parameters), to get the total charge drained at each stage.[2] The estimation of each contributor is compared with the real measurement to show the correctness of the proposed methodology.

The experimental setup we used in our application is shown on Fig. 5.3. The GINA

[2]In our case, all components are sourced to the same voltage level. The model is presented with currents instead of power, and charge instead of energy (i.e, normalized by the voltage). As the reference values in the datasheets of the components, batteries and scavengers are given in intensity units we avoid continuous conversions, and also facilitate the experimental measurement as only current proves are required.

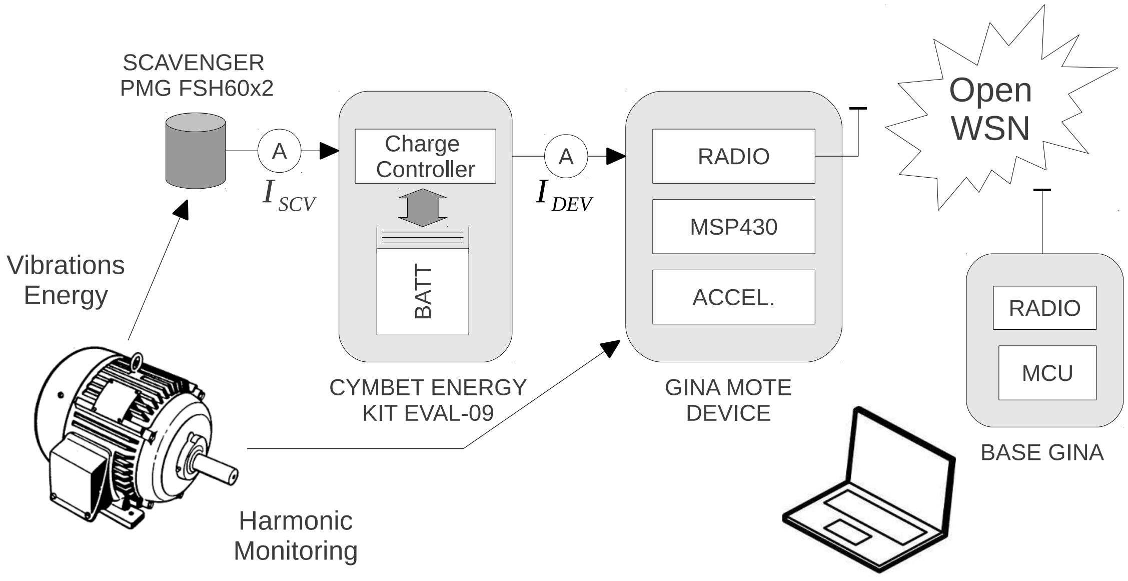


Figure 5.3: Experimental setup diagram.

platform [10] was used for sensing, processing and networking. It holds the Texas Instruments MSP430f2618 16-bit microcontroller, the Atmel AT86RF231 IEEE802.15.4 radio, and the STMicroelectronics LIS344ALHTR 3-axis analog accelerometer. For energy storage, a lithium polymer battery was used with a battery charger controller. The selected vibrational scavenger is the Perpetuum PMG FSH60x2, attached to the case of the motor in a position that maximizes both the vibrations transferred to the sensor as well as those being scavenged. All experiments are measured with the NI9203 16 Bits analog current acquisition module, with a resolution of $0.6\mu A$ per LSB.

5.3.1 Network Energy Profiling

To profile the energy consumed by TSCH networks we apply the model defined in [76] to determine the charge used in each slotframe with different network setups. This configuration has also been validated in an experimental setting with the GINA mote running the OpenWSN protocol stack [22]. In short, the average current required to maintain the network can be approximated by Eq. (5.4).

$$\bar{I}_{NET} \cong \frac{\bar{Q}_{MSG} \cdot N_{ACT}}{T_{SLOT} \cdot N_{SLOTS}} + \bar{I}_C \tag{5.4}$$

In Eq. (5.4), $\bar{I}_C$ accounts for the activity of the μC to manage the network (system), and can be regarded as constant. T_{SLOT} is a fixed network parameter. $\bar{Q}_{MSG}$, representing the average charge required per active packet, depends on the radio technology and the network state. A lower packet delivery ratio (PDR) implies a higher average current per packet arrived with success. This means that, once the number of active packets N_{ACT} in the slotframe is provisioned, the number of slots N_{SLOTS} in the frame becomes the control parameter for the network energy, giving a characteristic functional dependency on $I_{NET} \propto 1/N_{SLOTS}$ which can be seen in Fig. 5.4.

In this experiment slotframes from 10 to 50 slots have been used, all of them being configured to deal with the deterministic traffic of 3 packets per slotframe. The PDR measured in our setup was close to 100%. Fig. 5.4 shows the experimental data compared to the model adopted, and the predicted current consumption in environments with lower PDR.

5.3.2 Data Acquisition Profiling

In general, to estimate the energy required in collecting one sample, we have to look at a breakdown of the microprocessor/sensor activity during that sample. We have represented in Fig. 5.5 the following tasks:

- Sensor startup time (T_{SETT}).
- Microprocessor and sensor energy required to initiate sampling (could correspond to toggling a pin, or sending a command on a communication bus like I2C, SPI, etc).
- Analog-to-Digital conversion (T_{ADC}). This happens on the sensor if it is a digital one, on the microprocessor if the sensor is analog.

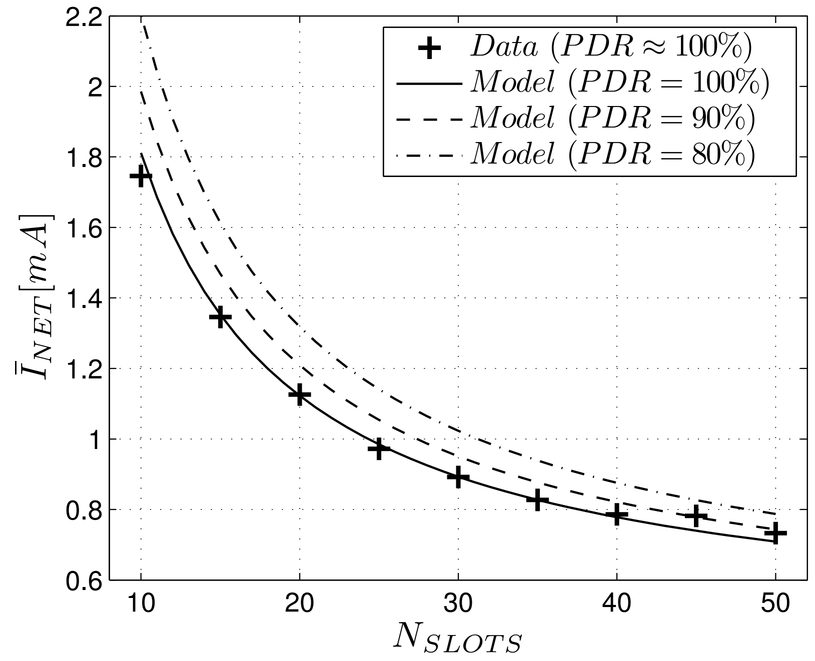


Figure 5.4: Network model for different values of PDR compared to experimental data. N_{ACT} is fixed to 3 active slots, and $T_{SLOT} = 0.015s$.

- Transfer of data from the ADC output to the processor memory (T_{CPU}). This task could be an internal memory operation or an external serial/parallel communication.

In our application example, we are dealing with an analog sensor that is turned on from the microprocessor, and which provides a reading through the processor's ADC. The sensor startup time is too long to turn it on and off between consecutive samples ($T_{SETT} \approx 5$ms), therefore, the sensor is required to remain in its On state while collecting the N_S samples. Consequently, sensor and processor must be considered independently.

The charge drained by the sensor in the data acquisition stage is computed using the total time the sensor must remain active while capturing N_S samples with a sampling period T_S. Then, the total charge is given by $Q_{SNR} = I_{SNR} \cdot N_S \cdot T_S$. On the other hand, as the ramp-up time of the CPU is fast enough, it can operate on a sample by sample basis. For each sample, the energy contribution can be obtained by looking at the number of cycles required by the internal ADC as well as the cycles required to move the data to

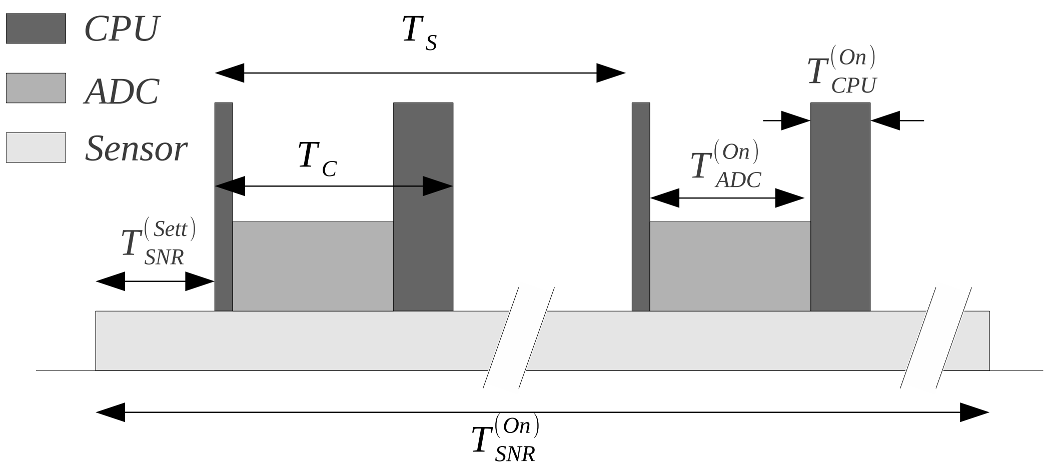


Figure 5.5: Energy consumption breakdown for one sample

memory. The time required to sample five ADC channels, convert them and commit them to memory is represented in Table 5.1, along with the total time T_C. Since each conversion is independent, the total charge required is proportional to the active time at each capture: $Q_{ADC} = I_{CPU} \cdot T_C \cdot N_S$.

When combining both contributions, a simple linear dependence on the number of samples is obtained $I_{ACQ} \propto N_S$. Finally, the average current of the acquisition block is computed by dividing the total charge by the time lapsed between consecutive records, as shown in Eq. (5.5). In this block two control parameters appear, the number of samples per record N_S and the time between consecutive records T_{RCD}.

$$\bar{I}_{ACQ} \cong \frac{N_S \cdot (T_S \cdot I_{SNR} + T_C \cdot I_{CPU})}{T_{RCD}} \tag{5.5}$$

5.3.3 Data Processing Profiling

In this section, we study the contribution to the energy cost of the data processing. The vibration monitoring application is mainly concerned with the frequency content of the acceleration signals. As such, a *Fast Fourier Transform* (FFT) is required to find the relevant dominant frequency and harmonics in the signal. Two different approaches

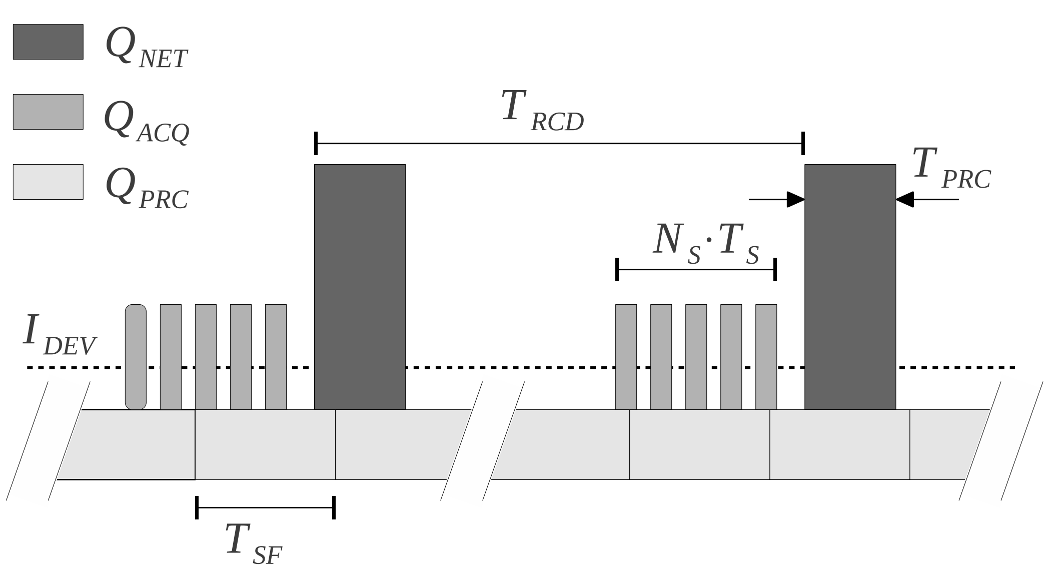


Figure 5.6: Application charge drained by different components

were addressed to obtain a proper estimation of the computation time of the FFT. First, a MATLAB implementation of the algorithm was used to estimate the number of cycles required, following the method used in [74]. In the second approach, the code was moved to a device specific implementation and was simulated with an Instruction Set Simulator (ISS)[3].

Table 5.2 compare both approaches, showing that the time estimated through the num-

[3]e.g. IAR workbench Simulator

Task	Cycles(MHz)	$\hat{T}[us]$	$T[\mu s]$	$\delta T[\%]$
Sample and Hold (ADC)	5×13 (5)	13.0		
A-D conversion (ADC)	5×16 (5)	16.0		
Samples Storage (CPU)	164(16)	10.2		
System wake-up		1.1		
Total Conversion T_C		40.3	39.4	2.2

Table 5.1: Time required by the acquisition of one single sample, broken down in different tasks. Estimated conversion time $\hat{T}_C$ is compared with measured time T_C. ADC and CPU parameters are obtained from MSP430 datasheet.

ber of MATLAB operations did not differ significantly from that derived from the instructions simulated in the ISS, and the later is in perfect agreement with the time measured with the algorithm running on the MSP430 processor. This demonstrates that it is possible to estimate the number of cycles required by the data processing algorithms at design stage, even without the final source code implementation. This processing time is used to obtain a fairly accurate energy consumption estimation, which is critical when designing the final solution over the platform.

N_{FFT}	$T_{PRC}[ms]$	$\hat{T}_{MAT}[ms]$	$\delta T_{MAT}[\%]$	$\hat{T}_{ISS}[ms]$	$\delta T_{ISS}[\%]$
64	14.6	14.3	2.1	14.7	0.4
128	33.5	33.2	0.8	33.5	0.0
256	75.3	75.8	0.6	75.4	0.1
512	167.8	170.2	1.4	168.0	0.1
1024	369.8	377.8	2.1	370.3	0.1

Table 5.2: Time and charge consumed in FFT processing. Simulated results $\hat{T}_{MAT}$ and $\hat{T}_{ISS}$ are compared with real measurements T_{PRC}

For modelling purposes, it is important to identify a functional dependency on a controlled parameter. The Radix-2 implementation of the FFT has a well known $Nlog(N)$ complexity. The associated energy cost in Eq. (5.6) is proportional to this relation, being $\bar{Q}_{OP}$ a magnitude representing the average cost per operation. Fig. 5.7 shows the fitting of Eq. (5.6) to the experimental data.

$$\bar{I}_{PRC} \cong \frac{\bar{Q}_{OP} \cdot N \cdot log(N)}{T_{RCD}} \tag{5.6}$$

5.3.4 Joint Model Validation

Once individual energy consumption contributors have been profiled, a joint model can be built as a tool to understand the main contributors on energy consumption for a specific application setting.

As asserted before, the main parameter involved in network consumption is N_{SLOTS},

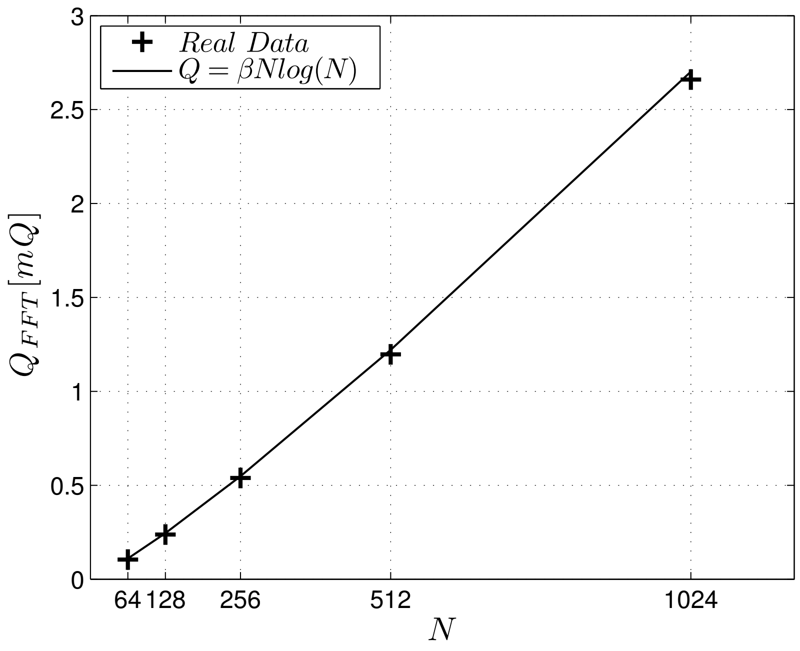
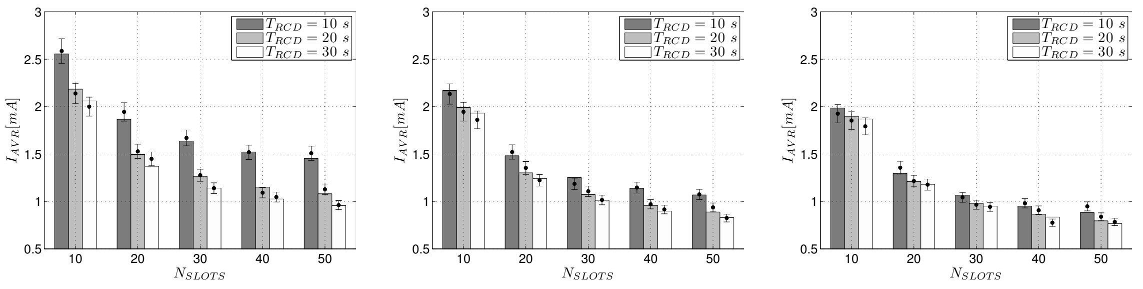


Figure 5.7: Charge drained to compute a N-point FFT. Numerical value for β can be found in Tab. 5.3.

which is related to the number of *Active* and *Sleep* slots. Assuming a fix number of *Active* slots, by incrementing N_{SLOTS} we are introducing *Sleep* slots to the schedule and therefore reducing the average consumption.

In terms of sensing and processing, two remarks should be made. First, a record is defined as the process of waking up, taking N_S samples and computing an FFT to analyze them. We assume that the number of points computed by the FFT and the number of samples taken by the ADC are the same $N_S \doteq N$. This means that the number of points in a record is a parameter that affects simultaneously the energy expenditure of both sensing and processing procedures. Second, once the number of points to be sampled and analyzed is fixed, the duty-cycled behavior of the application makes the average power depend directly on the time between records T_{RCD}. As the time between records is increased, less power is consumed. Therefore, the time interval between consecutive records gives us the time scale for power averaging.



(a) $N_{FFT} = 1024$ (b) $N_{FFT} = 512$ (c) $N_{FFT} = 256$

Figure 5.8: Comparison of Model predictions and experimental results for different parameter configurations.

Eq. (5.7) combines the three contributions (5.4), (5.5) and (5.6) differentiating between technological and applications parameters. Constants α, β, γ and δ only depend on the particular choice of sensor, MCU and radio technologies respectively. From the individual contribution, α represents the charge per sample $\bar{Q}_S$, β is interpreted as a cost per operation $\bar{Q}_{OP}$, while γ is an estimator of the average charge per message $\bar{Q}_{MSG}$. These constants can be easily modified to evaluate alternative technologies. In turn, N_{SLOTS}, N and T_{RCD} are application parameters that can be tuned in order to meet the specifications, once established the specific technology.

$$\bar{I}_{DEV} = \frac{\alpha N}{T_{RCD}} + \frac{\beta N log(N)}{T_{RCD}} + \frac{\gamma N_{ACT}}{T_{SLOT} N_{SLOTS}} + \delta \tag{5.7}$$

Contribution	Parameter	Model	Fitting	Units
Acquisition	α	3.50	3.49	$[\mu C]$
Processing	β	0.269	0.263	$[\mu C]$
Network (Radio)	γ	68.9	63.9	$[\mu C]$
Network (System)	δ	0.433	0.475	$[mA]$

Table 5.3: Technological parameters obtained by modeling compared to experimental data fitting.

Table 5.3 compares the value of α, β, γ and δ obtained by modelling with the experimental data fitting for a GINA mote running the OpenWSN protocol stack. With this technology fixed, Fig. 5.8 shows the experimental results compared with the predicted by Eq. (5.7) for different application configurations. The dots in the figures represent the measured current values, while the bars around them represent an allowable deviation of 5% due to variations in the PDR or other environmental conditions. The vertical bars show the estimated values by the model, plotted for different numbers of slots in a slotframe N_{SLOTS}, different recording intervals T_{RCD}, and a different number of samples collected and processed N. These results demonstrate the correctness of the model.

5.4 The Model in Action

By using this model, the applications engineer can make better informed technology-related decisions (both hardware and software) at the design stage. This methodical approach to wireless sensing allows for a reduction in the prototyping stage, and a quicker route to successful deployments. This section discusses how the presented model can be used for *i)* understanding the energy balance of an application, *ii)* assessing on the energy harvester selection and *iii)* coping with scavenging dynamics by adapting the application behavior.

5.4.1 Application Energy Balance

Experimental measurements on a system only provide the energy consumed by the entire system without giving any knowledge about the distribution of the consumption among the different subsystems. This makes it difficult to identify the main contributors to the energy consumption and the parameters on which depends the application, thereby limiting the scope of energy optimizations. The presented model, however, enables us to determine the amount of energy consumed by each of the subsystems in the application and understand the energy balance between components.

Fig. 5.9 presents a simulation obtained by applying Eq. (5.7) to different network and recording period configurations, considering 1024 samples per record. Bars present the contribution to the energy consumption of the network, sampling and processing components, according to the number of slots per slotframe and the recording interval.

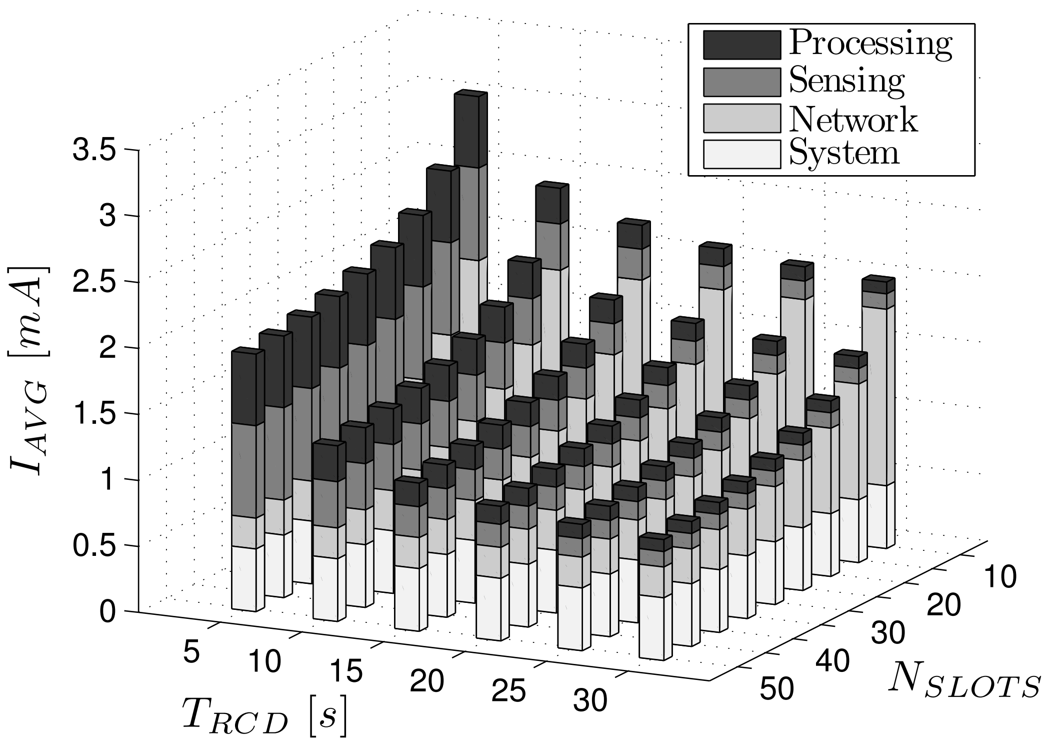


Figure 5.9: Contribution of recording interval and network configuration using 1024 points for the FFT (processing time fixed).

An asymptotic behavior can be seen in both axes of Fig. 5.9. Holding the interval time fixed, the overall energy consumption is reduced when increasing the number of slots in a slotframe. However, the asymptotic decrease limits the amount of energy that can be saved. At a certain point, increasing the number of slots in the network does not significantly reduce the energy consumed. Analogously, as the recording interval increases, the energy savings decrease. This graphical representation can be used as a tool to determine which of the parameters yields the highest energy savings once optimized.

5.4.2 Scavenging

Scavenger selection is guided by the condition defined in Eq. (5.1), and a desirable situation comes when the $\bar{P}_{SCV} \geq \bar{P}_{DEV}$ condition arises. Dimensioning of the scavenger depends to the configuration and application requirements of the wireless sensor devices. On the one hand, a fixed network and application configuration can easily be used to draw the upper bound of $\bar{P}_{DEV}$ and therefore select the right size for the scavenger. On the other hand, given a particular scavenger, the network and application parameters can be tuned to meet the available average current condition.

To draw an example of the second approach, a GINA mote was connected as shown in Fig. 5.3. The vibration of an industrial pump has been measured for 24 hours, in order to characterize the variability in the amplitude of vibrations and so the energy produced. Fig. 5.10 shows the spectrogram obtained for a chunk of 1 hour. In this particular application, the maxim peak of the spectrum have been found at 60Hz. At this frequency, the measured amplitude of acceleration is almost constant in time: $\overline{a} \pm \sigma_a = 0.0985 \pm 0.0016$ $[g]$. In our example, we selected the PMG FSH60x1 model, with a resonator adjusted at 60Hz. This device provide a current of $\bar{I}_{SCV} \approx 1.6\, mA$ for a 0.98 [g] (Fig. 5.11). Based on this response, Fig. 5.12 shows the suitable subset of parameters to make the application self-sustainable, defined by the region above the white curve. The dashed lines show the expected current inside 3σ limits.

5.4.3 Dynamic Scavenging

In many applications, the incoming energy to the system is only available intermittently. In other cases, the scavenger cannot provide the expected energy because some environmental variables have changed. In both cases, the application should be able to dynamically adapt to the new conditions so as to remain energetically sustainable. This is accomplished by selecting a suitable parameter and mapping the consumption to the expected energy input. In our application, the number of slots is fixed once the network has been established (the length of the slotframe cannot be changed without involving a cascade of changes in

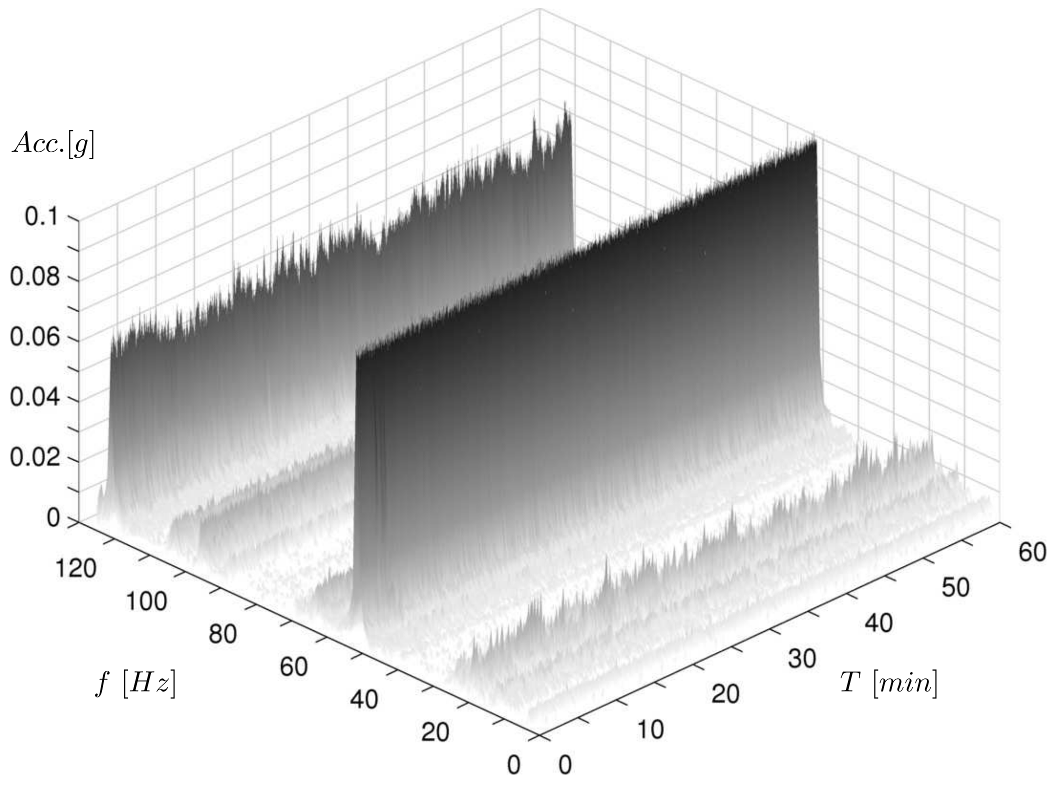


Figure 5.10: Spectrogram of the vibration magnitude during one hour of monitoring.

other nodes). *Active* slots can be deactivated and converted to *Sleep* slots, therefore reducing the throughput of the node. However this comes with a considerable energetic cost to the network as it might trigger certain rescheduling in other nodes. Then, a more suitable control parameter is the time lapse between records T_{RCD}. Alternatively, the number of points to compute the FFT can also be reduced, thus compromising the quality of the harmonic analysis.

To dynamically adjust energy consumption to variations of the energy scavenged, the application needs to keep track of the amplitude of the fundamental vibrational harmonic[4], analyze it and estimate the expected input current for the next cycle accordingly. With this information, the time for the next record can be scheduled. Fig. 5.13 shows the operational regions for different network schedules. To select the interleaving time, the vibrational amplitude is analyzed and the expected normalized current is computed. This value is used

[4]The fundamental harmonic is defined by the specific frequency that provides the main contribution to the harvested energy. That should be the nominal frequency of the selected harvester.

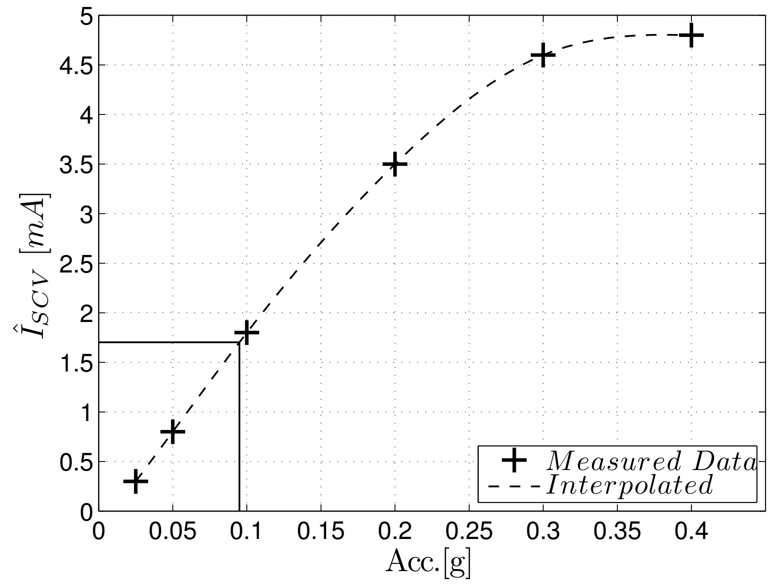


Figure 5.11: Vibration amplitude mapped to the current supplied by the harvester for this amplitude.

to determine the timeout for the next wake-up by finding the T_{RCD} in the border of the region.

5.5 Final Thoughts on the Model

This chapter addresses the convergence of energy scavengers with industrial wireless sensing and actuating applications. A methodology based on a parameterizable model has been presented in order to understand energy spending and facilitate scavenger selection. The methodology aims to reduce technology adoption/integration risks as energy consumption can be estimated precisely without the need of prototyping or building actual devices. We show that by decomposing the sources of energy consumption, optimization can be done in a more accurate manner, and system designer efforts can be put in the right direction consequently minimizing costs and risk. A precise estimation also enables a clear dimensioning of the components of the devices, especially with respect to the energy buffer and

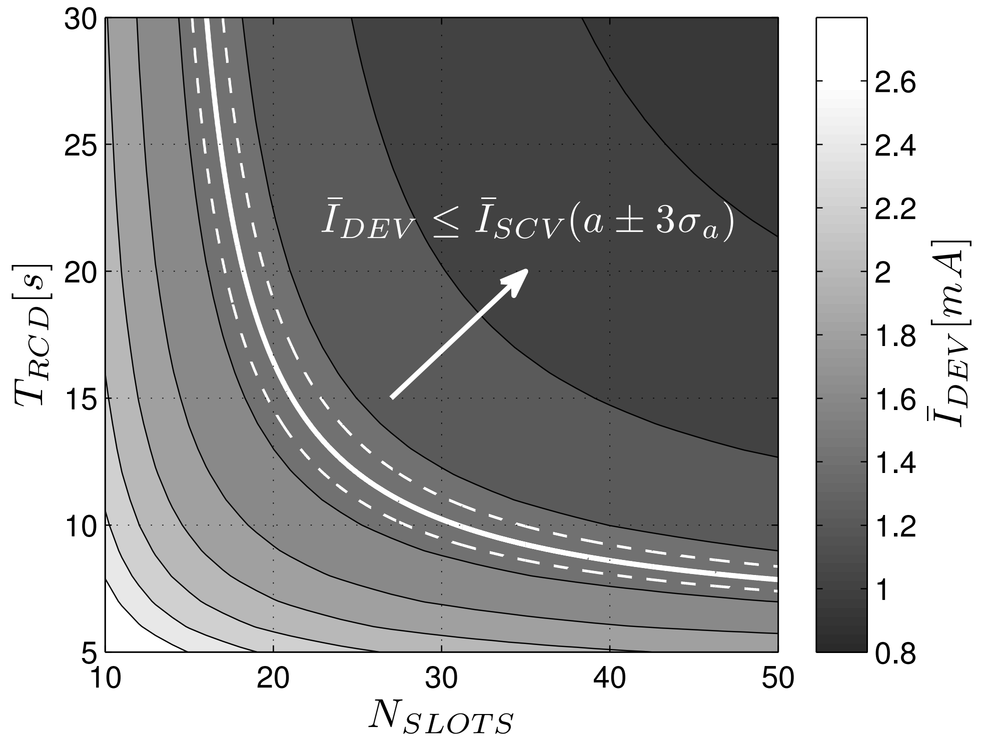


Figure 5.12: The gray scale of the side bar represent the device average current. Given the estimated production of the specific harvester (1.5mA in this example), the white line limits the self-powered region, that is, an acceptable combination of parameters that guarantee an autonomous operation.

harvester. It has been shown that TSCH networks facilitate the energy profiling due to their determinism and slotted structure.

Throughout this chapter, we make use of an industrial application to illustrate how an analytical model emerges as a tool to facilitate the application configuration and scavenger selection at pre-deployment stages. Given a TSCH-based wireless application and knowing its bandwidth, sampling and processing requirements, an accurate estimation of energy demands is used to determine what scavenger is required to make it self-sustainable. In addition, the parametrized model can be used to enable different modes of operation in case of varying requirements. The presented model has been validated experimentally using a wireless sensor network platform running the OpenWSN protocol stack.

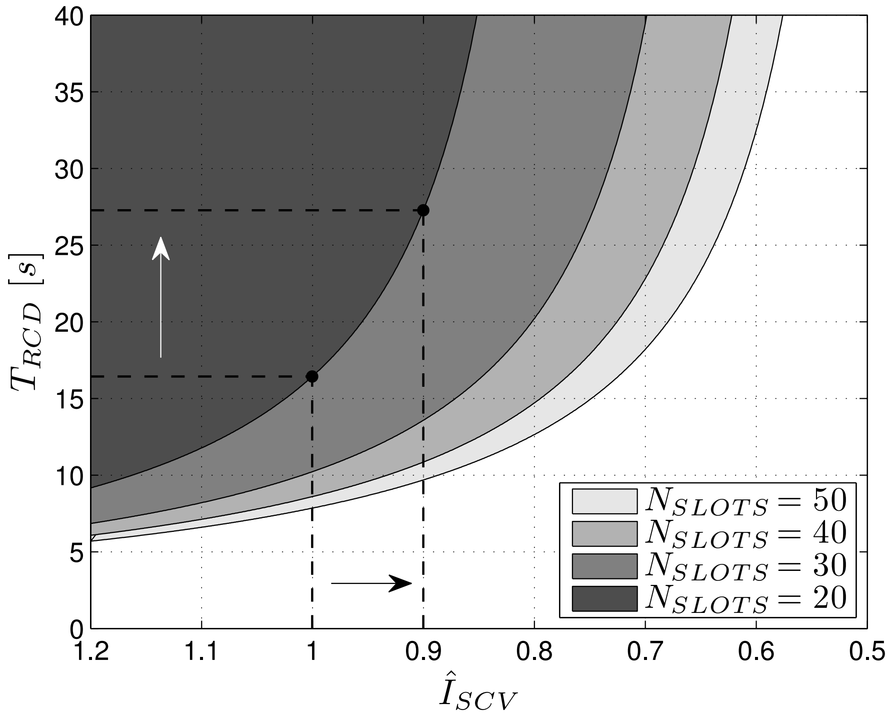


Figure 5.13: Dependence of the time between records and the normalized harvester current for different network configurations. Gray areas represent the feasible zones. The system must react to a reduction of the harvested current (black arrow) with an increase of the time lapsed between records (white arrow).

The presented methodology can be easily applied to a wide range of applications and sources of energy, and can be summarized in five simple steps: i) Find a source of energy in your environment. ii) Measure the magnitude of the available energy and its duty-cycle iii) Run a simulation within the feasible limits of application parameters iv) Select a suitable harvester sized accordingly to the energy available and the first estimation of the application consumption v) Fine-tune the application parameters accordingly with the selected harvester.

Conclusion

This thesis has demonstrated the possibility of achieving wireless applications that go beyond simplistic sensor sampling and reporting. Light was also shed on the scope of challenges facing the developer. As seen in the valve monitoring study, there are often sensing challenges on resource-constrained devices. The MEMS-based "peel-and-stick" solution demonstrates that, in spite of these challenges, a satisfactory performance can be achieved (valve angle accuracy of $\pm 5^\circ$ was recorded for quarter-turn models, while an accuracy of $\pm 10\%$ of a turn was obtained with the multi-turn valves). Taking an idea from concept to prototype to solution comes with its own hurdles. This was most apparent in the perimeter security chapter. Over a period of about two months, the cost-effective and scalable wireless solution demonstrated a 100% detection rate for all 91 attempted intrusions, and no false alarms. A lifetime of about 8.5 years is also achieved using a standard lithium C-cell battery. Finally, choosing the right algorithm and tuning it to the application at hand requires time and dedication. In the gas leak detection project, many iterations and refinements in the detection routines were necessary to find a suitable configuration. One of the most successful configurations achieved a gas plume detection rate of 91% with seven false alarms over three days. In terms of localization, the system estimated the leak locations to within three meters of the actual leak source.

By now it is understood that configuring a TSCH network is not as straightforward as setting up a Wi-Fi router. Every parameter can have dramatic effects on the performance of the network. Energy and reliability are the two main trade-offs to keep in mind. The network energy modeling tool of Chapter 4 can be very practical in evaluating various configurations. The last chapter also quantifies the overall energy consumption of the application (i.e. sensing, computation and communication). This tool shows a clear methodology for sizing energy scavengers and storage devices.

As a final note, our critical industrial environments are now acquiring the right wireless infrastructure. However, they are still awaiting all the innovative applications which will utilize this infrastructure to make their operation safer, more efficient and more environmentally friendly. With a good understanding of the underlying technologies and challenges, all of these undiscovered applications are waiting to happen.

Bibliography

[1] "WirelessHART Specification 75: TDMA Data-Link Layer," HART Communication Foundation Std., Rev. 1.1, 2008, hCF_SPEC-75.

[2] "ISA-100.11a-2011: Wireless Systems for Industrial Automation: Process Control and Related Applications," International Society of Automation (ISA) Std., May 2011.

[3] "802.15.4e-2012: IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer," IEEE Std., 16 April 2012.

[4] SmartMesh IP,Linear Technology, Milpitas, CA, USA, 2014, http://www.linear.com/products/Wireless_Sensor_Networks_-_Dust_Networks

[5] Hebert, D., ed. "Wireless Tech for Remote Valve Monitoring," Control 24 no. 11 (2011): 47.

[6] Kurtis Jensen, "The Reliability and Security of Wireless Valve Monitoring," Valve Magazine 20, no. 3, Summer 2008.

[7] Wireless Valve Monitoring System, 2012, Westlock Controls. September 2014, http://flowandcontrol.com/wp-content/uploads/2012/07/Wireless-brochure-WESPBR-09001-US-1204.pdf

[8] Radomsky, I.; Fuchs, R.; Kalman, I.; Nemenoff, A.; Gal, O., "Device and system for monitoring valves," U.S. Patent 7,886,766, issued February 15, 2011.

[9] Honeywell OneWireless XYR 6000 Valve Position Sensor, 2013, Honeywell. 13 March 2013, http://sensing.honeywell.com/honeywell-sensing-xyr6000-valve-position-sensor-productsheet.pdf

[10] Mehta, A.; Pister. K.S.J., "WARPWING: A Complete Open-Source Control Platform for Miniature Robots," in International Conference on Intelligent Robots and Systems (IROS).IEEE/RSJ, 2010.

[11] Kline, M.H.; Yeh, Y.; Eminoglu, B.; Najar, H.; Daneman, M.; Horsley, D.A; Boser, B.E., "Quadrature FM gyroscope," Micro Electro Mechanical Systems (MEMS), 2013 IEEE 26th International Conference on, pp.604,608, 20-24 Jan. 2013 doi: 10.1109/MEMSYS.2013.6474314

[12] Favre, J.; Jolles, B.M.; Siegrist, O.; Aminian., K., "Quaternion-based fusion of gyroscopes and accelerometers to improve 3D angle measurement," Electronics Letters 42, no. 11 (2006): 612-614.

[13] Cleveland, William S. "Robust locally weighted regression and smoothing scatterplots," Journal of the American statistical association 74 no. 368 (1979): 829-836.

[14] "Perimeter Security Sensor Technologies Handbook," Electronic Security Systems Engineering Division, North Charleson, South Carolina, 1997

[15] Yousefi, A.; Dibazar, A.; Berger, T.; "Intelligent fence intrusion detection system: detection of intentional fence breaching and recognition of fence climbing," IEEE Conference on Technologies for Homeland Security, pp.620-625, 12-13 May 2008

[16] Yousefi, A.; Dibazar, A.; Berger, T.; "Application of Non-homogenious HMM on Detecting Security Fence Breaching," Proceedings of the ICASSP (2010)

[17] Wittenburg, G.; Dziengel, N.; Wartenburger, C.; Schiller J., "A system for distributed event detection in wireless sensor networks," In Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, pp. 94-104. ACM, 2010

[18] Wald, A.; Wolfowitz, J., “Optimum character of the sequential probability ratio test,” Ann. Math. Statist., vol. 19, pp. 326339, 1948.

[19] Veeravalli, V., “Sequential decision fusion: theory and applications,” Workshop on Foundations Information/Decision Fusion with Engineering Applications, Proceedings of, August 1996

[20] Tsitsiklis, J., “On threshold rules in decentralized detection,” Decision and Control, 1986 25th IEEE Conference on, vol.25, no., pp.232-236, Dec. 1986 doi: 10.1109/CDC.1986.267213

[21] Fellouris, G.; Moustakides, G., “Decentralized Sequential Hypothesis Testing Using Asynchronous Communication,” Information Theory, IEEE Transactions on, vol.57, no.1, pp.534-548, Jan. 2011, doi: 10.1109/TIT.2010.2090249

[22] Watteyne, T.; Vilajosana, X.; Kerkez, B.; Chraim, F.; Weekly, K.; Wang, Q.; Glaser, S.D.; Pister, K.S.J., “OpenWSN: a Standards-based Low-power Wireless Development Environment,” Transactions on Emerging Telecommunications Technologies, vol. 23, no. 5, pp. 480–493, 2012.

[23] Hydra Asset Protection, Integrated Security Corporation, 2014, http://www.isc-hydra.com/index.php/how-hydra-operates.html

[24] “Directed Inspection and Maintenance at Gas Processing Plants and Booster Stations,” United States Environmental Protection Agency, BiblioGov 2013, ISBN 978-1288576357

[25] “Industry Sector Emissions,” United States Environmental Protection Agency, August 2014 [http://epa.gov/climatechange/ghgemissions/sources/industry.html]

[26] Gurney, D.; Alleman, D.; Kulp, T., “Development of hydrocarbon vapor imaging system for petroleum and natural gas fugitive emission sensing,” United States Department of Energy, 2004

[27] Laframboise, G.; Karschnia, B., “Improve exploration, production and refining with ‘add-at-will’ wireless automation,” Hydrocarbon Processing, p. 35 - 38, 2010

[28] "The selection and use of flammable gas detectors," UK Health and Safety Executive, 2004

[29] Murvay, P.; Silea, I., "A survey on gas leak detection and localization techniques," Journal of Loss Prevention in the Process Industries, 25, p. 966 - 973, Elsevier, 2012

[30] Liu, X.; Cheng, S.T.; Liu, H.; Hu, S.; Zhang, D.Q.; Ning, H.S.; "A survey on gas sensing technology," Sensors 12 (2012), 96359665

[31] GDU-Incus Ultrasonic Gas Leak Detector, Emerson Process Managment, 2014

[32] Kester, R. T., "A Real-time Gas Cloud Imaging Camera for Fugitive Emission Detection and Monitoring," Applied Industrial Optics: Spectroscopy, Imaging and Metrology, Optical Society of America, 2012

[33] Nofsinger, G. T.; Smith, K. W., "Plume Source Detection using a Process Query System," Proceedings of the Defense and Security Symposium Conference, Bellingham, WA, SPIE, 2004

[34] Huseynov, J.; Baliga, S.; Bagherzadeh, N.; Bic, L.; Dillencourt, M., "Gas-leak localization using distributed ultrasonic sensors," 16th SPIE Conference on Smart Sensor Phenomena, Technology, Networks, and Systems II, San Diego, CA, 2009

[35] Weimer, J.; Sinopoli, B.; Krogh, B. H., "Multiple source detection and localization in advection-diffusion processes using wireless sensor networks," 30th IEEE Real-Time Systems Symposium (RTSS), IEEE, 2009

[36] Chraim, F.; Pister, K., "Wireless Valve Position Monitoring: A MEMS Approach," 39th Annual Conference of the IEEE Industrial Electronics Society, 2013

[37] Chraim, F.; Pister, K.S.J., "Smart Fence: Decentralized Sequential Hypothesis Testing for Perimeter Security," Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Sensor Systems and Software, Vol 122, p. 65 - 78, 978-3-319-04165-0, 2013

[38] Lavielle, M., "Optimal segmentation of random processes," Signal Processing, IEEE Transactions on, volume 46, number 5, p. 1365 - 1373, 1998

[39] Varon, C.; Testelmans, D.; Buyse, B.; Suykens, J.; Van Huffel, S., "Robust artefact detection in long-term ECG recordings based on autocorrelation function similarity and percentile analysis," Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE, p. 3151 - 3154, 2012

[40] Micó, P.; Mora, M.; Cuesta-Frau, D.; Aboy, M., "Automatic segmentation of long-term ECG signals corrupted with broadband noise based on sample entropy," Computer methods and programs in biomedicine, Elsevier, volume 98, number 2, p. 118 - 129, 2010

[41] Doherty, L.; Lindsay, W.; Simon, J., "Channel-specific wireless sensor network path data," ICCCN, IEEE, 2007, pp. 89–94.

[42] Toscano, E.; Bello, L.L., "Multichannel Superframe Scheduling for IEEE802.15.4 Industrial Wireless Sensor Networks," IEEE Trans. Industrial Informatics, vol. 8, no. 2, pp. 337–350, 2012.

[43] Burri, N.; Rickenbach, P.V.;Wattenhofer, R., "Dozer: ultra-low power data gathering in sensor networks," IPSN 07, 2007.

[44] Watteyne, T.; Mehta, A.; Pister, K.S.J., "Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense," Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN), October 2009.

[45] Schmid, W.; Dutta, P.; Srivastava, M.B., "High-resolution, low-power time synchronization an oxymoron no more," Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, ser. IPSN '10, New York, NY, USA, ACM, 2010, pp. 151–161.

[46] Pister, K.S.J.; Doherty, L., "Tsmp: Time synchronized mesh protocol," Proceedings of the IASTED International Symposium on Distributed Sensor Networks (DSN08), 2008.

[47] Kohvakka, M.; Kuorilehto, M.; Hännikäinen, M.; Hämäläinen, T.D., "Performance analysis of ieee 802.15.4 and zigbee for large-scale wireless sensor network applications,"

Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks, ser. PE-WASUN '06 New York, NY, USA: ACM, 2006, pp. 48–57.

[48] Pollin, S.; Ergen, M.; Ergen, S.; Bougard, B.; Der Perre, L.; Moerman, I.; Bahai, A.; Varaiya, P.; Catthoor, F., "Performance analysis of slotted carrier sense IEEE 802.15.4 medium access layer,"Trans. Wireless. Comm., vol. 7, no. 9, pp. 3359–3371, Sep. 2008.

[49] Wang, Q.; Yang, W., "Energy consumption model for power management in wireless sensor networks," Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on, 2007, pp. 142–151.

[50] Casilari, E.; Cano-Garcia, J.; Campos-Garrido, G., "Modeling of current consumption in 802.15.4/zigbee sensor motes," Sensors, vol. 10, no. 10, pp. 5443–5468, May 2010.

[51] Khader, O.; Willig, A., "An energy consumption analysis of the wirelessHART tdma protocol," Computer Communications, vol. 36, no. 7, pp. 804 – 816, 2013.

[52] Palattella, M.; Accettura, N.; Dohler, M.; Grieco, L.; Boggia, G., "Traffic aware scheduling algorithm for reliable low-power multi-hop IEEE 802.15.4e networks," Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on, 2012, pp. 327–332.

[53] Morell, A.; Vilajosana, X.; Vicari, J.L.; Watteyne, T., "Label switching over ieee802.15.4e networks," Trans Emerging Tel Tech., 2013.

[54] Accettura, N.; Palattella, M.R.; Boggia, G.; Grieco, L.; Dohler, M., "Detas: a decentralized traffic aware scheduling technique enabling iot-compliant multi-hop low-power and lossy networks," Second IEEE WoWMoM Workshop on the Internet of Things: Smart Objects and Services, IoT-SoS, 2013.

[55] Thubert, P.; Watteyne, T.; Palattella, M.R.; Vilajosana, X.; Wang, Q., "Ietf 6tsch: Combining ipv6 connectivity with industrial performance," The Seventh International

Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, July 2013.

[56] Stanislowski, D.; Vilajosana, X.; Wang, Q.; Watteyne, T.; Pister, K.S.J., "Adaptive synchronization in ieee802.15.4e networks," IEEE Transactions on Industrial Informatics, vol. 9, no. 10, pp. 600–608, Jun. 2013.

[57] Tubaishat, M.; Madria., S.K., "Sensor networks: an overview," Potentials, IEEE, 22(2):20–23, 2003.

[58] Steingart, D.; Polastre, J., "Energy harvesting white paper," White Paper, 2008

[59] Khaligh, A.; Zeng, P., "Kinetic energy harvesting using piezoelectric and electromagnetic technologies: State of the art," Industrial Electronics, IEEE Transactions on, 57(3):850–860, 2010.

[60] Torah, R.; Glynne-Jones, P.; Tudor, M.; O'Donnell, T.; Roy, S.; Beeby, S., "Self-powered autonomous wireless sensor node using vibration energy harvesting," Measurement Science and Technology, 19(12):125202, 2008.

[61] Waterbury, A.; Wright, P.K., "Vibration energy harvesting to power condition monitoring sensors for industrial and manufacturing equipment," Proc. of the Institution of Mech. Engineers, Part C: Journal of Mechanical Engineering Science, 227(6):1187–1202, 2013.

[62] Carmo, J.P.; Goncalves, L.M.; Correia, J.H., "Thermoelectric microconverter for energy harvesting systems," Industrial Electronics, IEEE Transactions on, 57(3):861–867, 2010.

[63] Colomer-Farrarons, J.; Miribel-Catala, P.; Saiz-Vela, A.; Samitier, J., "A multiharvested self-powered system in a low-voltage low-power technology," Industrial Electronics, IEEE Transactions on, 58(9):4250–4263, 2011.

[64] Wang, Q.; Hempstead, M.; Yang, W.; "A realistic power consumption model for wireless sensor network devices," Sensor and Ad Hoc Communications and Networks, 2006.

SECON'06. 2006 3rd Annual IEEE Communications Society on, volume 1, pages 286–295, 2006.

[65] Liu, V.; Parks, A.; Talla, V.; Gollakota, S.; Wetherall, D.; Smith, J.R., "Ambient backscatter: wireless communication out of thin air," Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, SIGCOMM '13, pages 39–50, New York, NY, USA, 2013. ACM.

[66] Lu, B.; Gungor, V.C., "Online and remote motor energy monitoring and fault diagnostics using wireless sensor networks," Industrial Electronics, IEEE Transactions on, 56(11):4651–4659, 2009.

[67] Nasiri, A.; Zabalawi, S.A.; Mandic, G., "Indoor power harvesting using photovoltaic cells for low-power applications," Industrial Electronics, IEEE Transactions on, 56(11):4502–4509, 2009.

[68] Tan, Y.K.; Panda, S.K., "Energy harvesting from hybrid indoor ambient light and thermal energy sources for enhanced performance of wireless sensor nodes," Industrial Electronics, IEEE Transactions on, 58(9):4424–4435, 2011.

[69] Magno, M.; Boyle, D.; Brunelli, D.; O'Flynn, B.; Popovici, E.; Benini, L., "Extended wireless monitoring through intelligent hybrid energy supply," Industrial Electronics, IEEE Transactions on, 61(4):1871–1881, 2014.

[70] Gandelli, A.; Mussetta, M.; Pirinoli, P.; Zich, R.E., "Optimization of integrated antennas for wireless sensor networks," Proc. SPIE, volume 5649, pages 9–15, 2005.

[71] Mussetta, M.; Shadmehr, H.; Grimaccia, F.; Gandelli, A.; Zich, R.E., "Optimization of a radio frequency energy harvesting device," Evolutionary Computation (CEC), 2012 IEEE Congress on, pages 1–5, 2012.

[72] D'hulst, R.; Sterken, T.; Puers, R.; Deconinck, G.; Driesen, J., "Power processing circuits for piezoelectric vibration-based energy harvesters," Industrial Electronics, IEEE Transactions on, 57(12):4170–4177, 2010.

[73] Jiang, X.; Polastre, R.; Culler, D., "Perpetual environmentally powered sensor networks," Proceedings of the 4th international symposium on Information processing in sensor networks, IPSN '05, Piscataway, NJ, USA, 2005. IEEE Press.

[74] Zordan, D.; Martinez, B.; Vilajosana, I.; Rossi, M., "On the performance of lossy compression schemes for energy constrained sensor networking," ACM Transactions on Sensor Networks, 2014.

[75] Muszynska, A., "Vibrational diagnostics of rotating machinery malfunctions," International Journal of Rotating Machinery, 1(3-4):237–266, 1995.

[76] Vilajosana, X.; Wang, Q.; Chraim, F.; Watteyne, T.; Chang, T.; Pister, K.S.J., "A Realistic Energy Consumption Model for TSCH Networks," Sensors Journal, IEEE, vol.14, no.2, pp.482,489, Feb. 2014 doi: 10.1109/JSEN.2013.2285411