Benefits of building Wireless Sensor Networks on Commodity Hardware and Software Stacks

Nigel B. Bajema*, Jarrod Trevathan*, Neil W. Bergmann†, Ian Atkinson*, Wayne Read*, Adam Scarr*, Yong Jin Lee*, and Ron Johnstone‡
*eResearch Centre – James Cook University, Townsville, Australia
†School of Information Technology and Electrical Engineering – University of Queensland, Brisbane, Australia.
‡School of Geography, Planning and Environmental Management – University of Queensland, Brisbane, Australia

***Abstract*—The majority of wireless sensor networks are built on bespoke platforms, that is, custom designed and built hardware with a light weight software stack. There are a number of advantages to this approach. First, the ability to closely match and minimise the resource requirements (e.g., power consumption and communications protocols) to those that are suitable for the intended deployment. Second, as an entire hardware and software stack is often designed or at least optimised for each deployment, the latest advances can be quickly incorporated. However, this model generally requires the expertise of hardware and software engineers to design and build the system. In turn, this increases the cost and tends to shift the focus away from the initial science towards the development of the wireless sensor networks. This paper explores the utility and practicality of building wireless sensor networks based on commercially available embedded single board computing platforms using standard consumer operating systems. Our test bed was built using Gumstix computing platform, running a Linux Operating System (OS) with a java-based middleware coupled to low-cost scientific grade sensors. Test deployments have found this to be a highly versatile solution, able to leverage the flexibility of commodity hardware and software while maintaining reasonable utility.**

***Index Terms*—WSN; Commodity Stack**

I. Introduction

A Wireless Sensor network (WSN), in its simplest form, is made up of a number of nodes comprised of a computation element coupled with a wireless transceiver and a number of transducers embedded into a sensor field. Whilst conceptually simple, actual construction and deployment of WSNs are complicated by the need to balance a number of competing constraints. These include available power, meaningful sample rates/geographical distribution, size requirements, processing and filtering requirements, security requirements, robustness requirements, and the lifetime of the deployment. Furthermore, there are issues relating to archiving, management, provenance, and quality assurance of the collected data to deal with. In spite of this WSNs are widely recognised as having the potential to significantly enhance the way scientists collect data from the field.

WSNs have enormous potential to change the way we interact with the world and significant consequences for the way we conduct research. In fact it is estimated that by 2020 in-situ sensing will measure everything, everywhere [1], [2]. There is a long way to go before we reach this prediction, however a lot of work is being done towards this. Currently researchers are using WSNs in the following domain spaces: military [3], environment [4], biomedical [5], structural [6], and habitat monitoring [7] to name just a few. Another stepping stone on the journey is the development of commercially available WSNs that scientists and others practitioners can purchase and deploy.

Unfortunately most commercial WSN offering are "still geared towards research prototypes and are currently not yet mature enough for deployment in practical scenarios" [6]. Even though there are a number of commercial WSN providers their equipment is "typically only kits of building blocks and a non-trivial integration effort is required" [6]. Building a commercially available WSN system is a primary goal of the Smart Environmental Monitoring and Analysis (SEMAT) project [8].

A great deal of research is going into WSNs pushing micro controllers into increasingly smaller power footprint [9]. However there are those who are realising the benefits of utilising newly available high (processing) power commercially available single board embedded computers, and building these into low/high power hybrids [10]. Whilst Bespoke platforms can be specifically tuned to the requirements of any specific deployment, the

level of customisation required in both hardware systems and software can increase costs and defuse focus of a given project. These systems generally fall on the low end of computing power, to meet their stringent power requirements, and can be less flexible due to their highly targeted nature. However, as the power requirements of commodity single board computers drop and the processing power grows, the opportunity to leverage commodity hardware and software stacks becomes a reality in the domain of WSNs. This has the potential to remove a lot of the integration issues that Stajano *et.al.* [6] suggest inhibits the deployment of current commercial WSN platforms.

SEMAT aims at producing a commercially available WSN with minimal end user integration and programming requirements. This will be achieved through a multi-stage prototyping and testing schedule. The first stage, consisting of a purely commodity approach, explores the utility and practicality of building WSNs based on commercially available embedded single board computing platforms using standard consumer operating systems. Our test bed was built using the Gumstix computing platform, running a Linux Operating System (OS) with a java-based middleware coupled to low-cost scientific grade sensors. Test deployments have found this to be a highly versatile solution, able to leverage the flexibility of commodity hardware and software while maintaining reasonable utility.

This paper is organised as follows: Section II describes the architecture and deployment of our system. Section III presents some result on power usage and CPU utilisation for the proposed architecture. Section IV provides some concluding remarks.

II. Architecture and Deployment

The design of a WSN is heavily informed by the spatial and temporal resolution requirements of the data collection. In this section we will discuss the evolving deployment goals of our Deception Bay test bed and the resulting architecture of our WSN.

A. Deployment Goals

In an effort to determine the causes of Lyngbya algae outbreaks in Queensland's Deception Bay, the DHI and the University of Queensland (UQ) developed a model for their prediction [8]. The primary goal of this test deployment was to collect the data required by the model. The secondary goal was to establish the feasibility of building WSNs from cheap commodity hardware and software solutions. As such, the input requirements for

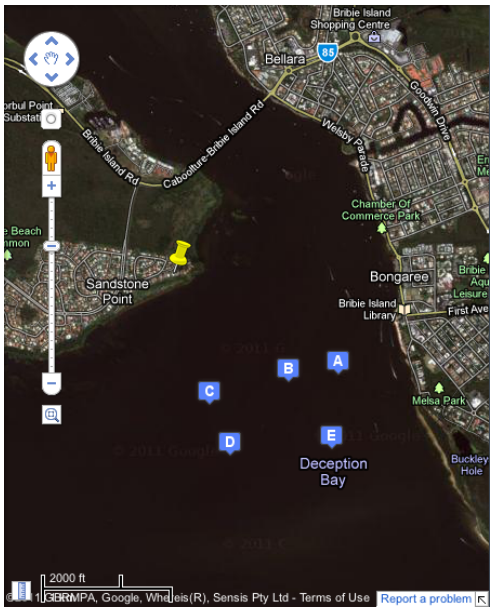

Fig. 1. The geographical configuration of the Deception Bay Deployment.

the model and the commodity design constraints informed the decisions about the geographical positioning, type, number, and sampling period of the nodes and their sensors.

The geographic configuration of the Deception Bay test bed underwent a number of changes during its planning. Initial estimates projected a maximum range no larger than 500m from any buoy to the base station. During development these estimates were revised up, firstly to 1000 meters and then to 1700 meters. Figure 1 illustrates the final geographical configuration of the deployment. The location of the base station is indicated by the yellow pin and the node ranges were as follows: Node A - 1.4km, Node B - 1.3km, Node C - 1.0, Node D - 1.4km, and Node E - 1.7km.

Although the environmental parameters that impact the growth of Lyngbya algae falls outside the scope of this paper, the number of sensors and the sampling period has a direct impact on the design constraints. To meet the input requirements of the above model we needed four sensors sampling at fifteen minute intervals. This test bed, although it was deployed in a marine environment, was for a relatively short period and not that remote. Therefore, issues resulting from bio-fouling were mitigated by regular maintenance periods carried out for the duration of the deployment. The next section describes the design we settled on to meet the goals of our Deception Bay deployment.

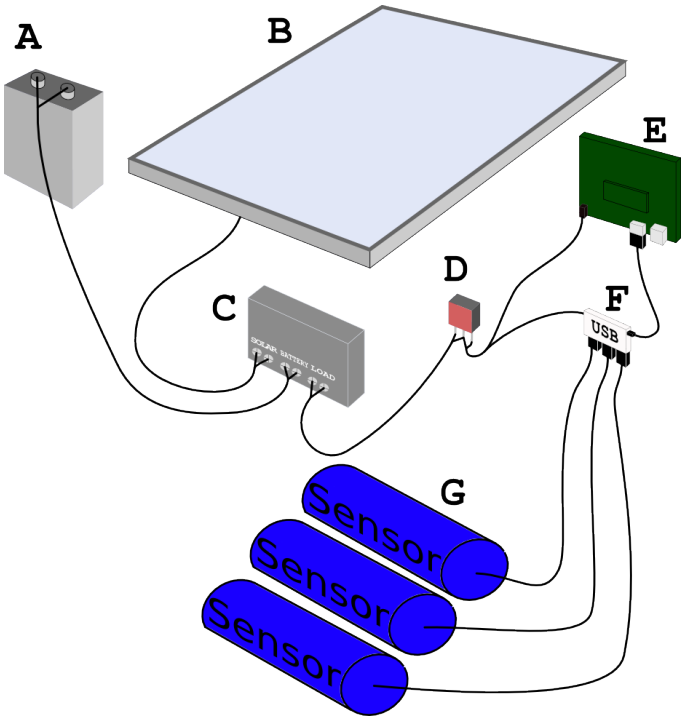

Fig. 2. The Architecture of a SEMAT Node.

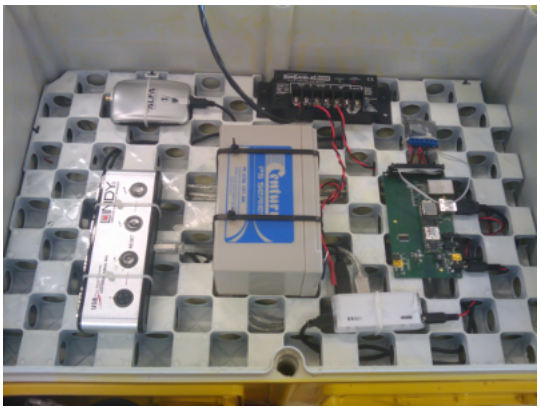

Fig. 3. A SEMAT Node.

B. Hardware

This section will discuss the hardware we used to build our WSN nodes, as pictured in Figure 3. The hardware used to construct our WSN nodes can be divided into four sections:

- The power system, this includes the solar panel, battery, and regulators.
- The computation and communication system.
- The sensor systems, including their interface adaptors etc.

The power system consists of elements A, B, C, and D from Figure 2.

A A 12 volt sealed lead acid battery.

B A 10 watt solar panel.

C A solar regulator – a Morning Star SunSaver-6L Solar Controller. The solar regulator manages the power flows between the solar panel, the battery and the power for the rest of the system.

D A switch mode regulator – a TRACO TSR1-2450 DCDC 1A step down switching regulator powers the Overo and its various components.

The computation and communication subsystem is comprised of a Gumstix Overo Air, a Chestnut43 expansion board, and an USB Wi-Fi module. A Gumstix Overo Air is a 600MHz ARM Cortex-A8 CPU with 256MB of RAM, Bluetooth and low powered Wi-Fi (802.11 b/g). A Chestnut43 expansion board provides interfaces for Ethernet, USB A(Host)/mini-AB/mini-B, Audio, and a 40-pin header providing access to the Overo's I^2C and SPI buses, and six ADCs. During the evolution of the geographical configuration requirements, with the maximum link distance growing from 500m to 1.7km, it became apparent that the on-board Wi-Fi chipset was not going to be powerful enough. Fortunately, the commodity nature of our deployment allowed us to find a suitably powerful substitute and simply plug it into the Overo's USB port; we ended up using a Alfa AWUS036NH. This of course had an impact on the power budget.

As the WSN node spends most of its time in standby (see Section II-D on Duty Cycles) it requires sensors that can operate independently during its downtime. Odyssey units were chosen for our sensing platform because of the following characteristics:

- Relatively cheap.
- Autonomous data logger and sensor.
- Already used by practitioners in the field.
- Can interface with our system.

Odyssey sensors were designed to be used as standalone units with a manual deployment cycle. Each Odyssey unit contains both a data logger and one or more sensors. These units would be individually programmed in the lab then deployed into the field. After their deployment period they would be collected, have their data read, be re-programmed, and finally re-deployed.

Integrating the Odyssey units with our system required us to address both physical interface and communication protocol issues. As Gumstix Overo Airs don't have any of the RS232 ports required to interface with Odyssey units. A USB to 4xRS232 adaptor was used to provide this capability. The software provided by the vendors

for use with the Odyssey units only works in a Windows PC environment and cannot be used in an ARM Linux environment. As the vendor would not provide a description of the protocol required to communicate with the Odysseys units, we reverse engineered it. With the ability to communicate with the Odyssey units the WSN node could then task them with a logging schedule, enter standby mode, then wake up and retrieve the data from the Odysseys.

C. Software

The software stack for our system can be divided into three layers:

- Linux was used for the OS;
- The Sensor Abstraction Layer;
- The infrastructure for storing, processing, and visualisation in the cloud[1];

The Linux distribution was a customised image compiled using the Overo branch of the Gumstix-oe clone of the OpenEmbedded repository[2]. We based our image on the omap3-console-image, tuned to use the power management kernel and the various pieces of software we required e.g., Java Virtual Machines (JVMs), Sensor Abstraction Layer (SAL), etc. There are two JVM distributions available for OpenEmbedded that we able to run SAL with. Firstly the Cacao distribution, which uses the GNU Classpath core class libraries. Secondly the OpenJDK distribution, which consists of three VMs coupled with an open sourced version of the original Java core class libraries. These VMs are Zero – an assembly free implementation, Shark – a VM containing a Just In Time (JIT) compiler, and Cacao – the Cacao VM coupled with the OpenJDK libraries.

The Sensor Abstraction Layer (SAL) [11] is a Java based middleware that provides plug and play access to heterogeneous sensors via plug-ins. SAL's ability to provide plug and play functionality is constrained to the capabilities of the underlying hardware. SAL can be deployed in a number of different configurations:

- As a standalone service;
- Embedded into other middleware;
- Exposed to the network as a Remote Procedure Call (RPC) service;

As a standalone service, SAL can be provided with a work flow containing the actions to be performed on the sensors and write the results to disk. Using SAL in an embedded or RPC configuration provides access to all available sensor via an Application Program Interface (API). With SAL's flexibility of deployment and pluggable sensor infrastructure, SAL seeks to reduce unnecessary duplication required by middleware authors to manage heterogeneous sensors.

[1]The third layer, the services that reside in the cloud, fall beyond the scope of this paper.

[2]http://gitorious.org/gumstix-oe

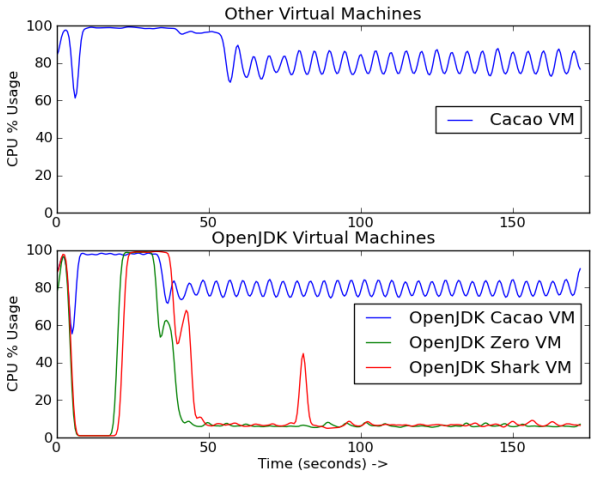

Fig. 4. CPU Utilisation of various Java Virtual Machines on the Gumstix Overo.

D. Duty Cycle

As detailed in section II-B our WSN nodes were powered by a 10 watt solar panel. A duty cycle was used to ensure that the node's power usage fell within the power budget. This duty cycle consisted of the WSN nodes running in suspended mode for most of the time and waking up at 2 hour intervals between the hours of 10AM and 4PM and once at midnight. Taking the advice of Barrenetxea *et.al.*, "You should have as much remote control facilities on your deployments as possible." [12]; we also scheduled a 1 hour maintenance window to allow us to reconfigure the nodes if necessary.

III. Results

Every aspect of the design of a WSN node has an impact on its power budget. Figures 5 and 6 display the power usage of our WSN node. Figure 5 shows the battery voltage and the power flow for a typical sunny day with a shady afternoon and figure 6 shows the power usage of a single duty cycle at midnight. Each node uses 1 watt of power when suspended (which is effectively wasted) and has a peak power usage of 5.6 watts. The provisioning of the duty cycle also needs to take into account the variability of solar power due to cloud cover etc. The large amount of wasted power and

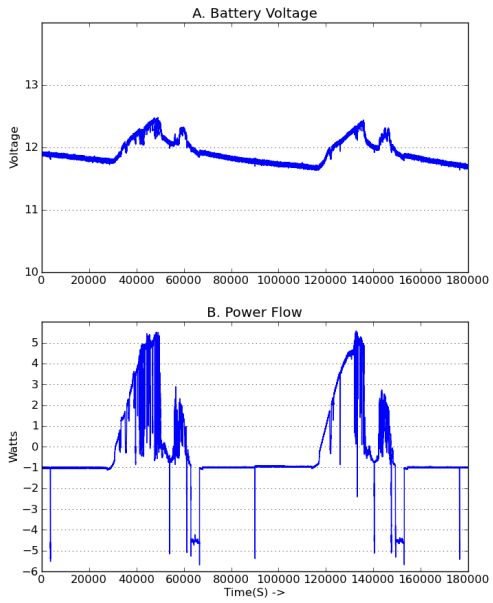

Fig. 5. Battery Voltage (A) and Power Usage (B) for 2 days. Positive values charge the battery, negative values discharge the battery.

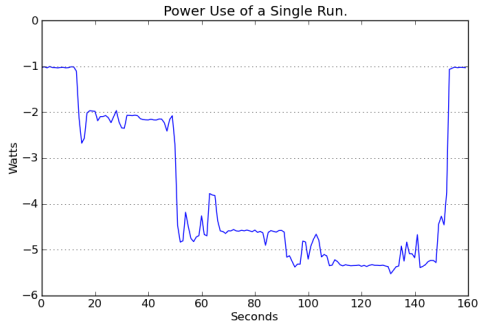

Fig. 6. The power usage of a single run, from suspend to suspend.

considerations for the variability of the power supply produces a duty cycle with relatively few periods of uptime and requires a larger power supply/store then otherwise necessary. This duty cycle also makes autonomous re-taskable data loggers necessary so that data logging can continue when the node is asleep.

CPU usage of the WSN node is another element that has an impact on the power usage. SAL has the largest impact on the CPU utilisation of the WSN node, so it is important to select a JVM that is efficient as possible. Figure 4 displays the CPU usage for each VM/library combination whilst running the SAL RMIAgent. Both Cacao VMs can be immediately illuminated from consideration, leaving the remaining OpenJDK VMs – Zero, and Shark. We speculate that the poor performance of the Cacao VMs is an issue with that particular implementation. Of these remaining VMs, we went with the assumption that in the long run the VM with the JIT was going to perform better. Future work will investigate whether using the VM with a JIT actually leads to improved performance.

IV. Conclusions

Whilst presenting the expected issues with power management, commodity hardware and software stacks provide a great deal of flexibility to meet the demands of an evolving target environment. The flexibility of our approach was demonstrated when the changing communications range requirements made our existing hardware unsuitable, and all that was required was to find an appropriate replacement and plug it in. Even though this deployment only had five nodes, the number required for the scientific model in section II-A, this deployment demonstrated the viability of our approach.

The commodity nature of our approach means our system can be easily reconfigured to use alternate subsystems (*i.e.*, the power, communications, and sensing subsystems). Although we used Wi-Fi for communication in this deployment, our system can use alternatives such as 3G, 900 MHz radios, or satellite. Likewise, the Odyssey loggers can readily be replaced if the application calls for different sensors. Furthermore, the Linux environment makes a large array of software available to the platform and allows end users to leverage their existing skill sets to interact with, and manage the nodes.

Along with the reconfigurable nature of our system, one of the biggest benefits of our approach is its cost. Each of our deployed nodes cost approximately $2000, with the loggers accounting for 45% of this total.

This paper explored the utility and practicality of building WSNs based on commercially available embedded single board computing platforms using standard consumer operating systems. Our test bed was built using Gumstix computing platform, running a Linux OS with a Java-based middleware coupled to low-cost scientific grade sensors. Test deployments have found this to be a highly versatile solution, able to leverage the flexibility of commodity hardware and software while maintaining reasonable utility.

The current version of the SEMAT system places all the computational intelligence on floating surface buoys, with wired connections to sensors in the water column on the sea floor. The surface Linux-based nodes are

responsible for collecting sensed data, storing it locally, and transmitting results back to the land-based data store.

Future work includes investigating ways to further reduce the amount of wasted power. We propose doing this with a programmable power supply that can be configured when to turn the power on and off. We are also planning on investigating combining our current platform with low powered devices to create a high/low power hybrid system similar to the Gumsense project [10].

Our next version of the SEMAT system adds low-power computational nodes below water. These low-power nodes act as underwater sensor hubs. Only a single cable to connect the surface buoy and the underwater hub is required. Furthermore, reconfiguration of the sensors only requires new connections to be made to the underwater hub.

Like the surface buoy, the underwater hub and sensor interfaces are based on readily available, open source commodity hardware and software. The underwater hub hardware is based on the Arduino microprocessor platform, and the software is based on the TinyOS system [13]. The underwater hub will be able to switch on and make regular data measurements using considerably less power compared to using the surface node. The surface node does not need to be powered up for each measurement (perhaps at intervals of minutes), only for each data download (perhaps at intervals of hours). The final planned enhancement to the system would be to replace the connection between underwater sensor hub and underwater sensors with a short-range (a few mm) wireless connector using inductive power transfer and Zigbee data transfer. This would allow sensors or data stores to be changed underwater, greatly easing the burden of servicing longer-term deployments.

ACKNOWLEDGEMENT

This work was supported in part by the Queensland Government National and International Research Alliances Program. The authors would also like to thank Stuart Kininmonth from the Australian Institute of Marine Science (AIMS) for his help and guidance, and AIMS for the use of some of their infrastructure.

REFERENCES

[1] Y. Liu, D. Hill, A. Rodriguez, L. Marini, R. Kooper, J. Myers, X. Wu, and B. Minsker, "A new framework for on-demand virtualization, repurposing and fusion of heterogeneous sensors," *2009 International Symposium on Collaborative Technologies and Systems, CTS 2009*, pp. 54–63, 2009.

[2] D. Butler, "2020 computing: Everything, everywhere," *Nature*, vol. 440, no. 7083, pp. 402–405, 2006.

[3] G. Simon, G. Balogh, G. Pap, M. Maróti, B. Kusy, J. Sallai, Á. Lédeczi, A. Nádas, and K. Frampton, "Sensor network-based countersniper system," in *SenSys'04 - Proceedings of the Second International Conference on Embedded Networked Sensor Systems*, ser. SenSys'04 - Proceedings of the Second International Conference on Embedded Networked Sensor Systems, Baltimore, MD, Mar 2004, pp. 1–12.

[4] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, no. 2, pp. 18–25, 2006.

[5] L. Schwiebert, S. K. S. Gupta, and J. Weinmann, "Research challenges in wireless networks of biomedical sensors," in *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, ser. 7th Annual International Conference on Mobile Computing and Networking, Rome, 16 July 2001 through 21 July 2001 2001, pp. 151–165.

[6] F. Stajano, N. Hoult, I. Wassell, P. Bennett, C. Middleton, and K. Soga, "Smart bridges, smart tunnels: Transforming wireless sensor networks from research prototypes into robust engineering infrastructure," *Ad Hoc Networks*, vol. 8, no. 8, pp. 872–888, 2010.

[7] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, ser. Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, 28 September 2002 through 28 September 2002 2002, pp. 88–97.

[8] J. Trevathan, I. Atkinson, W. Read, N. Bajema, Y. J. Lee, and R. Johnstone, "Developing low-cost intelligent wireless sensor networks for aquatic environments," in *Proceedings of the 2010 Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, ser. Proceedings of the 2010 Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, S. Marusic, M. Palaniswami, J. Gubbi, and P. Corke, Eds. Australia: IEEE, December 2010, pp. 13–18.

[9] M. Hempstead, M. J. Lyons, D. Brooks, and G.-Y. Wei, "Survey of hardware systems for wireless sensor networks," *Journal of Low Power Electronics*, vol. 4, no. 1, pp. 11–20, 2008.

[10] K. Martinez, P. Basford, J. Ellul, and R. Spanton, "Gumsense - a high power low power sensor node," in *6th European Conference on Wireless Sensor Networks*, January 2009.

[11] G. Gigan and I. Atkinson, "Sensor abstraction layer: A unique software interface to effectively manage sensor networks," in *Proceedings of the 2007 International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP*, ser. 2007 International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP, Melbourne, VIC, Mar 2007, pp. 479–484.

[12] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli, "The hitchhiker's guide to successful wireless sensor network deployments," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 43–56.

[13] N. W. Bergmann, M. Wallace, and E. Calia, "Low cost prototyping system for sensor networks," in *Proceedings of the 2010 6th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2010*, ser. 2010 6th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2010, Brisbane, QLD, Jul 2010, pp. 19–24.