A Phone-Centered Body Sensor Network Platform: Cost, Energy Efficiency & User Interface

Lin Zhong[*†] **Mike Sinclair**[‡] **Ray Bittner**[‡]

[†]Dept. of Electrical & Computer Eng., Rice University, Houston, TX 77005

[‡]Hardware Devices Group, Microsoft Research, Redmond, WA 98052

Abstract

We have designed a Bluetooth-based body sensor network platform for physiological diary applications and have addressed its challenges in cost, energy efficiency, and user interface. In our platform, an internet-capable phone serves as the center and manages every network member. We designed a Bluetooth sensor node for general sensing devices to join the network without much alteration. Since Bluetooth imposes a large power overhead, we have taken extreme care to minimize its duty cycle. We also incorporated a wrist-worn device as the user interface. It displays information under the instruction of the phone in an ambient fashion, and enables the user to interact with the network conveniently. By leveraging resources on the phone, we are able to minimize the cost and energy consumption of the sensor nodes and the wrist-worn device.

1 Introduction

We are interested in using a wireless body sensor network (WBSN) to continuously collect physiological information for disease diagnosis over a lifetime for everyone. This *physiological diary* application drove us to design a WBSN to address new challenges in *cost, energy efficiency* and *user interface*. Over time, the design must become low-cost to promote user adoption. It must be energy-efficient since frequent battery changes for many sensors will deter users. Finally, it must be able to interact with the user in an ambient, non-interruptive fashion. This paper presents our design of a WBSN platform to meet these challenges.

We recognized that powerful mobile phones have penetrated all walks of life and consequently made a mobile phone the center of our WBSN platform to provide the central intelligence and to manage all network members. This decision has impacted the platform in the following ways. First, our platform is based on Bluetooth due to its high availability on mobile phones, as compared with ZigBee (IEEE 802.15.4). Bluetooth, however, typically consumes more power and has less support for advanced ad hoc networking features. Nevertheless, we have been able to overcome these two shortcomings in our platform. Second, members in our WBSN may only talk with the phone directly, leading to a simplified network. By shifting most tasks to the phone, we further reduce sensor node cost and energy consumption. Third, members in our WBSN export their services through application-programming interfaces (APIs) on the phone. The APIs hide most of the implementation details of the sensors from users and application developers, who only need to write code for the phone to obtain services from the WBSN. While the introduction of a mobile phone helps us meet the cost and energy efficiency challenges, we introduce a wrist-worn device, called *Cache-watch*, into the WBSN to meet the user interface challenge. Under the control of the mobile phone, it quietly displays information regarding the WBSN and user health. It also takes user input to manage the WBSN. To the best of our knowledge, our platform is among the first to extensively explore the role of a mobile device and user interface design in the context of WBSN.

The paper is organized as follows. In Section 2, we provide a system view of the platform and address its design. We then address its energy efficiency issues and user interface in Sections 3 and 4, respectively. We discuss related work in Section 5, and conclude and outline future work in Section 6.

2 System design

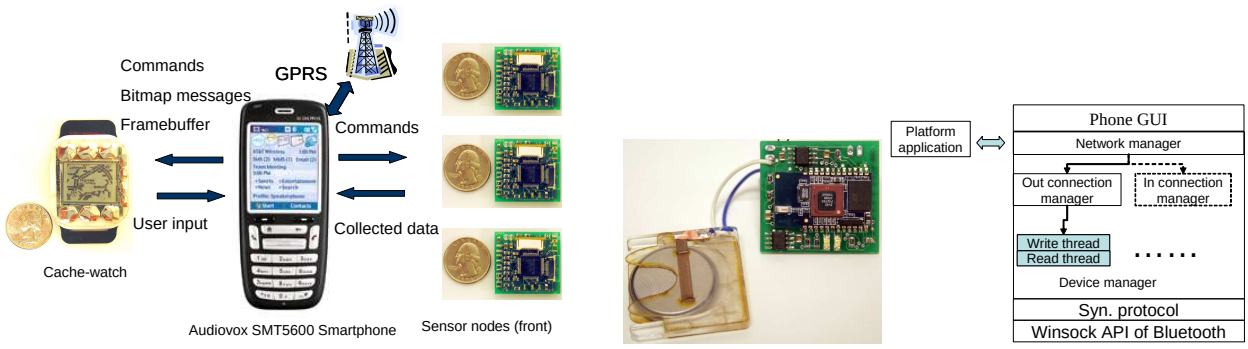
2.1 System view

Our platform consists of one mobile phone, multiple sensor nodes, and a Cache-watch as illustrated by Figure 1(a). Sensor nodes and the Cache-watch communicate directly with the phone via Bluetooth. They are *peripherals* of the phone. The platform provides a set of APIs for applications on the phone to manage the network, collect data from the sensors, and interact with users via the Cache-watch. A typical application retrieves data from sensors from time to time and put sensors into deep sleep mode in between. Data may be reported to an Internet server which provides an analysis, or the phone may perform its own analysis, which is then displayed through the Cache-watch.

2.2 Hardware

The sensor node is powered by a rechargeable 60mAh coin-size Lithium-ion battery and is hosted on a double-layer 1.3"x1.3" PCB. Figure 1(a) shows the front view of the sensor node. Figure 1(b) shows the back view of the sensor node board. There are only four integrated circuit units on the board: a TI MSP430

[*]Lin Zhong was an intern with Microsoft Research in the summer of 2005



(a) System view (b) Sensor node (back) (c) Phone software organization

Figure 1. Phone-centered body sensor network platform

micro-controller, a KC21 Class 2 Bluetooth OEM module [2], and two linear regulators. The first regulator regulates the power supply to 3.3V for most of the electronics, while the second is under the control of the MSP430 to power the Bluetooth module. The board also has a JTAG programming interface. The Cache-watch essentially uses the same PCB, with an attached 128 by 96 dot-matrix LCD.

The sensor node is designed to be a general analog signal capturing device. It can measure up to 10 analog channels with the 12-bit ADCs internal to the MSP430.

2.3 Software

Peripherals: Unlike many wireless sensor nodes, ours do not require an OS because the introduction of a phone significantly reduces their complexity. Software on the sensor node is developed in C and compiled into 4KB of MSP430 firmware, running in an interrupt-driven mode. Most of the time, the MSP430 is in a power-saving mode with an active timer. It wakes up to execute the corresponding handler upon an interrupt. The UART interrupt handler implements an application-layer protocol, called the SYN protocol, to talk with the phone as we will discuss in Section 2.4. The Cache-watch software is different from that of sensor nodes only in that it implements display updates and capacitance measurements in the timer interrupt handler. Its UART interrupt handler also implements a different subset of the SYN protocol.

Phone: An Audiovox SMT5600 smartphone with Windows Mobile 2003 SE is used in our prototype. Since the phone fulfills most of the network functionality, its software is quite complicated and is implemented as middleware with Visual C++. The software architecture of the phone is illustrated by Figure 1(c). The software has a user interface for users to register and remove peripherals and to specify parameters for the WBSN. The phone controls its peripherals and exchanges data with them through the SYN protocol via Windows socket APIs for Bluetooth.

The software, *network manager*, first creates two threads to manage outgoing and incoming connections. It is the phone's interface between the platform and its user, and the interface between the platform and any application using it. It determines when the phone will connect to a peripheral based on user settings, history, and application requests. It also exports the collected sensor data to interested applications.

The *out-connection manager* creates and maintains two threads for each registered peripheral, one for reading and the other for writing. The two threads for a peripheral device are collectively referred as its *device manager*. The out-connection manager also provides power management for Bluetooth on the phone according to the communication schedule and exchanges data with device managers. The device manager is in charge of managing the corresponding Bluetooth socket connection according to the schedule.

2.4 Network

Since our platform is built with standard Bluetooth modules, we must utilize the available parameters from the Bluetooth protocol stack for the SYN protocol. The protocol consists of six types of commands. Three commands are specific to the Cache-watch. They will be detailed in Section 4. INSTRUCTION commands are used by the phone to request data from sensor nodes; SENSOR_DATA commands are used by sensor nodes to send data to the phone; POWER commands are used by the phone to instruct a peripheral to power off its Bluetooth module for a certain period of time. This is the basis of the energy-saving scheduled communication mechanism as will be addressed next.

3 Design for energy efficiency

3.1 Energy profile

Most of the time, the phone is in STANDBY mode. It consumes about $20mW$ when Bluetooth is disabled, and $21mW$ when Bluetooth is enabled in PENDING mode with default parameters from Windows Mobile. When transferring data at 115Kbps the phone consumes about $440mW$. It is worth noting that when the SIM card is removed, the phone consumes only about $4mW$ in STANDBY mode. Therefore, the power overhead from Bluetooth in STANDBY mode is minimal as compared with that from staying on the cellular network. During these power measurements the phone display was powered off. For peripherals, however, the KC21 Bluetooth module dominates power consumption. While non-Bluetooth components consume less

than $1mW$ in total, the KC21 module consumes more than $8mW$ even in SNIFF mode. When it is in PENDING mode, seeking a connection with the default settings, it consumes about $74mW$ Pending Power, or P_P. It consumes about $160mW$ when transferring data at 115Kbps. Not surprisingly, we give a higher priority to peripherals in Bluetooth energy optimization.

3.2 Energy optimization

Bluetooth imposes the most power overhead for WBSN activities. Since we do not have access to the KC21 Bluetooth stack firmware, we rely on tuning high-level Bluetooth parameters, which makes our techniques applicable to most commercial-off-the-shelf (COTS) Bluetooth modules and complementary to most other low-power Bluetooth techniques.

Scheduled communication: In our WBSN, the phone schedules its data exchange with each peripheral. If the connection were maintained, the peripheral would consume at least $8mW$, which is the Bluetooth power consumption in SNIFF mode and will lead to an unacceptably short battery lifetime. If, on the other hand, the phone schedules the next communication with the peripheral after each exchange, both parties will know when the next connection will be required, and a lower power state can be used. Fortunately, physiological information only requires sampling once every several tens of seconds for the purpose of disease diagnosis in the worst case. Such a long communication interval makes it possible to power Bluetooth off for sensor nodes between samples and re-establish the connection before the next communication. The energy consumed during the active Bluetooth connection period is the overhead of WBSN activities, and this is our focus of energy optimization. Figure 2(a) shows the power trace for the connection period of a sensor node with default manufacturer Bluetooth setting. The sensor's Bluetooth module remains in PENDING mode for about 150ms before it discovers the phone's Bluetooth module and establishes a connection. The overall sensor node energy cost for establishing a connection from the powered-off state will be about $0.11Joule$. This overhead is justifiable if the communication interval is longer than 15 seconds. We must note that the time the peripheral's Bluetooth module spends in the PENDING mode before establishing a connection, T_P, depends on not only the peripheral's but also the phone module's Bluetooth settings. In this experiment, both parties use the default manufacturer settings. To minimize T_P for peripherals, the phone's Bluetooth module is actually powered up one second earlier.

Bluetooth parameter tuning: In PENDING mode, a Bluetooth module conducts paging/page scanning periodically. Increasing in the paging/page scanning duty cycle reduces T_P but raises P_P, power consumption in the PENDING mode. We have observed that the energy consumed in the PENDING mode decreases as the duty cycle increases.

Protocol optimization: Let us continue examining Figure 2(a). After the connection is established, the KC21 Bluetooth module remains connected for more than one second before receiving the first command from the phone. This delay costs more than half of the sensor energy consumption after a connection is established. We are currently investigating the cause.

After receiving the INSTRUCTION command, the sensor node samples the data and sends it to the phone. Then, it waits to receive the POWER command from the phone to schedule the next communication. It powers off the Bluetooth module immediately after that. By allowing commands to be queued in the sensor, we are able to reduce the rounds of conversation and thus the energy spent in waiting. For example, when a sensor node and the phone are connected, the phone sends the INSTRUCTION and POWER commands first and then waits to get data from the sensor. The sensor will power off its Bluetooth module immediately after sending the data. Queuing commands in the sensor reduces the Bluetooth energy overhead by 8%.

Figure 2(b) shows the average power consumption of a sensor node with different communication intervals. As the communication interval is increased from 15 seconds to 5 minutes, the average power consumption deceases from $20mW$ to $2mW$.

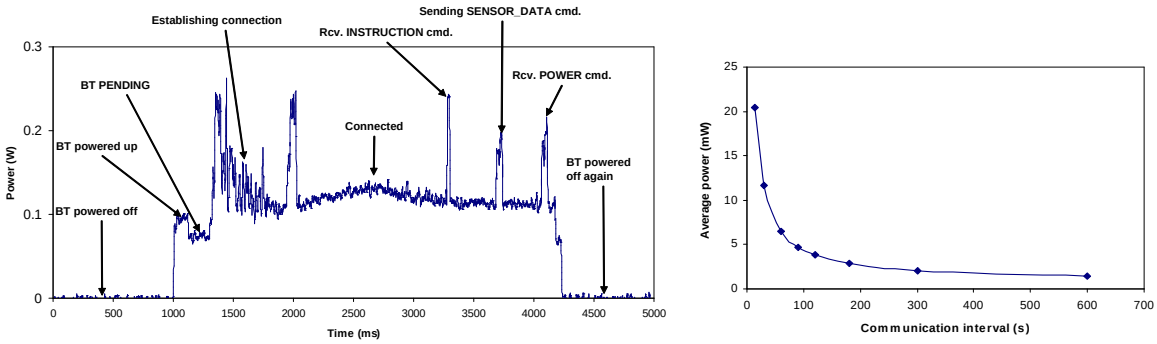
4 User interface design

Very few WBSN designs have considered the user interface, which is, however, critical to our intended applications. First, the user may wish to see their physiological data at any time, which should be as convenient as checking the time. Second, the user may wish to control certain sensor nodes, e.g., to adjust the sampling frequency, which should be as convenient as adjusting the watch time as well. Therefore, we have created a new version of the Cache-watch, as illustrated in Figure 1(a). An older version of the Cache-watch was described in [7, 8]. We next describe the new version.

4.1 Services

The new Cache-watch provides similar services as the old one. For the *active service*, it stays connected with the phone, which updates the Cache-watch display in real-time via Bluetooth as directed by user input. The active service enables the user to interact with the phone and thus the whole WBSN from the wrist. It is implemented using the FRAMEBUFFER and WATCH_INPUT commands. The Cache-watch uses a WATCH_INPUT command to send user input to the phone; the phone uses a FRAMEBUFFER command to update the Cache-watch display framebuffer. This simplifies the Cache-watch design significantly while leveraging the computing capacity and programmability of the phone.

For the *passive service*, the Cache-watch talks with the phone by scheduled communication in a way similar to the sensor nodes. When they are connected, the phone can send the Cache-watch bitmap messages, e.g., a text message in its bitmap format with a MESSAGE command. The Cache-watch can store 20 messages and display them according to their meta data, which are



(a) Power trace

(b) Average sensor node power consumption

Figure 2. Sensor node power consumption

also specified by the MESSAGE command. Like the active service, the phone generates a bitmap message, determines where in the Cache-watch memory it should go, and determines how it should be displayed. This again simplifies the Cache-watch design.

4.2 Input method

Most wrist-worn devices, including the old Cache-watch, use buttons for input. The new Cache-watch, however, uses capacitive touch-sensing to reduce design cost. As shown in Figure 1(a), there are two rows of three or four metallic pads. The MSP430 uses a simple technique to measure the capacitance of a pad and determine whether the pad is being touched. It can also tell whether a row has been swiped from one end to the other. As a result, eight different inputs can be generated. The phone and the Cache-watch interpret the inputs for the active and passive services, respectively.

5 Related work

It is beyond the scope of this short workshop paper to conduct an exhaustive survey. Instead we have to focus on several most related works. Most body-sensor networks [1,5] use IEEE 802.15.4, probably due to its popularity with ad hoc wireless sensor networks. Leopold *et al.* [3] studied Bluetooth as a candidate for ad hoc wireless sensor networks. In these studies, PDAs and mobile phones may collect data from the sensor networks. However, they do not play as extensive a role in management and computation as they do in our platform. As a result, sensor nodes in these studies must still run TinyOS instead of leveraging the resources on a mobile device to simplify their design. Liszka *et al.* [4] used an internet-capable Palm Tungsten to collect ECG data through Bluetooth. However, as in the studies mentioned above, the mobile device does not play any role more complicated than simply collecting data. Similar industrial efforts to use mobile devices for health monitoring have also been reported. Nevertheless, we believe that our platform is among the first to extend the role of a mobile device and to include a wrist-worn interface.

6 Conclusion and future work

We have presented our design of a wireless body sensor network platform to tackle the challenges in cost, energy efficiency, and user interface for physiological diary applications. By introducing an internet-capable phone as the network center and manager and leveraging its resources, we were able to simplify the sensor node design and improve its energy efficiency drastically. We studied techniques such as scheduled communication, Bluetooth parameter tuning, and protocol optimization to further reduce Bluetooth energy consumption in peripherals. We also included a Cache-watch in the platform so that users can interact with the network conveniently.

However, we are still in an early stage of the design and evaluation of the platform. As we have acknowledged many times in the paper, there are still problems to solve, especially to further reduce the energy consumption of Bluetooth in peripherals. Moreover, although our sensor node is extreme simple, the first version of the PCB design is still sizable to facilitate debugging. We wish to minimize its size for the next version with a better OEM Bluetooth module and a customized battery like that used in Eco [6].

References

[1] Imperial College, London. Body sensor network node. *http://www.doc.ic.ac.uk/vip/ubimon/bsn_node/index.html*, 2004.

[2] KC21 Bluetooth Class 2 OEM module. *http://www.kcwirefree.com/docs/KC21_Datasheet.pdf*.

[3] M. Leopold, M. B. Dydensborg, and P. Bonnet. Bluetooth and sensor networks: a reality check. In *Proc. ACM SenSys*, pages 103–113, 2003.

[4] K. J. Liszka *et al.* Keeping a beat on the heart. *IEEE Pervasive Computing*, 3:42– 49, Oct.-Dec. 2004.

[5] K. Lorincz *et al.*, "Sensor networks for emergency response: challenges and opportunities," *IEEE Pervasive Computing*, Oct.-Dec. 2004.

[6] C. Park, J. Liu, and P. H. Chou. Eco: An ultra-compact low-power wireless sensor node for real-time motion monitoring. In *Proc. IPSN*, Apr. 2005.

[7] L. Zhong and N. K. Jha. Energy efficiency of handheld computer interfaces: Limits, characterization, and practice. In *Proc. USENIX/ACM MobiSys*, June 2005.

[8] L. Zhong, M. Sinclair, and N. K. Jha. A personal-area network of low-power wireless interfacing devices: System & hardware design. In *Proc. ACM Mobile HCI*, Sept. 2005.