Wireless Sensor Network for Habitat Monitoring on Skomer Island

Tomasz Naumowicz*, Robin Freeman†‡, Holly Kirk†, Ben Dean†, Martin Calsyn‡, Achim Liers*, Alexander Braendle‡, Tim Guilford†, Jochen Schiller*

*Freie Universität Berlin, Institute of Computer Science, Germany
Email: {firstname.lastname}@fu-berlin.de
†University of Oxford, Department of Zoology, United Kingdom
Email: {firstname.lastname}@zoo.ox.ac.uk
‡Microsoft Research, Cambridge, United Kingdom
Email: {firstname.lastname}@microsoft.com

***Abstract*—In the natural sciences, research often relies on extensive manual investigation. Such methods can be error-prone and obviously don't scale well. The development of autonomous data acquisition systems such as Wireless Sensor Networks (WSN) has provided a method to significantly reduce manual work and, as such, has the potential to enable researchers to address previously infeasible scientific questions.**

However, making the transition from WSN deployments in a laboratory to real-world deployments is still very challenging. Creating robust, error-free systems that are able to run autonomously in real-world environments without manual supervision has proven to be complex and, therefore, the number of successful collaborations between computer scientists and natural scientists is still limited.

Here, we describe our successful attempt to design and deploy a WSN to monitor seabirds on Skomer Island, a UK National Nature Reserve. We summarize the evolution of the system over a period of three years, share insights on selected design decisions, and discuss both, our experience and the problems we have encountered.

***Index Terms*—Wireless Sensor Networks, Deployment**

I. Introduction

Research in Wireless Sensor Networks (WSN) has predominantly been focused on hardware design, self-organization, various routing algorithms, or energy saving patterns. Today, this trend is changing and an interest in real-world scenarios involving WSNs is evident. Many research groups have started to deploy WSNs in real-world environments, e.g. for structural monitoring [1], environmental monitoring [2], [3], [4] or habitat monitoring [5]. We can now begin to address the challenges arising from real-world deployments rather than only those from simulations or lab based experiments.

In this paper we present our deployments to Skomer Island in 2008 and 2009. We also discuss changes from our pilot solution deployed in 2007. It documents the hardware and software development during the system design and deployment phase. We share insights into the hardware platform used and the software solutions applied.

The principle contributions of this paper are the outcomes of our hardware design being made available[1] to the research community, discussion on the issues with generic hardware, and a suggestion of environmental sensors that we found to be reliable in field deployments. We also propose a straightforward firmware update solution based on storage card replacement, and share suggestions on useful improvements of typical watchdog and software timers implementation in custom firmware.

A. Skomer Island

Skomer Island is a UK National Nature Reserve located off the west coast of Pembrokeshire, Wales. The University of Oxford, UK, has an existing research program on the island investigating the behavior and spatial ecology of the Manx Shearwater (*Puffinus puffinus*), a burrow-nesting, highly pelagic seabird that spends most of its life at sea. These birds rely on the ocean ecosystems to which they attend, and are sometimes referred to as integrators of oceanic resources. Their behavior can inform us both about the health of the ecosystems they inhabit and can also act as a model for the behavior of a variety of similar seabirds.

B. Evolution of the WSN Deployments on Skomer Island

The spatial behavior of Manx Shearwaters is being actively investigated using miniature GPS loggers (Guilford *et al.* in [6], [7]). This work relies on very intensive manual techniques. Researchers have to replace the GPS loggers on a regular basis in order to download the data and to charge the batteries. In the past they performed manual burrow inspections every 20-30 minutes to recapture tracked birds. This didn't scale well and limited the number of animals that could be feasibly tracked or monitored.

1) The deployment in 2007: The first WSN we deployed on Skomer Island in 2007 detected the birds' activity around entrances to the burrows and notified the researchers about the arrivals of Radio Frequency Identification (RFID) tagged individuals. This was the core feature of the deployed system

[1]http://skomerisland.codeplex.com

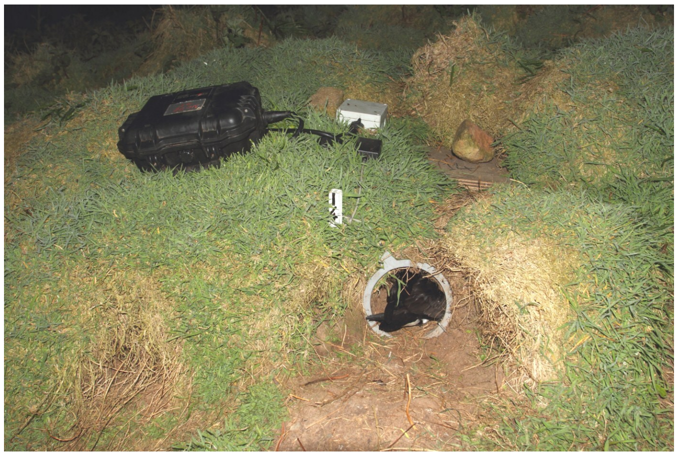

Fig. 1. The sensor node deployed on Skomer Island. The black waterproof case containing the device is visible on the left. The circular RFID antenna is installed outside of the burrow at its entrance and is visible in the middle of the picture where a bird is entering the burrow.

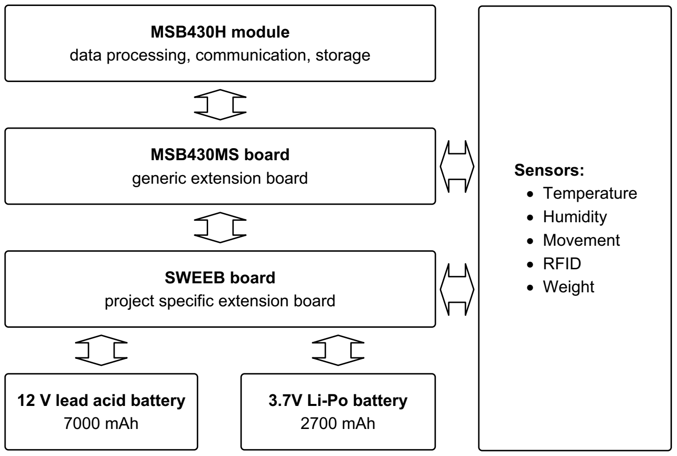

Fig. 2. Overview of components used to assemble a single sensor node deployed on Skomer Island.

and allowed to reduce the frequency of burrow inspections. Additionally, we configured this system to monitor the air temperature and humidity inside and outside of the burrows. The WSN was deployed from March 2007 until September 2007. One of the key reasons for this pilot deployment was to investigate the potential impact of such a system on the welfare of the animals being monitored. During the 2007 deployment we found no difference between the birds observed and controls in breeding success or fledging date. The design of the WSN and our experiences from the deployment are presented in [8].

2) The deployments in 2008 and 2009: During the subsequent expeditions in 2008 and 2009 we intended to focus mainly on improving the usability and reliability of the system. Some hardware aspects of the sensor nodes had to be updated as few problems were highlighted during field trials (e.g. we learned that the spring-loaded connectors we used to attach sensors were hard to use in poor weather conditions). We also had to develop a simple firmware update strategy to simplify future updates.

However, the goal was also set on expanding the sensing capabilities of the improved WSN. The updated WSN should additionally help answering questions about the amount of food being delivered to the burrows during the breeding season. This should be realised indirectly, by monitoring the weight of tagged birds entering and leaving the burrows. The new requirements were challenging and made it necessary to entirely redesign the system we used during the 2007 deployment.

II. System Overview

We deployed a WSN with 20 (March 2008 – September 2008) and 30 (March 2009 – September 2009) battery powered sensor nodes. There was a dedicated data sink on the network. In contradiction to the 2007 deployment, it wasn't relaying the collected data back to the mainland. We removed this feature and reduced the complexity of the system since researchers were present on the island during each expedition. The WSN spanned a mesh network with bi-directional communication support using the reactive routing Micro-Mesh-Protocol [9].

The devices were sealed in waterproof cases and placed next to the monitored burrows. Sensors were mounted at the burrow's entrance and attached to each unit using a single rugged connector (see Fig. 1). Each node was powered by a primary 3.7 V Lithium-ion polymer (Li-Po) battery with 2700 mAh capacity and was equipped with a secondary 12 V (7 Ah) lead acid battery. At each burrow, the following parameters were monitored:

- air temperature and humidity inside and outside of the burrow,
- movement at the entrance to the burrow measured using passive infrared (PIR) sensors ,
- identity of individual birds tagged with RFID tags that were entering the burrow (measured using a custom made RFID antenna),
- weight of individual birds as they were entering or leaving the burrow (measured using a custom made scale).

Each node logged the monitored data on its local SD card and transmitted selected data entries over the radio to the data sink. Node status parameters such as battery voltages or network connectivity status were monitored as well.

Figure 2 shows a overview of the components used to assemble a single sensor node. Selected components are discussed in the following sections.

III. Hardware Design

We used the Modular Sensor Board (MSB) platform [10] in our WSN deployments. The MSB platform was designed with focus on modularity. Its components can be stacked together enabling easier adaptation of the hardware to new requirements. The core of the platform are the MSB430 and MSB430H boards. They are both equipped with the essential components: the Texas Instruments MSP430F1612 microcontroller and a 868 MHz radio transceiver.

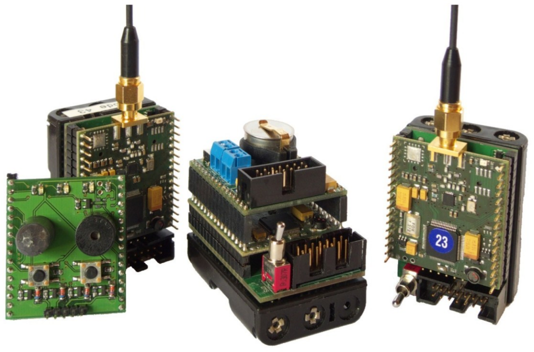

Fig. 3. Three MSB430H core modules mounted onto battery packs and equipped with two different, project specific extension boards (the antennas were cropped from the picture).

You can build project specific extension boards with dedicated sensors but reuse the same core module across multiple deployments. This saves development time and costs. Figure 3 shows three MSB430H core modules equipped with a battery pack and different sorts of extension boards.

We designed two extension boards during preparations for the Skomer 2008 and 2009 deployments: a generic one during the very early development stage, and another one which was tailored to the specific deployment requirements.

A. Core Module: MSB430H

During the 2007 deployment we used the MSB430 core module from the MSB platform. For the years 2008 and 2009 we decieded to use a updated revision of the device, the MSB430H core module.

The boards differ in the installed radio transceiver: the Chipcon CC1020 was replaced with a more recent Chipcon CC1100. The main advantages of the CC1100 over the CC1020 are its hardware support for packet handling and data buffering, integrated wake-on-radio functionality and support for higher data rates.

Support for packet handling and data buffering simplifies firmware development significantly. You don't need to continuously analyze the raw data streamed from the transceiver. Tasks like detecting the preamble of a radio packet, dealing with addressing, Forward Error Correction, or CRC computations are realized in hardware. If configured properly, the CC1100 will notify the microcontroller using a GPIO about buffered packet when its destination address matches the node's address. When the CC1100 is used, the microcontroller can be put in a low power mode for longer periods of time or execute other application specific tasks.

The integrated wake-on-radio functionality can be used to switch the CC1100 periodically between sleep and listening modes in specified intervals without any interaction of the microcontroller. The scheme implemented in hardware is similar to X-MAC [11] but in this solution not a short preamble but an entire packet is being transmitted in a burst.

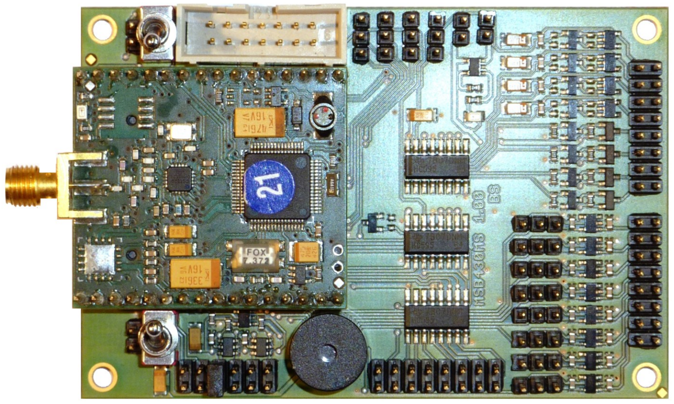

Fig. 4. The MSB430MS extension board with a MSB430H core module.

According to the data sheet, the CC1100 supports data rates of up to 500 kbit/s (the older CC1020 is rated with 153.6 kbit/s). The effective data rate strongly depends on the application. We used the following configuration optimized for low power consumption: data rate of 400 kbit/s, enabled hardware packet handling, and wake-on-radio support with 542 ms sleep intervals. The payload available for our application had the size of 50 bytes. After adding the headers, preamble and the sync word, 76 bytes per 50 byte long packet were transmitted. All those factors lead to net data rates in the range of 30 kbit/s.

The antennas were mounted inside of the waterproof case. We observed the expected degradation of the radio range to only 50 meters (the line of sight).

The raw data rate of 500 kbit/s found in the data sheet can be used as the net data rate only in the Continous Transmission Mode of the CC1100. In this mode the CC1100 behaves as a regular bit streaming transceiver.

The remaining specification of the MSB430H board didn't change. Both devices, the MSB430H and the MSB430, are driven by a low power Texas Instruments MSP430F1612 microcontroller run at 8 MHz and are equipped with an SD/MM memory card slot. Various digital and analogue sensors may be attached to 32 externally available add-on connectors. Those connectors are wired directly to the GPIO pins of the microcontroller. The power consumption of the MSB430H ranges between 250 μA and 115 mA depending on the application.

B. Generic Extension Board: MSB430MS

In most cases, the MSB core module needs to be equipped with an extension board. Especially, when you want to interface to external devices. We tried to minimize the complexity of this task and envisioned an extension board (MSB430MS) that could be reused across many deployments without modifications. In order to cover most of the use cases, we designed it in close collaboration with researchers at the University of Oxford who would be deploying and using the system in the field.

The schematics of the MSB430MS and hardware drivers are available for download as open source under the BSD license.

We hope that by doing this we can help others during their hardware design phase. Also in cases when the MSB platform isn't going to be used, one could reuse parts of the schematics in his own designs.

The MSB430MS board with a installed MSB430H module is shown in Figure 4. The board provides power for the MSB core module and is equipped with multiple connectors for attaching project specific hardware. The device is powered from a Li-Po battery. The Li-Po battery can be automatically charged from an external 5 V power supply. Voltages of the Li-Po battery and of the external power supply can be monitored in software.

The board is equipped with 8 digital outputs on TTL-level and 8 digital inputs. Those inputs can be either operated on TTL-levels or be regulated down from up to 12V. Power supply for up to 8 external devices can be controlled: 4 devices can use the regulated 3V power, the remaining 4 can use the unregulated external power. Furthermore, the MSB430MS supports four RS232 interfaces via a integrated multiplexer. It's also equipped with two 16 bit analog-to-digital (AD) converters. Power switch, reset switch, a JTAG connector, and a beeper are included as well.

Power supply of most of the components on the board can be individually controlled by the MSB core module, so only the required ones can be left enabled. Managing that many features by a limited number of available IOs was realized by a series of three shift registers.

The board has physical dimensions of 9 cm by 6 cm (3.5" by 2.4"). The interface pins were specified to be compatible with IDC sockets in order to make it easier for researchers to attach external devices. Unfortunately we didn't apply this rule to all interface pins. The IDC sockets couldn't be used with some connectors on the MSB430MS and we had to fall back to jumper wires.

C. Project Specific Extension Board for Skomer Island

Some deployment specific requirements couldn't be fulfilled by using only the generic MSB430MS board. Interfacing to specialized sensors turned out to be difficult, e.g. the Panasonic Passive Infrared Motion sensor (AMN33111J) we used required adding two dedicated 100 kΩ resistors. Another complex task was attaching the RFID reader which was selected at a later stage of the project. The reader required to be powered with 12V but the MSB430MS didn't support 12V power supply. Also, wiring external devices to many interface pins scattered all over the MSB430MS showed to be a very time consuming and error-prone task (particularly when items needed to be replaced in the field).

We decided to design and assemble a dedicated interface board comprising the additional components (codename SWEEB). During this process, we added the following features that weren't originally supported by the generic MSB430MS device:

- Voltage regulator that allowed us to use 12 V lead acid batteries as an energy source. High capacity and 12 V voltage were required to operate the RFID reader for longer periods of time.
- Power supply control for two external 12 V devices that we used to operate the RFID readers.
- Connector for self-powered Real Time Clock (RTC) from Maxim [12].
- Integrated socket used to attach all external sensors.
- Button that researchers used to suspend the data logging for safe storage card removal.

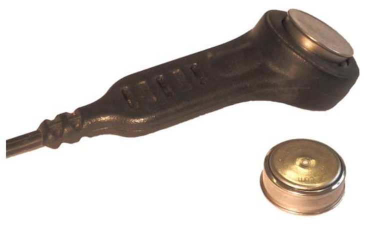
Fig. 5. Two Maxim DS1923 temperature and humidity loggers, stand-alone and in a mounting clip.

The new interface board and the MSB430MS board were wired together and enclosed in a waterproof case during the assembly of the device at the research station. To facilitate easy deployment and removal of devices in the field, a single external connector (Samtec SCR2 [13]) was used to attach the sensors.

D. Selection of the Environmental Sensors

Monitoring the basic environmental parameters namely temperature and humidity is a popular by-product of a WSN deployment. Usually sensors like Sensirion's SHT11 or SHT75 are used to accomplish this task. These are low-cost, readily available sensors that are relatively simple to interface with. However, the drawback of those devices is that they come unpackaged. You have to find a solution for waterproofing and mounting them properly. This process is very time-consuming and the results aren't always reliable as Barrenetxea *et al.* report in [14]. They invested some time in mounting and sealing the SHT75 sensors and still encountered failures of those sensors during deployment. Although we luckily didn't experience any issues with the SHT11 sensors during our Skomer 2007 expedition, we also spent a lot of time with packaging those devices. With this expirience, we decided to move to a more robust platform. Reports about possible complications with the platform we used previously, supported our decision.

We selected the DS1923 iButton from Maxim (Fig. 5). The DS1923 is a temperature and humidity sensor with integrated data logger support, 8 kB of non-volatile memory, and a independent power supply. It can be deployed in harsh environment immediately, since it's already enclosed in a waterproof steel can. Mounting clips for interfacing to it are available off the shelf as well. The higher unit costs are negligible if you consider the effort of mounting and waterproofing that isn't required anymore.

iButton sensors communicate using the 1-wire protocol. This protocol supports bus topology and allows you to add or remove devices on the fly without modifications of the software. It's important to note, that the 1-wire protocol is very sensitive when it comes to timing. Your firmware needs to provide support for software timers with μs resolution. We experienced many problems with proper timing at that high resolution in the initial stages of the software development. Our delay function didn't always generate the delay we specified. Further investigation showed that the time required for a function call wasn't equal across changes in the software as new features were added and the program code was being relocated during the linking process. We solved this issue by keeping the entire 1-wire protocol related code close together in one section using the linker script.

We leveraged the integrated data logger support of the DS1923 in order to improve the reliability of the data collection. We setup the DS1923 devices to log air temperature and humidity every 10 minutes. The sensor nodes executed data download task periodically. This ensured that we wouldn't lose any sensor reading, even during down times of the sensor node.

E. *Weight Monitoring*

One of the goals of our recent WSN deployments was to investigate how much food a Manx Shearwater brings home and how much their weight varies over a nesting season by weighing them each time they enter or leave their burrows.

The central component of the weighting solution is a strain gauge from a retail consumer scale and a custom made amplifier board. We mounted the strain gauge inside of a 5" (12.7 cm) PVC tube and used it to measure the weight of a 4" (10.2 cm) tube nested inside this assembly. The gap between those two tubes at the beginning and the end was covered by a matched PVC shield. The output from the amplifier was sampled by the AD converter on the MSB430MS board at 8Hz with 16 bit resolution. We achieved a precision of about a gram with this setup.

In the field, the weighting tube was inserted into the burrow entrance such that the birds had to pass through it on each trip in or out of their burrow. Field tests with dummy tubes indicated that birds were comfortable moving through them and their natural behavior didn't appear affected by the dummies or the subsequently deployed devices.

Due to temperature dependency of strain gauges in general and because of dirt accumulating over time at the bottom of the inner PVC tube, continual weight measurements were recorded, even when no birds were present. This helped us to collect reference data that can be used as a reference during data evaluation. Researchers performed reference weight measurements during periodic burrow maintenance. They used reference weights of 100 g and 500 g. The results of those measurements were used as additional reference points during data evaluation.

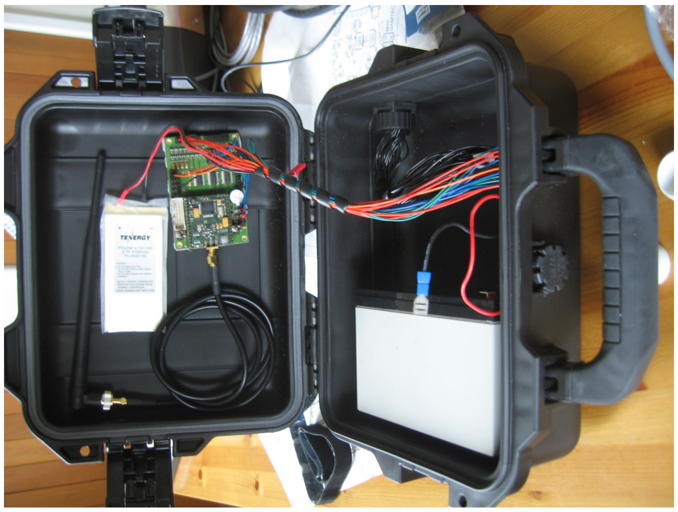

Fig. 6. The IP67-rated sensor housing with installed hardware modules and batteries: the Li-Po battery to power the microcontroller, and the 12V lead acid battery to power the RFID reader and charge the Li-Po as needed.

F. *Waterproofing*

During the first deployment on Skomer Island in 2007, short PVC tubes with caps on top were used as sensor housing. We didn't experience any issues caused by this simple waterproofing but maintaining the devices was very troublesome [8]: devices were hard to remove from housings, batteries changing was complex, and color-coded cables that we used to attach individual sensors were hard to identify during a night shift in the field.

Based on our experiences and reports from the research community [14] we moved to a more robust solution and decided to use IP67-rated packaging. IP Rating stands for *Ingress Protection Rating*. It's used to describe the level of protection against dust and water provided by the rated packaging. The IP67-rated Hardigg StormCase iM2050 we selected, provides complete dust protection and water protection up to immersion of one meter.

The core module with the extension board, the interface board, and a battery were closed inside of the case in the research station (see Fig. 6). The case was not intended to be open in the field at all. Sensors were connected to the device with one rugged 32-way SCR2 socket from Samtec [13]. The node was able to detect the event of connecting and disconnecting the socket and reacted respectively by enabling or suspending the data acquisition. In the case a node needed to be serviced, the entire unit could be quickly disconnected and replaced.

IV. Software Design

The WSN ran on a custom built firmware that has its roots in the ScatterWeb [15] firmware from the Freie Universität Berlin. The firmware supports cooperative multitasking and provides a low level API for a software based RTC, software timers, low power sleep modes, and UART communication with local and remote commands support. Micro-Mesh-

Protocol [9] for networking and SD card drivers with FAT16 support for data storage are included as well.

We developed drivers for the MSB430MS and SWEEB extension boards, for the 1-wire bus protocol, and for the 1-wire devices DS1904 and DS1923. Our implementation is based on a public domain reference library from Maxim.

We also added support for emulated UART RX functionality. The RFID reader we used in the project transmitted detected tags over the UART interface. In order to reliably read all detected tags, the UART interface of the microcontroller must be configured in RX mode at all times. This wasn't possible because on the MSB430H board the same interface is shared between the UART communication ports and the SPI ports used to communicate with the SD card. Every time a log entry had to be written to the SD card, the firmware reconfigured the USART to handle the SPI communication and switched back into the UART mode afterwards. All RFID tags transmitted by the reader in the meantime were lost.

We implemented the emulated UART RX functionality in software by levaraging a hardware timer. This made the RFID tag detection more reliable and independent of SD card access frequency. Our implementation could be easily extended to support receiving data from up to 8 UART devices in parallel (where the MSP430 microcontrollers support using only up to 2 UART devices in parallel).

In the following subsections, we present few selected aspects of the firmware developed during the work on the Skomer Island deployments.

A. Software Updates

During the initial phase of our deployments, firmware updates had to be rolled out often, quickly and reliable. We didn't support over the radio update functionality and updates had to be carried out manually.

We designed and implemented a bootloader that was preinstalled on every deployed MSB430H board. The bootloader supports FAT16 and is able to flash the node on demand with any application that was stored on the supplied SD card. The bootloader was to be configured with help of a simple text based configuration file that can be also edited remotely and contains the name of the application requested to be installed.

During each power up of the device, the bootloader is executed first. It checks the state of the currently flashed application: faulty applications that didn't report their activity properly at runtime are replaced with a fall-back application called *gold image*. The *gold image* could be configured either as a previous and stable version of the deployed firmware or a specialized recovery application.

In the next step, the configuration file of the bootloader is read, the specified software update file is analysed, and finally its compile time is extracted. The extracted compile time is then compared with the compile time of the currently installed application. The boodloader installs the specified application only when the compile times differ. A full update of a 40 kB application takes about 5 seconds. After an update, the installed application is being initialized.

The software update workflow was very straightforward. Researchers had to prepare new SD cards with software updates and insert them into all deployed nodes. This workflow isn't comfortable for large networks, but it worked reliably and was easy to understand. We got positive response from the research team on Skomer Island. The process can be now easily extended to support over the air updates, e.g. by putting a specialized *Over-The-Air Update* application on the SD card and switching to it on demand.

The bootloader with FAT16 support occupies 6908 bytes at the beginning of the flash. This corresponds to 14 segments that are in use (in the last segment, only 4 bytes are unused). The bootloader is independent from the actual firmware distribution installed on the device and it could be even used to switch the node between different software platforms. Applications need to fulfil the following requirements in order to be supported by the bootloader:

- the compile time and the address of the entry point of the application need to be stored at specified addresses,
- the application can't use the flash memory area reserved for the bootloader.

B. Working with the Watchdog

The hardware watchdog is generally used to perform a controlled system restart after a software failure. This feature helps you to protect your programs against unintended application hangs. Unfortunately, in the default configuration, the crucial information about the location of the software failure is being lost, although it could be recovered in most cases. We present few simple steps that could be applied in any WSN application in order to extend the watchdog functionality:

1) First, an area in RAM needs to be protected from being erased during application initialization. The easiest way to achieve that is to use the mspgcc's RESERVE_RAM macro. This allows you to reserve RAM area above the stack and pass data between the instances of your application.
2) The next step is to configure the watchdog timer to jump into a Interrupt Service Routine (ISR) when it expires, and not to reset the device.
3) Inside of the ISR, the value of the Program Counter that was preserved prior to the ISR call (accessed with the GET_FRAME_ADDR marcro) needs to be saved to the reserved RAM area and finally, the device should be restarted.
4) After a reset, you can access the value of Program Counter in the reserved RAM area and e.g. log it.

The value of the Program Counter and the related map file let you easily identify the function that was interrupted by the watchdog and was making your application unstable. You can extend this feature and also save information about known reset reasons. This lets you distinguish between resets your application requested on purpose and resets that happened due to application hang. This helped us e.g. to resolve blocking issues during development of hardware drivers.

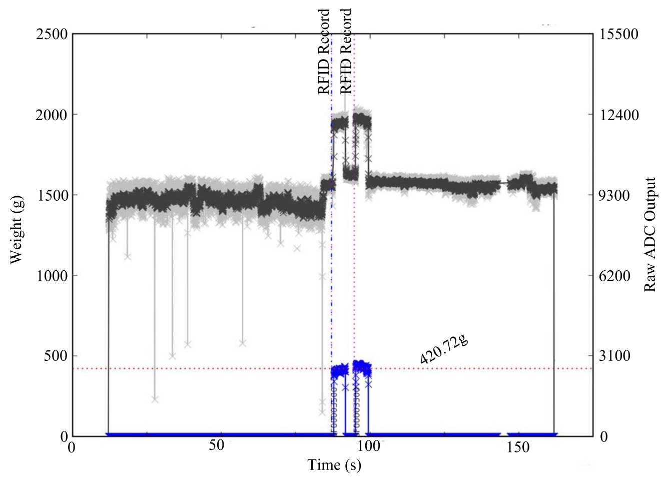

Fig. 7. Sample data collected during weight and RFID monitoring.

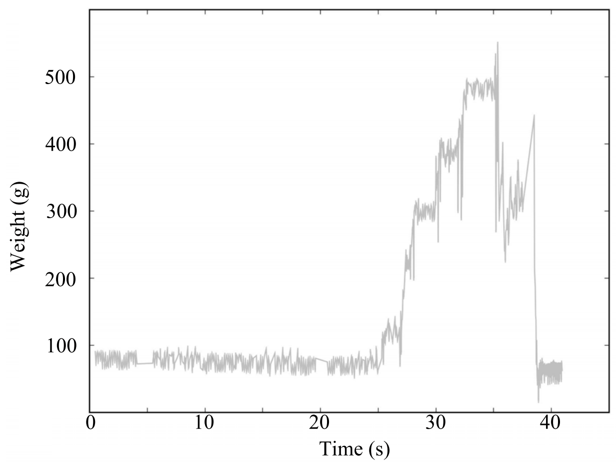

Fig. 8. Sample weight monitoring data.

C. Improving the Timers Support

Most WSN applications rely on software timer support. This essential feature is not only required to perform periodic tasks but also to drive the software RTC. Software timers are usually implemented using a simple low level ISR which generates periodic signals within the firmware. Although this design pattern is very straightforward to implement, it's not optimal for low power applications. It causes the microcontroller to activate from a low power mode up to 1000 times per second, depending on the resolution of the RTC. Surprisingly, even some established WSN operating systems still rely on this strategy.

In order to reduce the power consumption of the system we implemented a ”tickless” timer solution. It leverages the integrated features of the hardware timer. Detection of the hardware timer counter (TAR) overflow that occurs only every 8 seconds in our configuration is used to update the state of the software RTC and to update the list of registered software timer callbacks. One of the spare compare registers is used to process upcoming software timers that are scheduled to execute in less than 8 seconds. The address of the registered callback function is loaded withing the ISR and then, the callback function is called.

Will *et al.* [16] proposed a more sophisticated solution where the address of the registered callback function is stored directly in the Interrupt Vector Table. Their approach avoids unnecessary delays but it also limits the number of programmable software timers. Each programmable software timer is assigned to a timer compare register in hardware. You can't register more software timers than the number of available hardware compare registers using their implementation.

V. Results

Over the course of the 2009 season, a number of attempts were made to assess the reliability of weight measurements of returning birds. Test weights in the laboratory proved reliable (about 1g accuracy) and subsequent tests in the field with reference weights were also reliable. However, a number of problems remain with attempts to weigh individual birds: the birds' rate of movement through the tube, which sometimes is too fast to produce an accurate reading, and the retention of dirt within the tube which can effect weight estimate. Fig. 7 shows an example weighing event demonstrating both the measured baseline weight (and it's variation over time) and two extracted weighing events and their estimated values. RFID detection are also made corresponding to both weighing events. Fig. 8 shows a more complex weighing event where the actual weight of the visiting bird is very hard to determine.

Over the course of field trials, the reliability of devices has increased significantly, accurately recording both RFID detection and recording weight data (even when the signal is erroneous due to the practical problems described above). However, there still remain some issues with long term deployments - the devices are still power hungry (especially because of the RFID reader) and need manual attention to enable long term recordings (to change batteries, update firmware). A significant intended improvement is to deploy solar panels to enable long term, autonomous deployments.

VI. Lessons Learned

A. Usability of the WSN

In comparison with our deployment in 2007 we managed to significantly improve the usability of the devices. The support for the simple way of updating firmware by replacing an SD card or the improved sensor node packaging with only one rugged connector were received well by the researchers. It allowed them to quickly and easily replace nodes in the field. Even in conditions where devices are intended to be left in the field for long periods, it is important to facilitate maintenance to prevent errors and malfunctions in challenging environments.

B. Difficulties with Generic Hardware

We observed that the MSB430MS generic extension board wasn't sufficient to cover all deployment specific requirements. We needed to build an additional interface board. This was only a small device but it did cost us additional development time. We realized that this may also occur in upcoming deployments as well and thus came to the conclusion that it would be easier to build a tailored extension board in future projects. We are going to use the MSB430MS board schematics as the starting point and tailor it to the specific requirements. The low cost of prototype circuit board manufacturing and assembly also suggest that such project specific hardware will become more prevalent.

C. Firmware Selection

We used a local WSN firmware distribution from the Freie Universität Berlin in our project. While we were able to adapt it completely to our needs, we found that we had to implement some features that may have been available in other community driven efforts. Also, we would have possibly contributed some of the implemented extensions back to the community. Careful consideration should be made at the design stage to assess the advantages and disadvantages of different available distributions.

VII. Conclusions

Over the duration of the two expeditions to Skomer Island we were able to improve the reliability of the data collection. In the initial phase we had to redesign the previously established WSN completely in order to address hardware design problems and to add new features. Subsequently, we mainly focused on the quality of the weight measurement data we were gathering, which was influenced by the variations of the ambient temperature and the dirt being accumulated inside of the weighting device.

To aid further research, we decided to make the schematics of the MSB430MS board, the hardware drivers, the bootloader, and our firmware available for download as open source under the BSD license. All materials can be dowloaded from the http://skomerisland.codeplex.com website.

As in any WSN deployment, the configuration details of the network changed often with changing field conditions. We were updating the applications each time, but high WSN complexity prevented researchers from modifying the firmware themselves. Depending on the computer science team was a big disadvantage - it resulted in delays and also distracted researchers from the core goals of their work. Currently, we are investigating alternative ways of programming WSNs. We designed and prototyped a software factory that hides the complexity of software development for embedded systems [17]. The software factory will be evaluated by the research team during the next expedition to Skomer Island.

Acknowledgments

We thank the Wildlife Trust of South and West Wales, the Countryside Council for Wales, Juan Brown and Jo Millborough and many other staff of Skomer Island National Nature Reserve. We also thank D. Boyle and Prof. C.M. Perrins from the Edward Gray Institute for their great help throughout the project.

This work has been supported by the Computational Science Laboratory at Microsoft Research and the Natural Environment Research Council.

References

[1] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon, "Monitoring heritage buildings with wireless sensor networks: The Torre Aquila deployment," in *The 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, San Francisco, CA, US, 04 2009.

[2] K. Szlavecz, A. Terzis, S. Ozer, R. Musaloiu-Elefteri, J. Cogan, S. Small, R. C. Burns, J. Gray, and A. S. Szalay, "Life under your feet: An end-to-end soil ecology sensor network, database, web server, and analysis service," *CoRR*, vol. abs/cs/0701170, 2007.

[3] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter, "Luster: wireless sensor network for environmental research," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2007, pp. 103–116.

[4] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin, "Permasense: investigating permafrost with a wsn in the swiss alps," in *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*. New York, NY, USA: ACM, 2007, pp. 8–12.

[5] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a sensor network expedition," in *Proceedings of the European Workshop on Wireless Sensor Networks*, 2004, pp. 307–322.

[6] T. Guilford, J. Meade, R. Freeman, D. Biro, T. Evans, F. Bonadonna, D. Boyle, S. Roberts, and C. M. Perrins, "GPS tracking of the foraging movements of Manx Shearwaters puffinus puffinus breeding on Skomer Island," *Ibis*, vol. 150, no. 3, pp. 462–473, 2008.

[7] T. Guilford, J. Meade, J. Willis, R. Phillips, D. Boyle, S. Roberts, M. Collett, R. Freeman, and C. Perrins, "Migration and stopover in a small pelagic seabird, the Manx shearwater Puffinus puffinus: insights from machine learning," *Proceedings of the Royal Society B: Biological Sciences*, vol. 276, no. 1660, pp. 1215–1223, 2009.

[8] T. Naumowicz, R. Freeman, A. Heil, M. Calsyn, E. Hellmich, A. Brändle, T. Guilford, and J. Schiller, "Autonomous monitoring of vulnerable habitats using a wireless sensor network," in *REALWSN '08: Proceedings of the workshop on Real-world wireless sensor networks*. New York, NY, USA: ACM, 2008, pp. 51–55.

[9] N. Dziengel, M. Ziegert, Z. Kasmi, F. Hermans, S. Adler, G. Wittenburg, and J. Schiller, "A platform for distributed event detection in wireless sensor networks," in *First International Workshop on Networks of Cooperating Objects*, 2010.

[10] M. Baar, E. Köppe, A. Liers, and J. Schiller, "Poster and abstract: The ScatterWeb MSB-430 platform for wireless sensor networks," in *SICS Contiki Hands-On Workshop*, Kista, Sweden, 03 2007, p. 3.

[11] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks," in *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2006, pp. 307–320.

[12] *Maxim*. [Online]. Available: www.maxim-ic.com

[13] *Samtec*. [Online]. Available: http://www.samtec.com

[14] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli, "The Hitchhiker's Guide to Successful Wireless Sensor Network Deployments," in *The 6th ACM Conference on Embedded Networked Sensor Systems (SenSys 2008)*, 2008, pp. 43–56.

[15] *ScatterWeb*. [Online]. Available: http://scatterweb.mi.fu-berlin.de

[16] H. Will, K. Schleiser, and J. H. Schiller, "A real-time kernel for wireless sensor networks employed in rescue scenarios," in *LCN*. IEEE, 2009, pp. 834–841.

[17] T. Naumowicz, B. Schröter, and J. Schiller, "Prototyping a software factory for wireless sensor networks," in *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2009, pp. 369–370.