Hardware Design Experiences in ZebraNet

Pei Zhang, Christopher M. Sadler, Stephen A. Lyon, and Margaret Martonosi

Department of Electrical Engineering
Princeton University
{peizhang, csadler, lyon, mrm}@princeton.edu

ABSTRACT

The enormous potential for wireless sensor networks to make a positive impact on our society has spawned a great deal of research on the topic, and this research is now producing environment-ready systems. Current technology limits coupled with widely-varying application requirements lead to a diversity of hardware platforms for different portions of the design space. In addition, the unique energy and reliability constraints of a system that must function for months at a time without human intervention mean that demands on sensor network hardware are different from the demands on standard integrated circuits. This paper describes our experiences designing sensor nodes and low level software to control them.

In the ZebraNet system we use GPS technology to record fine-grained position data in order to track long term animal migrations [14]. The ZebraNet hardware is composed of a 16-bit TI microcontroller, 4 Mbits of off-chip flash memory, a 900 MHz radio, and a low-power GPS chip. In this paper, we discuss our techniques for devising efficient power supplies for sensor networks, methods of managing the energy consumption of the nodes, and methods of managing the peripheral devices including the radio, flash, and sensors. We conclude by evaluating the design of the ZebraNet nodes and discussing how it can be improved. Our lessons learned in developing this hardware can be useful both in designing future sensor nodes and in using them in real systems.

Categories and Subject Descriptors

B.0 [**Hardware**]: General; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.3 [**Special Purpose and Application-Based Systems**]: [Real-time and embedded systems]

General Terms

Performance, Design, Experimentation, Verification

Keywords

Sensor Networks, Sensor Deployment, ZebraNet, GPS

SenSys'04, November 3–5, 2004, Baltimore, Maryland, USA.

1. INTRODUCTION

Many uses for energy-aware sensor networks are quickly emerging from industry and academia. The numerous military and civilian applications of this technology have the potential to make a considerable impact on society. As a result, the sensor network research community has grown steadily over the past few years.

Typically, sensor nodes are comprised of sensors to collect environmental data, a low-power microcontroller to process information and control the system, a radio with limited range to transmit data from node to node, and off-chip memory to store data. The nodes are characterized by their small size and significant energy/resource constraints. In addition, once deployed, the nodes are difficult to retrieve, so they must function for months at a time on a battery pack. Also, the nodes need to be extremely durable as they will be released into potentially harsh environments. These issues affect all areas of hardware design, from the selection of the microcontroller to the design of power supplies.

Due to these fundamental constraints, hardware design guidelines for sensor nodes are different than those for other applications. While theoretical research on sensor network is ongoing, academic and industry researchers have increasingly deployed working prototype sensor networks on which theories can become reality. As part of the ZebraNet project, we have designed and built sensor hardware and software as well. ZebraNet is a mobile sensor system with sparser network coverage and higher-energy sensors (GPS) than many other sensor networks. As such, it operates in a distinct part of the hardware design space.

This paper discusses hardware design issues for sensor networks in general and our experiences with ZebraNet in particular. Through this paper we contribute both specific design thoughts as well as more general lessons learned. For example, while radio communication garners much attention in the sensor networks community, the less-glamorous topic of power supply design ended up requiring much more of our design time. This seems likely to be a general characteristic of many sensor network nodes because they can experience such large (relative) current fluctuations depending on which sensors and communication devices are on or off.

We also discuss our choice of a dedicated 16-bit microcontroller for system and protocol operation, rather than scavenging cycles from the GPS processor or adopting a simpler (e.g. 8-bit) CPU. Having a dedicated microcontroller eased software development and allows us to move cleanly to other non-GPS-based sensor applications. Furthermore, despite

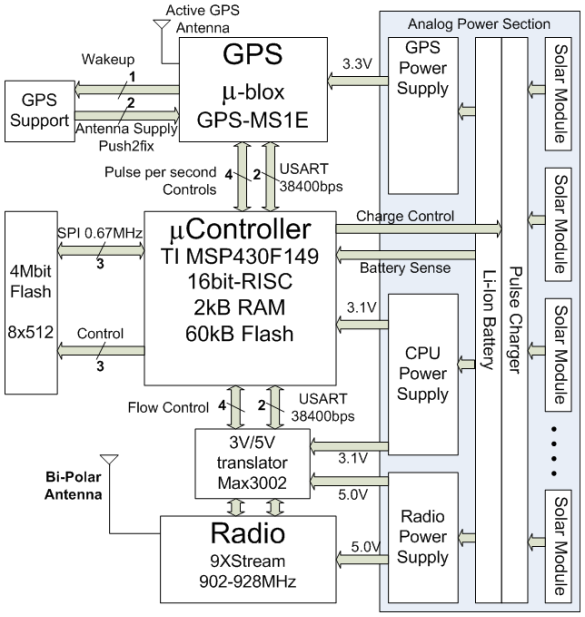

Figure 1: Block diagram of components, interface connections, and power supplies for the ZebraNet hardware version 3.

adding to our chip count, it reduces energy consumption by offering more efficient low-power modes.

Overall, for sensor networks to move from the theoretical realm to widespread deployment and real-world everyday use, sensor network researchers must share and learn from the design experiences of deployments along the way. Through this paper, we hope to offer a few of our lessons learned on a test deployment of mobile GPS sensors for wildlife tracking.

The remainder of this paper is structured as follows. Section 2 describes an overview of the ZebraNet system. Section 3 discusses microcontrollers for sensor nodes. In Section 4, we discuss peripheral devices such as sensors, off-chip memory, and the radio. Section 5 discusses hardware and firmware energy management techniques and design choices based on them. We provide experimental evaluations of our design in Section 6. Section 7 discusses our initial system deployment. Finally, we present related work in Section 8 and our conclusions in Section 9.

2. ZEBRANET SYSTEM OVERVIEW

The ZebraNet system, shown as a block diagram in Figure 1 and conceptually in Figure 2, is designed to support the research of biologists stationed at the Mpala Research Center in Kenya [25] by applying the latest sensor network technology to the field of animal tracking. ZebraNet consists of sensor nodes built into collars on zebras which take positional readings using a GPS unit and propagate them from zebra to zebra until infrequent communications percolate data to base stations [14].

Our system's three primary goals are to generate detailed, accurate logs of each zebra's position, to recover those logs for analysis, and to run autonomously for months at a time.

Detailed, Accurate Position Logs Ultimately, our positional logs need to provide the biologists with an accurate view into the daily migration pattern of a set of zebras.

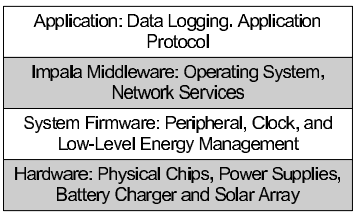

Figure 2: ZebraNet System Structure Overview.

Biologists' observations of zebra behavior show that taking readings every eight minutes provides them with enough samples to achieve this goal, but any interval longer than that is unacceptable.

High Data Recovery Rate Once the data samples are acquired, they must be collected for analysis. Given that the zebras are fairly mobile, that only a sparse few are collared and that they are spread over kilometer distances, we expect our collars to form an extremely sparse network, complicating the collection process. To move data through the network, nodes communicate via pairwise connections. In this system, latency is less important than eventually getting the position logs back to the biologists so the single-hop transmissions are sufficient to move the data between collars. Pairwise communications are also more reliable than multi-hop communications and this is important because the collars will only come into contact with each other occasionally. Periodically, a manned mobile base station will come into contact with a zebra and can download data from several zebras at once. We compensate for the high latency of data propagation by using a large flash memory to store position logs (see Section 4.2).

To maintain connectivity in such a sparse system, we design with radios with a range of over one kilometer. As a result, however, radio communications consume an order of magnitude more energy than the typical short range radios used in sensor networks. Figure 3 shows the power consumption when the system is running, including the period during which the radio turns on, sends a packet, and waits for a response from its neighbors until the radio communication period ends.

Autonomous Operation The collars must survive for months at a time in the wild with no human contact except for wireless interactions with the base station. This led us to add a rechargeable battery that scavenges energy through the use of solar cells. As a result our system can meet the survivability requirement, but our energy budget is strictly limited to the amount of energy that the solar cells can produce. The battery and solar cells are discussed further in Section 5.3.

The ZebraNet system is controlled by the Impala middleware layer [19][20]. Impala allows for a combination of scheduled and spontaneous events and in this implementation, GPS sensing and radio communication times are prescheduled. Several protocols are possible, but in our field deployment the information is propagated through the network using a flooding protocol. This allows the base station to receive the data from all of the collared zebras by encountering just one. In January 2004, we deployed seven nodes on zebras in Kenya. Based on the results from the first deployment of the system, we are currently making improvements

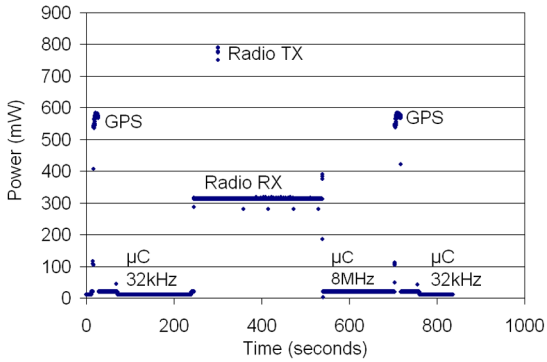

Figure 3: Power consumption of our system during a periodic data sampling and communication time measured with an oscilloscope. The GPS and Radio Transmit (Tx) points are the highest power. Radio receive (Rx) is moderate in its power consumption. The lowest-power portions are the microcontroller (μC) alone, running at either 32 kHz or 8 MHz. Every two hours when the radios active, the GPS reading is delayed four minutes so there is a 12 minute gap between readings in this figure.

with a broad-scale, long-term deployment in mind. More information about the deployment is provided in Section 7.

2.1 Hardware Evolution

Based on the system design parameters, the main design constraint is energy. ZebraNet nodes are built using low-cost, energy-efficient off-the-shelf components. From conception to realization, the ZebraNet node has gone through several design iterations. Table 1 shows the major changes between iterations of the design.

Version 0.1 was a proof-of-concept design [14]. In this version, we experimented with a number of design concepts including the use of a second, short-range radio for energy-efficient communications over distances of 100 meters or less. The system scavenged unused flash memory and CPU cycles from the GPS processor which reduced the overall board area, decreased latency by eliminating external wiring, and simplified the transfer of position logs from the GPS to long term storage. However using the GPS processor in this manner requires around 34 mA of energy even with the GPS functionality disabled [7]. Given our strict energy budget, our system required a lower energy solution, which came in the form of a separate microcontroller.

The major difference between Version 0.1 and Version 1 is the addition of a low-power microcontroller which controls all the peripherals and an external 2 Mbit flash memory to store data. These additions greatly simplify software development and ease design changes by allowing peripheral changes with only firmware level updates. Version 1 does not have the solar charging circuitry, but did include various exploratory designs of high-efficiency power supplies. The impact of these additions on the system performance and efficiency is presented in Section 5.

Version 2 is similar to Version 1, but the new layout only required half the board area of the old one. The design mostly explored issues with layout and cross-talk interference of on-board components. The power supplies were improved to reduce noise.

Figure 4: Photo of a ZebraNet node (2"x3"x1.25"). The left card includes the radio and the right card includes the GPS chip and the battery. A credit card is included in the picture for scale.

Version 3 is a complete system design and it was built into the collars that were deployed in central Kenya in January 2004. To complete the system, the battery was integrated with the on-board charger and solar cells. In addition, the power supplies were redesigned again to standardize the supplies for the GPS and the radio and to further reduce noise. A photograph of the most recent version of the sensing hardware is shown in Figure 4. It is powered using a Lithium Ion battery that is recharged with a solar array (described in Section 5.3). Since the energy provided by the solar array is limited, however, the system minimizes power consumption wherever possible. This leads directly into a discussion about the microcontroller, which is the primary topic of the next section.

3. THE MICROCONTROLLER

Choosing or designing a microcontroller for a sensor node leads to a basic tradeoff: any chip with such comprehensive control over a system should be fast and have plenty of memory, yet be energy-efficient. Since the power constraints typically dominate, this leads to significant computational and memory constraints.

In the earliest prototype of ZebraNet (Version 0.1) we ran the system off of the microcontroller co-located on the GPS chip, a 32-bit, 20 MHz Hitachi SH1 [14]. The package also has 1 MB of on-board flash memory of which 640 KB are designated for data storage. However, despite possible advantages in scavenging memory and CPU cycles from this chip, the advantages are outweighed by its 750 mA-hr per day energy requirement. In addition, the programming interface for the chip made implementing our software very complex.

Ultimately, given the application and the duty cycles of our sensing unit and the radio, we switched to a Texas Instruments MSP430F149 microcontroller [32]. This microcontroller manages the system operations and the peripherals. It is a 16-bit microcontroller with 2 KB of RAM and 60 KB of internal flash memory and two serial ports to communicate with the peripherals. A primary reason for selecting

	Version 0.1 [14] Aug. 2002	Version 1 Feb. 2003	Version 2 July 2003	Version 3 Nov. 2003
Power Supply	Off-board	Buck-boost and boost converters	Buck-boost and boost converters	Two buck-boost converters
Noise Reduction	Bypass Capacitor	Standard low ESR capacitors	Os-con capacitors	LC post filters and common mode choke
Radio	2 radio system	2.4 GHz	900 MHz	900 MHz
GPS	Cycle stealing from onboard CPU	Off-board antenna power	Ultra low noise linear regulator	Ultra low noise linear regulator
Data Flash	5 Mbit	2 Mbit	4 Mbit	4 Mbit
Battery charger	None	Off-board	Off-board	Pulse-charging
System Weight	1,151 grams	145 grams	136 grams	138 grams
Battery	N/A	Li-Ion, 2 A-hr 45 grams	Li-Ion, 2 A-hr 45 grams	Li-Ion, 2 A-hr 45 grams

Table 1: Design summary for different versions of ZebraNet hardware nodes.

this processor was that it supports multiple clocks. This allows us to use an 8 MHz clock while the GPS, radio, and flash modules are running. Otherwise, we switch to a 32 kHz clock.

As shown in Figure 1, the microcontroller communicates with the node's peripheral devices through the serial ports. The SPI and UART ports allow for synchronous and asynchronous serial communications with peripherals, respectively. Serial port constraints are a common limitation of energy-efficient processors. We would really like to have separate ports for the GPS, the radio, and the flash, but only two ports are available on the chip. To compensate, the GPS and the flash share a serial port, with the GPS utilizing the asynchronous connection to the microcontroller and the flash utilizing the synchronous connection. Although this setup prohibits the flash and the GPS from operating simultaneously, it allows the radio to use the flash while transmitting and receiving. This is important because it allows the node to minimize the amount of RAM necessary to complete the transaction; it reads data to be transmitted directly from the flash and writes data packets to the flash as they are received.

Given this configuration, the speed at which the microcontroller communicates with the peripherals is paramount. Radio and flash accesses typically have extreme time constraints, so delays can lead to lost information. For example, the radio only has a 132 byte buffer, so if the microcontroller cannot receive the information over the serial connection at least at the radio's over the air baud rate of 19.2 kbps, the radio buffer will overflow and data will be lost. Additionally, to conserve memory our software only allocates two 64 byte buffers in the microcontroller to receive data from the radio so if this data must be stored in long term memory, it must be transferred and written to the flash module at this rate too. Otherwise the data will be overwritten in RAM before it can be stored. These needs influence both the software design (a central focus of [11]) as well as hardware interface design, which is our central focus here.

The size of the registers must also be considered when choosing a microcontroller. For systems targeting sensors that require little processing and consume on the order of 1 mW of power or less, extremely low-power and narrow-bitwidth microcontrollers make sense. This is because in these situations, nodes can conserve energy while still meeting the system's performance requirements. But for systems characterized by high-power sensors and the need for high-burst-rate computations and communication, like ZebraNet, a more-capable microcontroller has proven useful. The ease of 16-bit addressing and computations has simplified our software development. Furthermore, properly supporting this notion of bursts of high-performance followed by extremely-low-energy ambient operation has led us to also begin researching custom sensor CPUs with very good performance-energy agility.

4. PERIPHERAL DEVICES

4.1 Radio

Our network is intended to be sparsely populated and highly mobile; nodes can go days or weeks without entering the radio range of another node. In addition, even if two nodes establish a connection, they may wander out of range during the transmission. Therefore, our system requires a radio range of 1-5 kilometers and a protocol that supports long blasts of data with occasional disconnections.

Since we needed kilometers of radio range, our high transmission power severely limited our selection of license-free operating frequencies. We chose the MaxStream 9XStream radio, which operates at 900 MHz and offers a specified range of "up to" 5 miles in ideal outdoor conditions [22]. Under normal operating conditions, however, the signal strength, and consequentially the effective radio range, is reduced by factors such as scattering, dispersion, and reflections. While the transmit power is only 21.5 dBm, the long range is possible since the radio can achieve a receiving side sensitivity down to -107 dBm. We also chose this radio because this transceiver module includes all the circuitry and logic needed to perform error checking and it provides a simple UART interface.

Our stringent energy budget also forces us to minimize the radio's duty cycle. Radio communications occur every two hours. Within the communication period, time-slotting reduces collisions among nodes. The GPS provides the real-time clock used for time-synchronization between the nodes, and the microcontroller keeps accurate time between GPS readings. As a result, we avoid the handshaking overhead associated with setting up a connection. In Version 0.1, we also experimented with a second, short range radio, but

given the sparse nature of our network, we determined that the extra board area and complexity was not worth the minimal benefit it provided.

Data is propagated through the system using a flooding protocol. Every two hours the nodes activate their radios, and search for other nodes in range. Each node that finds at least one neighbor transmits as many recorded positional logs as possible in its time slot. After five minutes, the communication session ends and the radio powers down. Any unsent data waits for the next communication interval. To improve communication efficiency, we are experimenting with various other protocols including a history-based protocol [19].

The radio transmits and receives data in packets of 64 bytes; each packet is designed to include a header, footer, and two GPS position samples. This design prevents position samples from being fragmented across multiple packets.

To further reduce power when the unit is not in use, the radio power supply can be shut off entirely, sinking less than 1 μA of current.

4.2 Off-chip Memory

Flash memory is ideal for sensor networks because it is non-volatile, fairly inexpensive, and compact. In our latest hardware design, we use the ATMEL 4-megabit AT45DB-041B DataFlash chip [1] to log sensed data. We chose this module because it is easily obtainable and has a small 8 mm * 5 mm footprint. The memory is divided into 2048 264-byte pages. The microcontroller communicates with the flash through the synchronous portion of the serial port it shares with the GPS at a rate of 667 Kbps. Although the flash module can operate at up to 20 MHz, this rate is fast enough to keep up with radio communications while reducing noise generated at high speeds.

There are three key issues associated with flash memory in general. First, pages in memory must be written and erased in their entirety. Erasing a page sets every bit in it high and write operations can only change high bits low. Therefore, a page must be erased before it can overwritten. Although writing to the same page twice without erasing is not recommended, in ZebraNet, a GPS position sample is only 28 bytes long, far less than the 264 byte page size. We get around this problem by writing 1's in all bit slots that we do not use. In the flash module, this is equivalent to telling it to do nothing to those bits. This means that we pass 264 bytes to the flash module to write 28 bytes of useful data. While this takes time, it does not directly affect performance because in our system, after a position fix there is a long idle period before the next sample. However, it could become a problem in a system in which sensors can generate data at any time and buffer size is limited.

Another problem with flash is that writes and erases are slow compared to other operations. For example, our Atmel chip requires 8 ms to erase a page and 14 ms to program one [1]. Therefore, erase operations need to be planned in advance and the write time must be accounted for in all algorithms that use the module.

Finally, flash memory modules have a limited write lifetime [15]. Given the high rate at which some sensor nodes can gather information, this constraint can become an issue in some applications. Our system achieves load-leveling by separating the flash into local and global data sections and treating each section as a circular buffer. Given the current system configuration, each node's flash memory is large enough to hold 26 days of its own positional data and 52 days worth of other node's data collected via collar-to-collar updates. Hence in our context, our system should be able to run for over 7000 years before the flash memory locations fail due to excessive writes!

4.3 Sensing Devices

The GPS unit is our primary sensing device and we use the μ–blox GPS-MS1E chip. While the specified power consumption of 462 mW is high, the GPS hot start time is only 2 seconds [7] so it was the best choice at the time. The low fix time minimizes the total energy consumed. Recently, Xemics introduced the lowest power GPS, the RGPSM002. The RGPSM002 has a hot start time of 12 seconds and power consumption of 62.7 mW [38]. Based on these specifications, this chip saves only 20 percent of the energy per fix than our current chip. However, through our tests we have found it necessary to keep our current GPS module on for an average of 25 seconds in order to improve positional accuracy, greatly increasing the energy savings. Therefore, we have begun to look at switching to the Xemics chip.

To meet the requirements of our system, we must take position samples every eight minutes. However, to conserve energy we would like to minimize the duty cycle of the GPS. When the GPS is turned on, it will report a reading every second and remains on for one minute or until it acquires an acceptable lock, whichever comes first. (With our GPS module, if the node cannot acquire a valid lock within one minute it probably will not acquire one soon at all.) The method for determining whether a lock is acceptable or not is discussed in more detail in Section 6.1.

The GPS communicates with the microcontroller at 38.4 Kbps—the maximum rate for the μ–blox chip—through the asynchronous portion of the serial port that it shares with the flash module. Due to port sharing, the GPS and the flash cannot operate simultaneously, so the position reading is stored in a buffer on the microcontroller until the GPS is shut down. The readings are processed by the microcontroller as they are received. As soon as the read is done, the system turns on the flash and records the data.

To conserve energy, the GPS chip runs off its own 3.3 V power supply which can be turned on and off by the software. Once either the GPS has been on for one minute or an acceptable lock has been obtained, the microcontroller cuts power to the GPS unit.

After selecting the sensing devices, the system designer must decide whether the microcontroller polls for sensor readings or the sensors wake the microcontroller when readings are available. This decision is primarily determined based on the duty cycle and energy consumption of the sensors. For example, a motion sensor might wait until it detects movement in the area before waking the microcontroller. While many sensors may work as the initiator, in our system the GPS consumes two orders of magnitude more energy than the microcontroller. Furthermore, it operates on a predetermined schedule. Therefore, in our case, it is much more energy-efficient to leave the processor on constantly and power down the GPS when it is not in use.

In the future we plan to include other sensors, including XYZ orientation sensors and acceleration sensors, to help the biologists better understand zebra behavior such as head positions.

5. ENERGY ISSUES

Since energy is the primary constraint of our system, efficient energy management is essential in every part of the system. In this section, we describe our techniques for energy management at the system level, our power supply design, and our solar charging mechanism.

5.1 System-Level Energy Management

There are three primary system level management techniques that we will describe in this section. They are the dual clock scheme, on-the-fly processing, and the timely use of components.

First, since the microcontroller is on constantly, the dual clock scheme saves significant energy. As mentioned earlier, the microcontroller supports two clocks. The hardware has both a 32 kHz clock and an 8 MHz clock. The system consumes twice as much energy when using the fast clock than when using the slow clock. Even in a system that requires data sampling as frequently as ours, the overall duty cycle is low so the fast clock is not always needed.

When the microcontroller is operating on the 32 kHz clock, it consumes approximately 0.05 mA more than it does in sleep mode. Since we use the on-chip A/D converter to control the battery charging mechanism and our peripherals operate on a fixed schedule, we decided use the slow clock as opposed to sleep mode for simplicity.

When peripherals are running, we switch to the 8 MHz clock. For the radio, we do not have a choice; the microcontroller running the 32 kHz clock cannot process data as fast as the radio receives it. With the GPS and the flash, though, the system could theoretically function on the slow clock. However, each of these peripherals consumes orders of magnitude more energy than the microcontroller running the fast clock. As a result, we conserve energy using the fast clock while these peripherals are running since it allows us to power them down earlier than if we used the slow clock.

Second, as originally mentioned in [11], concurrency-intensive operations are important in sensor nodes because of limited buffer space. However, concurrent operation also conserves power in our system in two ways: it uses some of the extra cycles the microcontroller spends waiting for bytes from the peripherals so that the processor can switch back to the slow clock earlier and it processes data before we need it so we can make faster decisions which ultimately leads to shorter peripheral on-time.

For example, as the position data comes from the GPS, low-level code extracts the current time as well as data to determine the accuracy of the position. If the lock is acceptable, the node will shut down the GPS unit and correct the system clock. This allows for a timely use of the GPS component and reduces the energy consumption by more than a factor of two compared to leaving the unit on for a minute and then turning it off.

Using the radio in a timely manner is also important. Since the GPS unit provides us with access to an accurate global clock, we can use low-level software to synchronize the nodes and use a time-slotted transmission scheme. This scheme eliminates the need for time synchronization communication rounds and saves power by eliminating a major cause of packet retransmissions.

5.2 Power Supplies

The demand for long life, small size, and performance has made power supplies a non-trivial part of our sensor node both in terms of complexity and design time. Since most sensor systems are designed with high performance and strict energy budgets, the power supply could determine the final competitiveness of the system. Furthermore, all sensor nodes require high amperage radio bursts balanced against low-power ambient operation. These large current fluctuations make voltage regulation a fundamental challenge for sensor hardware in general.

To minimize the power consumption and to maximize the performance of different peripheral devices, we designed separate power supplies for each module that are turned off independently of each other. The system is powered by a Li-ion battery, which can safely have voltages from 3.0 V to 4.2 V.

Linear Regulators We use linear converters to power our microcontroller and the GPS antenna. The TI MSP430 low-power microcontroller operates at 3.1 V and the battery can vary from 3.4 V at system standby to 4.2 V at full charge. For the microcontroller, this means the regulator only needs to step down. The minimum power consumption of the microcontroller is on the order of 100 μA, which makes the use of a linear regulator ideal because of the low quiescent current. We chose the TI TPS71501 regulator with a quiescent current of only 3.2 μA.

The GPS active antenna operates at 2.8 V and requires a low noise supply for the best accuracy and fastest acquisition time. Therefore, we use a low noise linear regulator (National LP2982-2.8) for the GPS antenna supply.

Switch Converters We use switch converters to power our GPS and radio. Switch converters are highly efficient and have the ability to both step up and step down voltage; however, they are electrically noisy and complex. In a design that uses a single battery source, though, the need to step up voltage sometimes makes it difficult to avoid using them.

The GPS requires 3.3 V to power the on-chip Hitachi microcontroller and analog processors. Since it draws 140 mA while attempting to find a lock, we need a buck-boost converter that can both step up and step down the voltage. We use the Linear Technology LTC3440 buck-boost converter which has up to 94 percent efficiency; it uses a four-switch arrangement that eliminates the need for a transformer usually required for buck-boost converters.

The radio requires a 5 V source and draws from 50 mA when receiving to 150 mA when transmitting. For simplicity, it uses the same type of voltage regulation chip as the GPS. Since a buck-boost converter has pulsating input and output currents, we reduced the switching noise by adding large, low equivalent series resistance (ESR) Os-con capacitors as well as LC post filters to stabilize the power line.

While a Cuk converter would have a better noise profile than a buck-boost converter, we decided against it due to the difficulty of closing the feedback loop. This problem would result in an unsuitable transient response for the system's large current swings [5].

5.3 Batteries and Solar Recharge

Due to ZebraNet's relatively high average power consumption (0.03 W - 0.07 W), our system would require a battery weighing approximately 10 kg to operate for 1 year. This weight is not acceptable for zebra tracking. Instead, our

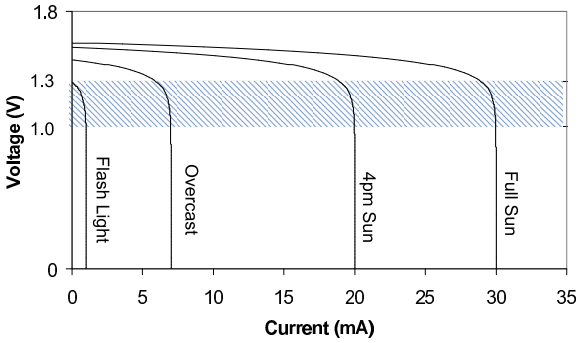

Figure 5: Solar module characteristics of three cells in series

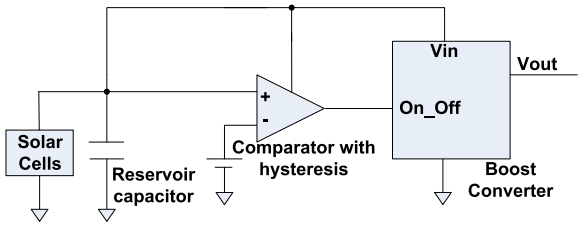

Figure 6: Solar Cell Module

system uses a rechargeable battery with a solar array. The additional weight of the battery and the solar array is light (200 g), so the size and weight are acceptable for this application.

5.3.1 Solar Cells

Our system utilizes a solar array of 14 solar modules each producing at maximum of 7 mA at 5 V.

Special care is required when designing solar modules. When solar cells are connected in series, shaded cells can be reverse-biased, and as a result, the entire string delivers negligible power. When the cells are connected in parallel, the voltage will be limited by the cell with the lowest voltage. If the amount of sunlight shining on a connected string of cells is significantly different, the cells with lower sunlight will dissipate power produced by the other cells and may burn out. This property requires us to design in isolation between solar cells to prevent damage.

Each solar cell produces approximately 0.55 V open circuit voltage. The cells stay at an approximately constant voltage until they reach their current limit, which varies depending on design and light conditions. This behavior means that some smart circuitry is necessary to get the maximum energy from the cells.

We chose modules with three cells in series because they minimize the number of the cells in series and still produces a high enough voltage to guarantee the boost converter can have reliable startup. Figure 5 illustrates the solar cells input voltage characteristics. Maximum power is produced at the corner where the cells change from constant voltage to constant current.

In order to receive maximum power from the cells, we need to keep the cell operating at the maximum power corner. To achieve this, we designed the circuit shown in Figure 6. This circuit has a simple comparator (TI TPS3103) controlling a boost converter (Linear Technology LTC3400) to keep the three-cell series operating near the maximum power corner voltages, between 1.0 V and 1.3 V. The boost converter boosts the input voltage to a usable 5 V and isolates solar modules from each other.

With this design, the outputs of different solar modules can be connected together in parallel to allow the power generated by each cell to be added together. This design also eliminates the possibility that solar cells will be damaged by current flowing backwards into them. The output of the boost converter is used to charge the battery. A large input reservoir capacitor reduces the on/off switching of the boost converter.

It is reasonable to wonder whether there are viable alternatives to solar cells. For example, energy can also be scavenged through mechanical conversion techniques such as vibration [29]. We can estimate the weight necessary to produce enough energy to power our system with the equation: $P = m * g * h * e$. In this equation, m is the mass of the moving weight, g is the gravitational acceleration constant, h is the height change per second, e is the conversion efficiency, and P is the power produced. If we assume that on a walking zebra the weight sways 0.1 meter vertically and the conversion efficiency of the system is 10%, the mass of the weight needs to be 1 kg to generate the needed 0.1 W. This is double the current weight of the collar. Even more weight would be needed if the zebra does not walk constantly. This added weight could negatively impact the survival rate of collared animals. In comparison, 14 of our solar modules combined weigh 100 grams and produce 0.4 W in full sun.

Piezoelectric techniques are also a possibility, where pressure from the animal's weight might be converted to electrical energy. For example, one might think of a power-generating horseshoe, where the weight of each step is converted into electrical energy. While this is promising in terms of the energy generated for a 400 kg animal like a zebra, there are significant reliability and packaging issues with this approach. For our purposes, solar cells offer both reasonable sizes and also the promise of useful reliability.

5.3.2 Batteries

A purely solar system would not allow the system to function at night or during bad weather. Yet these are the times where behaviors of these animals are most interesting and visual tracking is most hazardous. A compromise was reached to use solar modules which charge a 2 mA-h lithium-ion battery. The lithium-ion battery was chosen for its relatively constant voltage over its entire capacity range, varying from 4.2 V at full charge to 3.0 V at depletion. This battery would provide us with about 72 hours of normal operation without recharge. Lithium-ion batteries offer one of the best energy densities amongst different battery chemistries.

To charge the Li-ion battery, we implemented a pulse charging scheme with a MOSFET switch. Pulse charging is the least hardware intensive charging scheme. Since solar cells are current limited by default, no special hardware is necessary to ensure a safe charging current to the battery. The charger is controlled by the microcontroller and the battery voltage is sensed by the on-chip 12-bit analog-to-digital converter. When the battery voltage nears 4.2 V, the processor will begin pulse charging cycles. Off-time lengthens as the battery approaches full charge. Since the system is always consuming power in large pulses, no special programming is necessary to ensure charge termination.

The charging system does depend on the system to go into low-power mode when the battery approaches depletion.

6. EVALUATION

6.1 GPS Accuracy

Since the GPS unit consumes a great deal of energy, a key aspect of our design is to balance the energy usage of the GPS against the accuracy of the position locks. The most energy-efficient approach would be to turn on the GPS, take one fix, and then turn the GPS off again. We have found that this method is too inaccurate; locks acquired in this way from a stationary GPS running over a 24 hour period had a standard deviation of around 32.6 meters with many extreme outliers differing by more than a kilometer. This accuracy is poor compared to a standard deviation of about 5.3 meters in a GPS chip that is left on continuously. Inaccurate locks can compromise all of our results because we do not know which ones we can trust. Leaving the GPS on continuously drains the battery faster than we can recharge it, so that it not an option either.

Given these constraints, we developed a method that approaches the accuracy of leaving the unit on all the time while limiting energy consumption to close to that of the inaccurate first-lock method. To determine this method, we ran experiments on a GPS unit with an obstructed view of the sky over a 24 period.

We found two properties characteristic of GPS that can be used to determine the accuracy of the locks. First is the number of satellites in view of the GPS antenna. Although the GPS will attempt to determine its position if it can see at least three satellites, three-satellite locks are not as accurate as locks generated from four or more satellites. Therefore, we ignore any locks with less than four satellites.

The second property is the configuration of the satellites visible to the GPS antenna. Dilution of precision (DOP) is a metric of the geometric positions of the satellites [24]. The accuracy of the position decreases as DOP increases. Our GPS chip automatically throws out any fixes with a DOP greater than 10. Based on experimental results, we decided to further throw out any lock with a DOP greater than six. In addition, since the satellite configuration is unlikely to change much within the minute that the GPS is powered on, if the GPS unit receives ten consecutive readings in which the DOP is greater than seven, we turn the unit off. The ten readings give the GPS a chance to find an additional satellite. We chose this number based on experimental data that showed that if we did not find another satellite within ten seconds, we were unlikely to find one before the minute of on-time expired.

Finally, we find that the first two locks that the GPS claims are valid are in practice far less accurate than the third. So, we ignore the first two locks that the above methods would allow, and then accept the third.

With this algorithm, the GPS unit is powered on an average of 25 seconds every eight minutes and the standard deviation of the accuracy is 5.33 meters. This accuracy is comparable to leaving the unit on at all times. The primary disadvantage to this method is that in our experiment, it failed to acquire a lock roughly half the times the unit was turned on. However, since the locks it accepts are more reliable, this method increases the integrity of our data. Also, the experiment is somewhat conservative in terms of satellites in view, as only half of the sky is visible to the test setup. Although we expect the zebras to be mostly in an open terrain, obstructions do occur both due to brush/trees and also if the collar orientation aims the GPS antenna sideways or downwards.

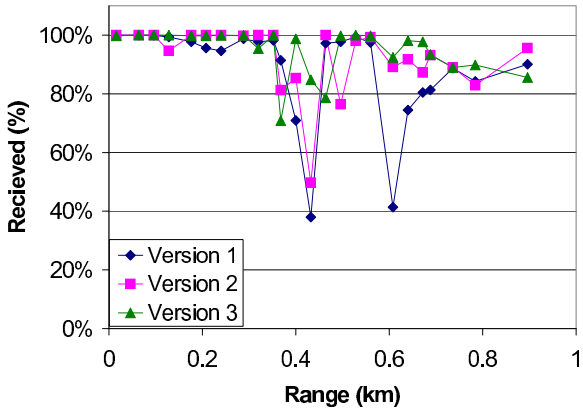

Figure 7: Percent of Successful Receive vs. Distance on Versions 1, 2, and 3.

6.2 Radio Range

Radio range is critical for ensuring network connectivity and therefore data collection success in our network. In this section, we present experimental data on the actual observed behavior of our "5-mile" range radios.

All radio tests were conducted on a 0.9 kilometer stretch of straight road outside of Princeton's engineering building. Tests were conducted between midnight and 6 AM, when curbside parking is illegal and few cars pass through. Trees and a few buildings line the sides of the street. This location, while not ideal, lets us compare the radio behavior from different design iterations in a naturally dispersive environment.

In the first experiment, we used a ZebraNet node as a transmitter and tested the transmit side of the nodes. The receiver is a development board with a low noise power supply. The data from the development board is then recorded on a connected computer. Figure 7 shows how the packet delivery success rate varies with distance. Each curve corresponds to one of the three hardware versions. The transmit success rate differ slightly between each version and stays fairly high through 0.9 kilometers. Version 3 shows a slightly higher success rate than other versions mostly due to a lower signal-to-noise ratio resulting from a better designed power supply. However, evidence of fading is obvious from the significant drop in success rate at around 0.4 and 0.6 kilometers. This fading effect makes it necessary to have error correction and detection algorithms even over short distances. For example, our flooding protocol replicates key peer discovery and acknowledgement packets to improve resilience to dropped packets.

Figure 8 shows receive power vs. distance. This experiment was also conducted on the 0.9 kilometer road described above. A packet is sent from the development board to a node which then retransmits the packet back to the source where the receive power is recorded. No data is available for Version 1 because it had such difficulty receiving that it could not execute the test. (This problem is discussed in more detail in Section 6.3.) For reference, we also plotted

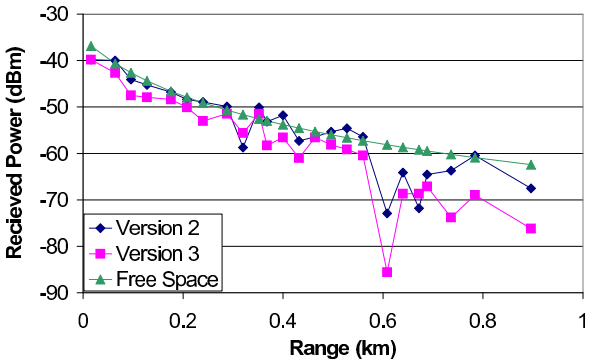

Figure 8: Radio Receive Power vs. Distance of Version 2, Version 3, and the ideal free space model.

the free space ideal curve, representing a quadratic drop in power versus range, along with the other curves.

This figure shows a near exponential decay in received power as range increases rather than the expected quadratic decay. This difference could be due to the location of the test site where the path loss is complicated by diffraction loss and scattering as well as interactions between the two. Our testing location seems to be similar to a homogeneous random scatter medium, where the same pattern is observed [17].

In addition, this figure shows that Version 3's received power is in general lower than that of Version 2, which could be caused by variations between different radio modules. However, as shown in Figure 7, Version 3 still has a higher successful reception rate. This is due to improvements in the board design that increased receiver sensitivity.

6.3 Power Supplies

The power supply design was improved through each version to reduce noise and, thus, improve the overall performance of the system. Version 1's power supply design underestimated the current requirements of the peripherals. This caused noise coupling throughout the entire system. For example, during radio transmission, noise coupled from the radio power lines to the microcontroller power resulted in occasional unintended microcontroller resets, as well as poor receiver performance. The left-hand graph of Figure 9 shows the oscilloscope trace of Version 1's radio and microcontroller AC power supply noise during transmission. From the radio power supply trace (the lower curve on the left-hand graph) we see that switching noise is almost 5 V peak to peak. This trace also shows the lower frequency triangle wave that was caused by the pulsing current output of the radio buck-boost converter. While both the high and low frequency noise affect the radio performance, only the switching noise is coupled to the microcontroller power supply as shown in the top trace of this graph.

In Version 2, the coupling problem was fixed by placing the power supply further from the digital ground as well as larger capacitors. We were able to reduce the shot noise but the low frequency noise component remained. This significantly increased the receiver sensitivity and improved the performance of the system. However, the radio power fluctuation was still too large and could be further improved.

The right-hand graph of Figure 9 shows the power supply of Version 3 in the same scale as that of Version 1. The addition of LC post filters in this version, coupled with careful PCB placement, significantly reduced the power noise. Fast transient response is maintained due to a very small inductance created by farad beads. This version also implements a soft start, which slows the voltage transitions to prevent current from rushing to the output and causing peripheral startup noise. The power supplies are much steadier in this version than in previous versions. As shown in Figure 7, this resulted in an increase in the percentage of packets received correctly.

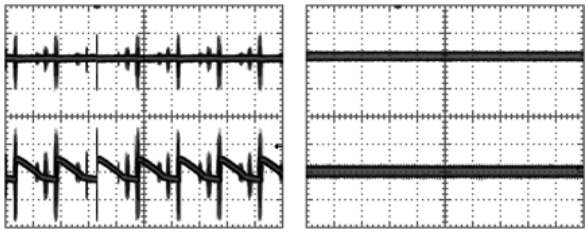

Figure 9: Version 1 (left) and Version 3 (right) Power supply noise. For each graph, the CPU power supply is shown on top (200mV/div), and the radio on the bottom (1V/div). The x-axis, time, is shown at 10μs/div.

Figure 10: Plains Zebra at Sweetwaters Game Preserve with ZebraNet collar. The collar is made of a white butyl belting material. The darker spots visible on the upper half of the collar are the solar modules which lie between the two layers of butyl belting, with openings exposed to allow sunlight in.

Overall, these iterations in power supply design have been a central part of ZebraNet hardware development. Since sensor networks exhibit large current swings, voltage regulation is an important issue. Furthermore, the small board area also makes board layout an important issue.

7. DEPLOYMENT EXPERIENCES

In January, 2004, we brought a set of ZebraNet nodes to Kenya and deployed them on zebras at the 100 sq. km. Sweetwaters game reserve in central Kenya. The goals for this deployment were to gain experiences with the impact of the collaring on wild zebras, to get information on the long-term behavior and reliability of the collars, and to (of course) collect some initial zebra movement data. For this test, our goal was to collar a total of 6-10 zebras; this is less than the full 30-node deployment that we intend to do

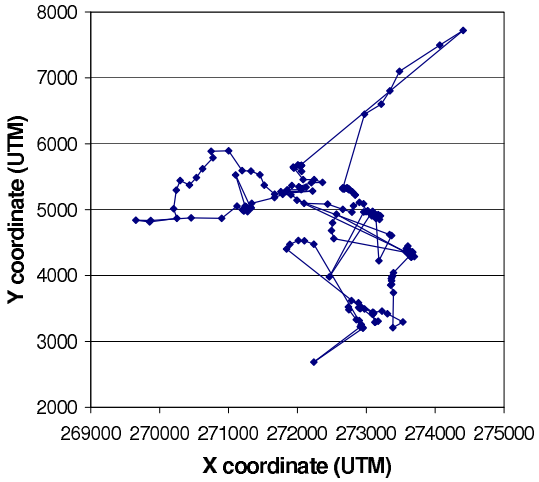

Figure 11: An example of GPS location data collected for one zebra over a 24 hour period in UTM coordinates. The pictured area is 36 square kilometers.

in roughly 12 months. This section describes some of our experiences and lessons learned.

On January 12, we collared seven zebras, six females and one male. Figure 10 shows a photograph of a collared zebra. Immediately after collaring, and for roughly a day afterwards, the collared zebras seemed to be affected by the collars, with additional head shakes being observed by the biologists. By the end of the first week, however, the collared zebras showed little difference from their uncollared counterparts in terms of behaviors like head shakes, eating, and social grouping.

Physical Design Figure 10 gives a sense of the collar's physical design. For reliability and ruggedness, the radio antenna is embedded between the layers of butyl belting that form the collar. This was an imperative part of the physical design, since most commercial tracking collars have external whip antennas that break off easily once deployed. The dipole antenna is made of a strip of copper tape and braid. The radio hops within a small range of frequencies, and the dipole's length is tuned for the center of these frequencies. To maintain a good radio range, one needs to reduce absorption of the signal into the animal's neck; we did this via a ground plane of woven conductive mesh separated 1 cm from the antenna by a foam dielectric.

Like the radio antenna, the GPS antenna also needs to be protected. It too is sandwiched between the butyl collar layers, and is designed to be oriented near the animal's mane, facing upwards to see satellites most easily and therefore improve the speed and accuracy of GPS fixes. Our experiments showed that the GPS antenna received well even inside the butyl layer. A v-shaped metal wedge in the collar was designed to keep the collar upright at all times, but because the collars were put on more loosely than designed for, the collars would rotate more than planned for, misdirecting the GPS antenna at times.

The photograph also illustrates the proper orientation of the solar modules, which are the dark squares on the "mane" portion of the collar. Since some of the solar modules wrap somewhat around the zebra's neck (and since the collar sometimes rotates to aim solar modules downward) we do not reach full charging efficiency; we designed for roughly half efficiency.

Zebra Data Collected Figure 11 shows a collection of the data points gathered by ZebraNet during this initial deployment. This collection comprises some of the most fine-grained movement data collected for zebras, and has helped give biologists new insights into the daily home ranges of different zebras, how they interact and overlap. In particular, to our knowledge, this is the first night-time data collected for zebras, which allows biologists to pose and answer questions about how zebra movements differ from day to night. Although more experiments are needed, the data indicates that zebras explore more wooded areas and river gullies at night, as compared to their daytime ranges primarily in open grasslands. It is presumed that their agility in these areas proves a greater asset against predators as nighttime falls.

Lessons Learned and Future Plans: Radio range is a key factor in ZebraNet's ability to percolate data. Perhaps the biggest technical issue in our deployment was that radios specified for 5-mile range, and tested in New Jersey to have a roughly 1-mile range, displayed shorter communication ranges once deployed. The radio ranges also displayed high variance amongst the different collars. We measured ranges of 100 meters to 1.2 km for the collars tested in Kenya. While it is difficult to completely assess the causes of this issue, we believe that some of the following factors lied in: the antenna's ground plane design, corrosion on the radio antenna's connection with the radio chip, and even simply the fact that zebras graze with collars close to the ground which increases the absorption and reduces the range.

Energy constraints drove our choices regarding the event schedule and duty cycle planned for data collection and percolation. Some of these, such as the two-hour interval between radio times, proved overly restrictive. Our future work will implement a more opportunistic scheme to allow collars to detect each other and communicate whenever they are within range, not only at two-hour intervals. To achieve this, we need a lower-energy radio, which can be achieved by custom-designing a radio with distinct characteristics for the receiver and the transmitter. We also plan to drop our communications to a lower frequency in order to naturally improve the range characteristics.

Our future plans also include switching to a newly-available GPS chip which offers quicker fixes at lower energy costs and increasing the number of sensors on the collar. In particular, we are currently testing XYZ orientation sensors that will assist the biologists in collecting data about the zebra's activities: a collar low to the ground is likely eating, while other patterns regarding how the head is raised may be mined for information on the zebra's social behavior, vigilance towards predators, and so forth.

8. RELATED WORK

This section touches on several of the most relevant categories of related work.

Sensing Hardware The Mica2 Mote developed at Berkeley and commercialized by Crossbow Technologies is in wide use [10]. Built from off-the-shelf components, Mica2 Motes have 8-bit Atmel ATmega 128L processors running at 4 MHz and input ports to support a variety of sensors. In laboratory experiments, Motes are often used to evaluate protocols and sensor network behavior [8][18][21]. Harvard researchers

have also established a remotely-programmable and measurable Mote infrastructure in their building to conduct a range of experiments [37].

Intel has also developed a Mote version [16] and has begun experimental deployments with it [12]. UCLA's Medusa-MK2 has a Berkeley Mote as its base [30]. It then adds a second 32-bit, 40 MHz AT91FR4081 ARM/Thumb processor to support computationally-intensive applications. The second processor can be turned on and off as needed to conserve energy.

MIT's μAMPS [23] and Rockwell Science Center's WINS nodes [27] both use 32-bit StrongArm processors and are much more powerful, and consume much more energy, than ZebraNet nodes or Mica2 Motes. μAMPS uses a 206 Mhz SA1110 processor and WINS uses a 133 MHz SA1100 processor. To compensate for the additional energy consumption, the processors spend most of their time in a 1mW sleep mode. In addition, μAMPS reduces energy via dynamic voltage scaling. Thus far, the μAMPS project has developed a low-power radio and a low-power DSP module for their nodes and they are planning to design a custom microcontroller as well [4].

Sensor Node Radios Once deployed, most sensor nodes are intended to remain stationary and in a densely packed configuration. As a result, they use radios with much shorter ranges—only 10-100 meters typically—which consume an order of magnitude less energy than our multi-mile radio. In fact, numerous research efforts inspired by these nodes, including the PicoRadio project [26], are devoted to further reducing the size and power consumption of radios in sensor nodes, which will in some cases lead to even smaller typical radio ranges. These drops in radio range will typically need to be remedied by higher network density [28].

Wildlife Tracking with Sensors Telenor R&D's Electronic Shepherd project [34] and UC Davis's Southern California Puma Project [35] are examples of projects using GPS-based sensors for animal tracking. They face size, weight, and energy constraints very similar to those of our project. The primary differences between our system and these systems are that they track over a more limited geographic area and they require fewer GPS readings and communications per day. This allows the sensor nodes to communicate directly to a fixed infrastructure rather than propagating the data from node to node. In other projects, researchers have exploited fixed infrastructure to track animals such as falcons [33] and salmon [2] without GPS.

Other Applications and Deployments In an early sensor deployment in 2002, 43 Motes were deployed on Great Duck Island, off the coast of Maine, to monitor the island habitat and bird population [31]. In addition to this deployment, there have been several other Mote deployments to facilitate environmental monitoring including [3], [13], and [36].

WINS nodes have been deployed on US Navy vessels through the Office of Naval Research [27]. Their long-term goal is to use the sensors to monitor the health of certain pieces of machinery that at the moment, crew members must routinely check to determine when parts need to be replaced.

Intel and Carnegie Mellon University are developing a software infrastructure called IrisNet [9] that would support a worldwide sensor network. As a first step, they have deployed a networks of cameras along the Oregon coastline to allow researchers to collect visual environmental data remotely. The University of Hohenheim is developing a Precision Farming system to help farmers improve quality and increase yield by providing them with sensors to help them determine the best places, conditions, and processes with which to grow and harvest crops [6].

9. CONCLUSION

This paper has presented our experiences developing hardware and low-level software for ZebraNet. We have developed a fully-functional, highly-mobile, energy-efficient sensing system that determines accurate positional data and can propagate it through the network. To reduce energy consumption, we have implemented several system-level energy-management techniques. Even with these methods, the high power of the GPS and the long system lifetime target requires that our hardware utilize a solar array for recharge. Our hardware choices work well for the sparsely connected, node-to-node communication model we require.

Another lesson that arises through our hardware versions concerns the choice of microcontroller. While we originally planned to scavenge cycles from the GPS processor [14], the importance of a "clean" software environment prevailed and we chose a second off-chip microcontroller in Version 1. By devoting a microcontroller to our sensor software rather than scavenging cycles from the GPS processor, we greatly simplified software development for our high-burst-rate computations and communications. In going with a second microcontroller, we were also able to reduce the energy budget since it can run at a much slower clock rate.

The severe energy constraints of sensor nodes mean that details really do matter. Items like voltage regulation become important in sensor hardware due to the extreme amperage fluctuations caused by units switching on and off, the need for low-noise power supplies for radio success, and the need for overall energy efficiency. Noise and crosstalk issues are further compounded in sensor nodes by the fact that small board area is a primary goal.

Overall moving from theory to experiment and practice is an invaluable research step. The current increasing focus on prototyping is helping the sensor research community progress towards the end-goal of practical, efficient, and reliable sensor networks in wide-scale deployment. A diversity of sensor hardware prototypes can offer a range of insights across the design space, and we view our ZebraNet hardware as contributing one further point to this space.

10. REFERENCES

[1] Atmel. AT45DB041B, 4Mbit, 2.7-Volt Only Serial-Interface Flash with 2 264B SRAM Buffers data sheet. `www.atmel.com`, June 2003.

[2] Census of Marine Life. POST: Pacific Ocean Salmon Tracking Project. `http://www.postcoml.org/`, 2003.

[3] Center for Embedded Networked Sensing. Research Infrastructure: James Reserve Local Area Power System and Network Enhancements. `http://www.cens.ucla.edu/Project-Descriptions/Research_Infrastructure`.

[4] A. Chandrakasan. MIT μAMPS Project. `http://www-mtl.mit.edu/research/icsystems/uamps/`.

[5] S. Ćuk. Switching DC-TO-DC Converter with Zero Input or Output Current Ripple. In *Proc. IEEE Industry Applications Society Meeting*, Oct. 1978.

[6] R. Doluschitz. Precision Agriculture - Applications, Economic Considerations, Experiences and Perspectives. In *EFITA 2003 Conference*, July 2003.

[7] P. Eggenburger. GPS-MS1E Miniature GPS Receiver Module Data Sheet. `www.u-blox.ch`, Oct. 2001.

[8] D. Ganesan et al. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks. UCLA Tech. Report CSD-TR 02-0013, Feb. 2002.

[9] P. Gibbons et al. IrisNet: An Architecture for a Worldwide Sensor Web. *IEEE Pervasive Computing*, 2(4):22–33, Oct-Dec 2003.

[10] J. Hill and D. Culler. Mica: A Wireless Platform for Deeply Embedded Networks. *IEEE Micro*, 22(6):12–24, Nov-Dec 2002.

[11] J. Hill, R. Szewczyk, et al. System Architecture Directions for Networked Sensors. In *Proc. 9th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Apr. 2000.

[12] Intel Corp. New Computing Frontiers–The Wireless Vineyard. `www.intel.com/labs/features/rs01031.htm`, 2004.

[13] Ireland National Centre for Sensor Research. Eco-Sensor Network Project. `www.ncsr.ie/ecosensorweb`, 2004.

[14] P. Juang, H. Oki, Y. Wang, et al. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *Proc. 10th Intl. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, Oct. 2002.

[15] H. Kim and S. Lee. A New Flash Memory Management for Flash Storage System. In *32rd Annual Intl. Computer Science and Applications Conference*, Oct. 1999.

[16] R. Kling. Intel Mote: An Enhanced Sensor Network Node. In *Intl. Workshop on Advanced Sensors, Structural Health Monitoring, and Smart Structures*, Nov. 2003.

[17] I. Kovacs, P. Eggers, and K. Olesen. Radio Channel Characterisation for Forest Environments in the VHF and UHF Frequency Bands. In *IEEE 50th Vehicular Technology Conference, Amsterdam*, pages 1387–1391, Sept. 1999.

[18] Q. Li, M. DeRosa, and D. Rus. Distributed algorithms for guiding navigation across a sensor network. In *2nd Intl. Workshop on Information Processing in Sensor Networks (IPSN)*, 2003.

[19] T. Liu and M. Martonosi. Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems. In *ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPoPP)*, June 2003.

[20] T. Liu, C. Sadler, P. Zhang, and M. Martonosi. Implementing Software on Resource-Constrained Mobile Sensors: Experiences with Impala and ZebraNet. In *Proc. Second Intl. Conference on Mobile Systems, Applications and Services*, June 2004.

[21] D. Malan et al. CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care. In *Intl. Workshop on Wearable and Implantable Body Sensor Networks*, Apr. 2004.

[22] Maxstream. 9XStream Wireless Modem Data Sheet and OEM manual. `www.maxstream.net`, June 2002.

[23] R. Min et al. Energy-Centric Enabling Technologies for Wireless Sensor Networks. *IEEE Wireless Communications*, 9(4):28–39, Aug. 2002.

[24] P. Misra and P. Enge. *Global Positioning System: Signals, Measurements, and Performance.* Ganga-Jamuna Press, Lincoln, Mass., 2001.

[25] Mpala Wildlife Foundation. Mpala research centre. `http://www.mpalafoundation.org/researchctr/`.

[26] J. Rabaey et al. PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking. *IEEE Computer*, 33(7):42–48, July 2000.

[27] Rockwell Science Center. Wireless Integrated Network Sensors (WINS). `http://wins.rsc.rockwell.com/`.

[28] K. Romer. Tracking Real-World Phenomena with Smart Dust. In *EWSN 2004*, Jan. 2004.

[29] S. Roundy et al. A 1.9GHz RF Transmit Beacon using Environmentally Scavenged Energy. In *IEEE Intl. Symp. on Low Power Elec. and Devices*, Aug. 2003.

[30] A. Savvides and M. Srivastava. A Distributed Computation Platform for Wireless Embedded Sensing. In *Proc. of Intl. Conference on Computer Design (ICCD)*, Sept. 2002.

[31] R. Szewczyk et al. Lessons from a Sensor Network Expedition. In *EWSN 2004*, Jan. 2004.

[32] Texas Instruments. MSP430 Mixed Signal Microcontroller. `http://www.ti.com/`, 2002.

[33] The Center for Conservation Biology. VAFALCONS. `http://fsweb.wm.edu/ccb/index.html`, 2002.

[34] B. Thorstensen et al. Electronic Shepherd—A Low-Cost, Low-Bandwidth, Wireless Network System. In *Proc. Second Intl. Conference on Mobile Systems, Applications and Services*, June 2004.

[35] UC Davis Wildlife Health Center. Southern California Puma Project. `www.vetmed.ucdavis.edu/whc/scp/`, 2004.

[36] University of Western Australia. Environment Monitoring of Soil Moisture with Wireless Sensor Networks. `http://www.csse.uwa.edu.au/adhocnets/WSNgroup/soil-water-proj/`, 2004.

[37] M. Welsh. Environment Monitoring of Soil Moisture with Wireless Sensor Networks. `http://motelab.eecs.harvard.edu/`, 2004.

[38] Xemics. Product Brief: RGPSM002 GPS Receiver. `http://www.xemics.com/`, 2004.

[39] W. Zhao, M. Ammar, and E. Zegura. A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks. In *Proc. ACM MobiHoc 2004*, May 2004.