The PSI Board: Realizing a Phone-Centric Body Sensor Network

Trevor Pering[1], Pei Zhang[2], Rohit Chaudhri[3], Yaw Anokwa[4], and Roy Want[1]

[1] Ubiquity Group, Intel Research, Santa Clara, CA, USA
[2] Princeton University, Princeton, NJ, USA
[3] Motorola Labs, Schaumburg, IL, USA
[4] University of Washington, Seattle, WA, USA

Abstract—**When designing a body sensor network, the mobile phone is a natural design point for data aggregation services. However, there is a missing bridge that allows sensors to communicate with existing commercial phones, even if they already possess sufficient storage and processing capabilities. The PSI board is a small expansion module that interfaces with commercially available cell-phones through a standard MMC/SD slot. Adding this expansion capability allows researchers to extend phone devices with additional capabilities, allowing the devices to easily serve as the hub of a wearable body sensor network. The PSI board has an integrated switching mechanism allowing transparent access to an MMC/SD card, an embedded microcontroller, accelerometer, expansion connector, and an IEEE 802.15.4 radio that can connect to a variety of commonly available wireless sensors. The capabilities afforded by the PSI board enable several new applications that would be difficult to implement given the sensors and capabilities currently found in existing cell-phone platforms.**

Keywords— **Body sensor networks, mobile phones, MMD/SD cards, IEEE 802.15.4, low-power systems.**

I. Introduction

Mobile phones provide researchers with a potential opportunity to use the computational and storage capabilities of these devices to serve as the hub of a wearable body sensor network. In the context of ambulatory medical monitoring, reducing the number of devices being carried, and reusing devices that a patient is already motivated to carry for personal communication is a considerable advantage. However, phones are typically closed systems from a hardware perspective and do not have much in the way of general purpose expansion capabilities, which might impact the industrial design necessary for commercial success. Because of this, it is often difficult to experiment with phones when new peripherals/sensors are needed to be integrated in the system.

The Phone System Interface (PSI) module is a bridge between wireless sensor networks and commercially available mobile phones. The PSI module allows a phone platform to be extended to perform integrated sensing (sensors combined with the phone), or to wirelessly communicate with a body area (sensor) network.

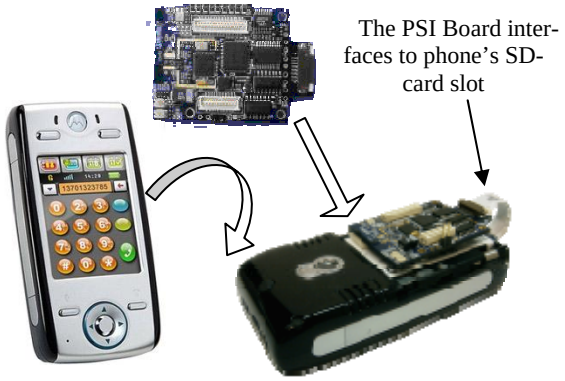

Fig 1: The PSI board attaches to the back of commercially available cell-phones, creating an integrated sensing platform. For deployment the battery panel can be replaced by a module that integrates with the PSI board

The module is a small board designed to interface to commercially-available cell-phone platforms through the phone's MMC/SD-card socket, as shown in Figure 1. An MMC/SD-interface is supported for the phone connection, along with an MSP430 embedded microcontroller, a 3-axis accelerometer, and an integrated CC2420 802.15.4 radio. Additionally, there are expansion connectors that can be used for a variety of purposes, such as adding Near Field Communication (NFC) capability. The prototype system combines the PSI board with Motorola's ROKR E2 and E680i/g mobile phone, designed to run the Linux operating system. Programmatic access to the PSI board is provided through custom kernel drivers. The phone itself provides the Bluetooth and GPRS capability for local and remote communication with devices and networks.

One important consideration used when designing the PSI module was its impact the phone experience. Since the phone is a highly personal device, users have high expectations and will not be willing to adopt the technology if the module interferes with their use of the device as a phone. The three main considerations here are battery life, phone operation, and physical size. By relegating many of the core sensing functions to a separate low-power embedded processor sub-system, the PSI board is able to manage sensing without excessive power consumption, while only adding a small amount of physical size/weight to the phone body.

This paper describes the architecture of PSI board, details several of the subsystems critical to system operation, and provides an overview of three applications models supported by PSI board capabilities. The design and implementation of the PSI board shows that it is possible to extend some commercial cell-phone platforms to create wearable sensing systems, and highlights several design challenges for such systems.

II. RELATED WORK

The PSI board touches upon two major bodies of related work: the phone platform and wearable sensor networks.

The Green Phone [4] is an open-source phone platform that has been recently released and allows developers to update and download any software they choose to the platform. This is a new capability because typical phone platforms do not allow changes to the core software. The Green phone, however, does not enable easy hardware modifications that would be necessary to enable such technologies as IEEE 802.15.4, adding an accelerometer or other sensors. So while the open *software* capability of the device would be useful for integrating with the PSI, it still is closed to hardware expansion

iStuff Mobile [2] is a prototyping environment that allows designers to easily add sensors and other capabilities to mobile phones. Unlike the PSI board, this system communicates all of the data captured from sensors to the nearby infrastructure where it is processed. So, although it works very well for prototyping applications and exploring new areas of interaction, it is not intended as a viable solution to integrate with mobile phone platforms.

The CodeBlue project [7] is just one example of a wearable sensor system built around several mote platforms, several of which support the 802.15.4 standard. A dedicated bridge [6] between a custom wireless sensor network and GSM system demonstrates the feasibility and attractiveness of the basic concept, but does not demonstrate integration with a phone. It also requires the user to carry and manage a separate unit. Integrating sensors with a phone using Bluetooth [1] allows sensors to be connected to a commercially available devices, but this architecture does not integrate very well with common sensor platforms.

IEEE 802.15.4, and the closely related Zigbee standard, are specifically designed to provide lower-power operation for sensor networks and are the radio of choice for the majority of such projects. Using Bluetooth in a sensor network capacity is likely to have an adverse power impact on the sensing nodes. Furthermore, tightly integrating the sensor communication with the existing phone Bluetooth subsystem requires that the phone be active during all communication, which will significantly drain the phone's battery. Efforts are being made in the industry to address Bluetooth's power impact on mobile devices with a new standard called Wibree; however, since Bluetooth is inherently a point-to-point or Master-Slave protocol, it complicates some sensor network architectures where the sensors nodes need to communicate to peers in addition to communicating with the Master node. 802.15.4 does not impose this limitation and so is better suited for such networks.

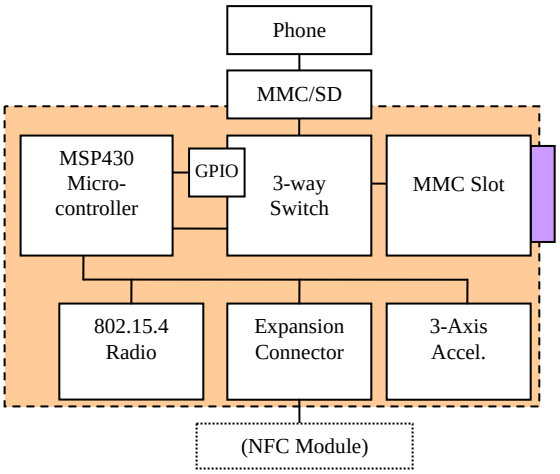

Fig 2: PSI Block Diagram. The basic module connects to a phone through its MMC/SD-card slot.

III. ARCHITECTURE

The basic PSI-board architecture connects together three major system components: host phone, MMC/SD storage card, and embedded subsystem. One key contribution of the PSI system is an understanding of the signaling and switching necessary to connect the PSI board to the host phone platform, without effecting basic phone operation.

A. Host Platform

The PSI system is designed to work with a series of commercially available Linux-based cell-phone platforms, primarily the Motorola E680i/g and ROKR E2 phones. These phones are based on the Intel/Marvel PXA-27x family of ARM-architecture processors, running at speeds of up to 400 MHz. Internal to the phones there are 32 MB of DRAM and 32 MB of FLASH. MMC/SD expansion cards are also available in sizes of at least 2 GB. The phones run an embedded version of the Linux 2.4 operating system.

The PSI board uses the same hardware interface as a MMC/SD card. The Linux PSI interface is implemented as a dynamically loadable kernel module that replaces the

default MMC/SD kernel module. Custom modifications were required to the phone startup sequence to prevent the standard drivers from loading.

The kernel PSI driver appears to user space applications as a special interface that allows basic communication between the user programs and the PSI firmware. The basic abstraction is a channel metaphor between virtual software channels and the individual hardware resources (e.g., radio, accelerometer, and expansion connector). A dedicated software daemon, PSID, runs on the phone and provides a socket interface for other programs. Many of the primary prototype appellations are written as Java Midlets, using a standard socket connection to the PSID.

B. Mechanical Attachment

The basic dimensions of the PSI board are 47x35x8 mm, while the battery for both the E680i and ROKR E2 measure around 55x35x5 mm (with a battery capacity of approximately 780 mAh). This size allows the PSI board to be attached to the back of the phone, on top of the battery. Conceptually, the PSI board could be built into a combined battery & expansion pack – a solution that would require more extensive industrial engineering, but permit the electronics to be supported within an integrated plastic housing.

To interface into the MMC/SD card slot on the phone, a small ribbon cable is used to feed from the slot to the card as shown in Figure 1 (e.g., mounted on the back of the device where the battery is). This cable can be easily constructed out of a standard MMC/SD-card to trans-flash adaptor card, which provides a compatible mechanical design that fits into the MMC/SD card slot.

C. Switch detector, GPIO module, Signaling

The core of the PSI board is a switching mechanism that arbitrates connections between the host phone platform and either the embedded microcontroller or MMC/SD card. There are three main operating modes provided by the switch: MMC, MSP, and CROSS. Each mode is entered through a separate signaling mechanism, described below, and is always initiated by the phone device. The switching mechanism itself uses a set of analog switches between the various components, allowing for transparent bi-directional communication with minimal signal impact.

When the system is in MMC mode (Switch Mode HIGH), all signals from the phone are routed to the MMC card, allowing transparent access to the card's contents. To trigger a switch to MSP mode, a monostable device, which turns a level-signal into a pulse, is used to detect an out-of-spec signal condition on the phone's MMC/SD control lines, highlighted in Figure 3. This signaling is similar to the

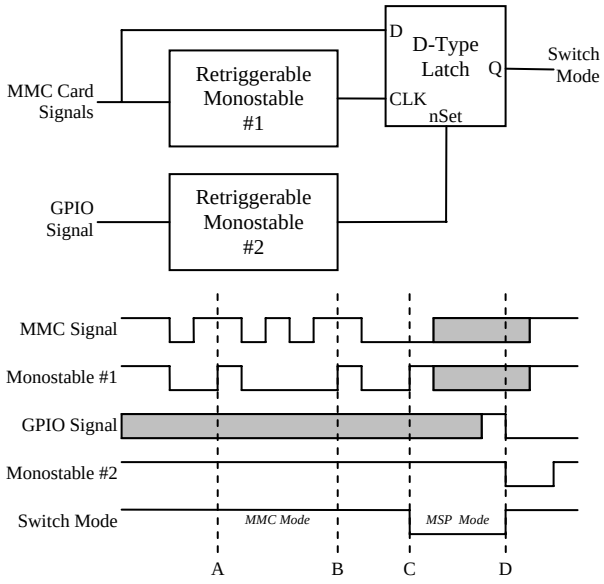

Fig 3: Simplified circuit diagram of switching mechanism, with timing diagrams for set & reset pulses. A) Normal short pulse is ignored because MMC is high when monostable fires. B) Multiple pulses are ignored because monostable is retriggerable. C) Single long pulse clocks low signal into latch, flipping switch. D) Falling edge on GPIO line sends limited duration pulse, switching back to MMC mode.

standard BREAK signaling on a serial UART line, wherein a data signal is held in a constant state for an extended length of time. Since the switching mechanism is out-of-spec for normal MMC/SD operation, this condition is only triggered by a specific intent to switch modes and will not adversely affect the card's normal operation.

While in MSP mode (Switch Mode LOW), the phone can use the MMC/SD interface to communicate with both the MSP and a dedicated GPIO module. Normal operation consists of communication with the MSP module, while the GPIO module is used for debugging (control over LEDs), in-circuit programming (described below), and switching out of MSP mode back to MMC mode. This is shown in Figure 3. While in MSP mode, the MMC/SD card is disconnected from the phone's data-bus, allowing it to retain any transaction state and quickly resume operation. In addition to the basic MSP mode, there is a special case used to program the MSP processor's program FLASH, detailed below. While in MSP mode, any spurious switch detect signals (used to switch *into* MSP mode) are implicitly ignored. Furthermore, a second monostable device is used to turn the GPIO switch-to-MMC-mode signal into a finite duration reset pulse, since as soon as the device switches back to MMC mode the phone can no longer communicate with the GPIO module.

In the current PSI implementation, the basic switch from MMC to MSP mode takes approximately 1 ms while switching MSP to MMC takes approximately 32 us. The speed of the switch into MSP mode is governed by the time of a single data-bit while running at the slowest allowable clock speed (100 kHz), which could theoretically be as low as 100 us. The speed of the switch into MMC mode is limited by the time necessary to issue a 2-byte SPI write to the GPIO module to change the state of the switching GPIO line. These switching times affect the efficiency with which the system can switch between modes, along with any software driver overhead on the phone side required to multiplex access to the shared resource.

D. Microprocessor Subsystem

The microprocessor subsystem is based around the MSP430 microcontroller, and is very similar in design to the many standard wireless sensors, such as the Telos-B [5]. Similar to many of these platforms, the PSI board also provides an expansion connector useful for attaching expansion boards, which can be used to experiment with new sensors or other capabilities.

The main advantage of the MSP processor in this context is its relative high-capability and low-power consumption. This is an important requirement so the PSI can collect and process data while keeping the phone in sleep-mode and thus avoiding draining the phone's battery. Although similar in hardware to a wireless sensor node, the phone/PSI system is different in that the PSI board shares the battery with the phone host, which means the user only has to remember to charge a single device.

The MSP runs a custom embedded program that provides basic communication between the host processor and the underlying data sources/sinks (such as the accelerometer and radio). Since the PSI board is designed to work in conjunction with the phone's host processor, most computation tasks can be relegated to the more capable host, removing the need for task-specific programming of the embedded platform.

IV. PHONE-PSI INTERFACE MODES

This section describes the operating conditions of the major modes of the PSI system, while the previous section described how the system was able to switch between the various modes. A Linux device driver handles arbitration between the various modes by using software mutexes to control access to the single physical resource and presents multiple interfaces to higher-level software layers.

A. MMC Mode

The MMC mode places the system in a default access mode where the phone can access the contents of the MMC/SD card just as if the card were inserted directly into the phone. Data is accessed through the standard device drivers for the phone and is completely transparent to higher-level software devices. Furthermore, this mode provides a mechanism for the phone to efficiently access data stored directly on the card by the CROSS mode, described below (for example, while the phone was in a sleep state).

There are two main limitations for the communication channel between the MSP processor and the host. First, MSP cannot signal the host when in MMC mode because all the available signals on the MMC/SD connection are used for active communication; therefore, in order for the phone to detect/receive any signal from the MSP subsystem, it must either remain in MMC mode whenever idle or at least occasionally switch to MMC mode to detect if any data is available. Second, the MSP cannot wake the host processor while it is asleep. This is a limitation of the specific processor hardware pins used to communicate between the host processor and MMC/SD subsystem.

B. MSP Mode

In MSP mode, the phone can talk directly to the MSP processor or the dedicated GPIO module. In this mode, it can participate in bi-directional data-transfer with the MSP device using the SPI serial protocol. Additionally, a signaling IRQ line can be used to indicate an interrupt condition back to the host. The MSP mode could be the default idle mode for the system, allowing the phone to quickly switch to MMC mode if it needs to access card data. This would allow the MSP to signal an interrupt condition, such as incoming data ready, to the phone if necessary.

One limitation presented by the current implementation is a restriction on the data-transfer bandwidth between the phone and MSP devices. The MSP processor only operates at 6MHz, which, although allows it to have a relatively low power consumption, does not allow it to process the data from the PXA quickly. Additionally, the minimum data-rate supported by the PXA hardware is 300 kHz, which is still fast enough that it would overwhelm the MSP FIFO hardware. Therefore, the data-transfer mechanism on the MSP uses a slower bit-bang interface that allows the data-rate to be appropriately throttled. There are several possible technical solutions to this problem, ranging from a faster embedded processor (which might increase its power consumption), or a dedicated hardware FIFO module (increasing system complexity and cost).

C. CROSS Mode

In this mode, the PXA effectively disconnects from the MMC bus and allows the MSP to talk directly to the MMC/SD card. This mode is useful because it allows the PSI board to continue operation even though the host phone system may be shutdown or asleep, and it also provides a more efficient data-path for storing data. Without it, the system would have to transfer data from the MSP to the PXA, and then from the PXA to the MSP, creating a communication bottleneck between the Phone and PSI board. Additionally, this capability can be used to store data directly to the storage card without having to activate the higher-power phone platform. In order to manage the data storage on the MMC card, standard filesystem partitioning can be used to create separate storage areas for MSP data and regular phone data.

In CROSS mode, the MSP talks directly to the MMC/SD card, and the phone is disconnects from the channel. Two signals remain active between the MSP and phone to facilitate signaling for returning to MSP mode (and then eventually to MMC mode, if necessary). Returning from this mode to MMC mode requires that the phone reinitialize its MMC card interface since the previous state of the phone's transactions will have been lost. CROSS mode is entered by the phone first switching to MSP mode, if necessary, and then issuing a command to the MSP to enter CROSS mode, at which point the phone disconnects from the MMC/SD electrical interface.

D. Inline programming

A sub-mode of the MSP mode is used to directly program the MSP from the phone device. This is useful because it allows the phone to update the MSP device without requiring it to be removed and plugged into a dedicated programming device. This could even be done in the field by delivering the code update through a GPRS connection. To implement this mode, the phone communicates with the embedded GPIO module to place the MSP in a reset state and applies the necessary control sequence to place the device in the programmable state. The phone can then program the device, and reset it to execute the new firmware.

Once placed in the programmable state, the MSP needs to receive programming data using a standard serial protocol at 9600 BAUD, which is not supported by the phone hardware on the relevant pins. Therefore, similar to the technique used for implementing a slow SPI connection, the phone uses a bit-bang serial driver to send data at the necessary 9600 BAUD to the MSP for programming. Fortunately, 9600 BAUD is slow enough that the phone's software driver can manually generate the necessary signals within the required tolerance.

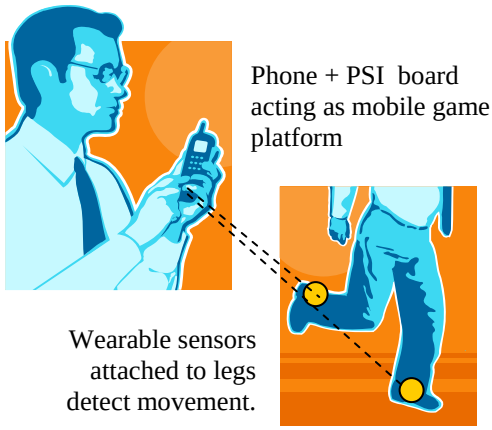

Fig 4: The PSI/Phone device and wearable sensors could provide the platform for a new class of mobile games that enable physical movement, such as a walking motion, to control actions in the virtual game world.

V. APPLICATIONS

The PSI board is a flexible platform that can support a wide variety of mobile and wearable sensing applications. Several of these target applications are described below.

A. Wearable Activity Monitor

One primary use of the PSI board is as a wearable activity monitor, a device that could monitor and report on the user's daily physical activity patterns. Some current work on activity monitoring [3], uses a separate sensing component, which, although it has some advantages, ultimately requires users to manage a separate mobile device. Since the PSI is tightly integrated with a user's cell-phone, it would be very easy for them to keep it with them most of the day. By making the wearable sensor ecosystem more accessible to end users, this design should increase the number of people able to use lightweight sensing applications.

Technical considerations of note are the ability to use the CROSS mode to store sensor data directly to the MMC card. In this case, off-line processing would be used (e.g., a desktop inference engine triggered when the phone is docked with a power source) that would be too processing intensive and power hungry to run on the mobile device itself. For some applications, the phone's processor can be used to perform the activity inferencing on the device, allowing for real-time interaction based on activity – possibly reminding users when they should, or should not, be doing some specific activity.

B. Physical Gaming

A network of wearable sensors attached, for example, to a user's arms and/or legs, enables a new class of physical game that would allow people to interact with the phone without being restricted to the phone's limited input capabilities. A racing game, for example, could be controlled by how fast somebody can shuffle their feet up and down (Figure 4), or arm and leg movements could control a fighting game. This capability would be similar to systems that use a wireless joystick to control a PC game – except the PSI system would enable the complete wearable sensor mobile experience.

The technical considerations of these kinds of systems require the management of the wireless sensor network, since the wireless nodes need to be powered, and wouldn't continually connected to the cell-phone's battery (unlike the PSI board itself). Furthermore, the system would need to be configured to associate the sensors with the mobile device. These basic power and association problems are similar to those found for wearable healthcare systems, but fortunately a wearable game would likely be less battery-critical because it would not necessarily be used all the time.

C. Gesture Recognizer

Another application of the Phone + PSI platform is to use an attached Near Field Communication (NFC) reader combined with accelerometer readings to facilitate connections between devices. The system would use NFC, which enables close proximity based communication between devices, to allow a user to specify *which* device to connect to with their phone, and then a gesture, detected by the accelerometers on the PSI board, indicating *what* should happen when the devices are connected. For example, the user could scan their home stereo system with the mobile phone, and then do a "play jazz" gesture with their device to initiate the music playing.

Currently, neither NFC nor accelerometers are commonly available in mobile phone platforms. Some platforms are starting to have NFC capability, but they still generally do not have accelerometers. The PSI board enables researchers and developers to experiment with capabilities and interaction models that otherwise would be very difficult to explore. In time, the research will demonstrate the usefulness and viability of these technologies, and they can then be incorporated into the standard phone platforms. Until that time, the PSI board is necessary to enable the underlying experimentation.

VI. CONCLUSION

The PSI board provides an integrated mechanism for extending the sensing and communication capabilities of commercially available cell-phone platforms. The device integrates seamlessly with existing phone applications and usage models, enabled by hardware and kernel switching abstractions. This switching mechanism requires a custom detection mechanism that automatically detects a special out-of-band switching signal that does not interfere with standard MMC/SD card operation.

By adding embedded processing, an accelerometer, and wireless-network communication capability without interfering with the normal use of the MMC card slot, the PSI board enables a variety of new applications that cannot be easily prototyped with existing phone platforms, which are typically closed to expansion. The PSI board, designed for extensibility, enables new classes of applications that would otherwise require users to carry a separate device with them.

REFERENCES

1. Baker, J. P., Bones, P. J., Lim, M. A.. *Wireless Health Monitor*. In Proceedings of the Electronics New Zealand Conference (ENZ Con 2006), November 13-14, 2006, Christchurch, New Zealand..
2. Ballagas R., Memon F., Reiners R., and Borchers J., iStuff Mobile: Prototyping Interactions For Mobile Phones In Interactive Spaces
3. Consolvo S., Everitt K, Smith I., Landay J.: Design requirements for technologies that encourage physical activity. CHI 2006
4. http://www.trolltech.com/products/qtopia/greenphone
5. http://www.xbow.com/Products/productsdetails.aspx?sid=126
6. Kumar A., Fazlur R. System for Wireless Health Monitoring. Sensors for Industry Conference, New Orleans, Louisiana, 27 January 2004.
7. Shnayder V., Chen B., Lorincz K., Fulford-Jones T, Welsh M.,, Sensor Networks for Medical Care, Harvard University Technical Report TR-08-05, April 2005.

Address of the corresponding author:

Author: Trevor Pering
Institute: Intel Research
Street: 2200 Mission College Blvd. MS RNB-6-61
City: Santa Clara, CA
Country: USA
Email: trevor.pering@intel.com