

An Experimental Hardware Extension Platform for Mobile Devices in Smart Spaces

Marios Michalakis
ECE Department
Northeastern University
Boston, MA 02115

Dimitris N. Kalofonos
Pervasive Computing Group
Nokia Research Center Cambridge
Cambridge, MA 02142

Bahram Shafai
ECE Department
Northeastern University
Boston, MA 02115

Abstract— *Mobile devices will increasingly assume a central role in the user interaction with smart environments. Currently there is an intense research interest on how to use mobile devices to make this interaction intuitive for non-expert users. For this reason, a number of new hardware enhancements are proposed. However, prototyping new hardware technologies and interfacing mobile devices can be a tedious and time-consuming task. In this paper, we present the Smart Sleeve, an extensible experimental platform that allows the easy addition of new sensing and communication technologies to mobile phones to facilitate research in pervasive computing. The Smart Sleeve comprises of a core system which can be extended with add-on modules that offer the new functionality. We present our experience designing and building the smart sleeve and some interesting add-on modules that add example functionalities, such as a “touch”-interface (Near Field Communication), location-awareness (indoor and outdoor location sensing), and motion (robotics) to commercially available mobile devices.*

Index Terms—smart covers, mobile enhancements, Near Field Communication, indoor-outdoor location, robotic interfaces.

I. INTRODUCTION

COMMERCIALY available mobile devices have evolved to feature significant computing and storage capabilities, as well as a multitude of wireless interfaces, e.g. Bluetooth, Infrared and WiFi. Currently, there is an intense research interest in enabling such devices to be used by non-expert consumers to intuitively interact and control their smart spaces. Often, this is achieved through the use of new hardware technologies that enhance their existing capabilities. Example enhancements include RFID readers, new radios (e.g. UWB, ZigBee), barcode scanners, lasers, and infrared cameras.

Experimenting with new hardware interfaces for mobile devices, when conducting research in pervasive computing, can be a difficult task. Commercially available mobile devices feature only a small number of sensing and networking technologies. Moreover, new emerging technologies appearing every year are tightly bound to the industry’s production cycle. This in turn affects researchers and developers who need to wait until a product featuring a new technology is released in order to use it in their experiments.

Therefore, it would be desirable to have a flexible and extensible common platform with the ability to easily add new communication and sensing enhancements to commercially available mobile devices.

In this paper we present our experience in building such a platform that we call the *Smart Sleeve*. The aim is to create an extensible hardware platform to facilitate research with new interfaces/sensors in pervasive computing. The smart sleeve is comprised of both hardware and software elements. As a system, it is divided in two parts: the core architecture and the add-on modules. The add-on modules attach to the smart sleeve and provide additional functionality to the mobile devices. We present our experience building the smart sleeve and some interesting example add-on modules. The first add-on module adds Near Field Communication (NFC) that enables an intuitive and inherently secure two-way “touch” interface. The second adds indoor and outdoor location sensing that enables context-aware interaction. Finally, the third one adds a robotic interface through which motion is added as part of the user interaction with a smart space.

The rest of the paper is organized as follows: Section II presents a brief overview of related work; Section III discusses the motivation behind the development of the smart sleeve and the add-on modules we chose; Section IV describes in detail the system design; Section V summarizes our prototyping experience; finally, Section VI presents our conclusions.

II. RELATED WORK

Related to creating extensible embedded platforms for mobile devices, one of the most notable existing approaches is the *HP iPAQ Expansion Pack*, which was developed to accommodate PC Cards, extra memory, or even a GPS receiver. A Linux-based experimental platform using it was created in the *handhelds.org* project [1], which aimed at encouraging and facilitating the creation of open software solutions for handheld computing platforms. Furthermore, there is the *Smart-Its* project [2]. Smart-Its are self-contained, stick-on computers that attach to everyday objects. These augmented objects become soft media, enabling dynamic digital relationships with users and each other.

Related to our NFC add-on module, there are a number of different ways that RfId and NFC technology have been utilized so far. NFC's very short operating distance is ideal for providing touch-based services for a smart space environment. An interesting application of NFC is to enable location-based mobile games [3]. Another approach to utilizing NFC in smart spaces is the iTouch project [4], where the user exchanges information and sets up elements of a smart space, such as a media server or an access point, by simply touching them.

Related to our location-aware module, there are a number of solutions that add location sensing to mobile devices. Most of them come in the form of GPS modules with Bluetooth interface and an application running on the host side that combines mapping with positioning coordinates. Nokia Wireless GPS Module LD-1W and HP iPAQ Bluetooth GPS Navigation System are among the more popular ones. Moreover, one approach relevant to our work is the HotTown project by Theo Kanter [5], where the author introduces a novel and scalable service architecture for context-aware personal communication. In another publication [6], Yilin Zhao discusses location technologies and proposes three methods for finding the location of a mobile phone.

Finally, related to our robotic module, there are interesting approaches mostly through university projects. One group from the University of Freiburg was able to integrate Pocket PCs to off-the-shelf toy robots and demonstrate that the robots could autonomously play soccer [7]. Also Fujitsu Laboratories have developed MARON-1 [8], a mobile phone-controlled robot for the home. It is envisioned that MARON-1 could be used for monitoring homes or offices at night or for checking up on persons requiring special care and monitoring.

III. MOTIVATION

Consumer-oriented smart spaces are becoming a reality. An example of this trend can be found in the Digital Living Network Alliance (DLNA) [9]. The DLNA is an alliance of leading companies in the Consumer Electronics (CE), mobile and Personal Computer (PC) industries to create interoperable networked consumer devices. Commercially available mobile devices are expected to assume an important role in the user interaction with such smart spaces, because they are personal, have adequate computing and storage capabilities, and feature a multitude of communication interfaces. One of the main motivations behind our smart sleeve project was enabling research in this area. We envision that the smart sleeve can be used as either an experimental platform for integrating new technologies to commercially available mobile devices, or as a means of developing compelling applications utilizing those new technologies.

One add-on module that we chose for our smart sleeve adds NFC capabilities to mobile devices. We chose NFC because it is a standards-based, short-range wireless connectivity technology that enables simple and secure two-way interactions among devices. The vision of NFC is to enable users to access content and services in an intuitive way by

simply touching smart objects and connecting devices just by holding them next to each other [10].

An additional add-on module that we chose for our smart sleeve adds both indoor and outdoor location sensing capabilities to mobile devices. Our motivation was location awareness. Location awareness can be used, for example, for navigation and mapping, workforce tracking and finding points of interest.

Finally, as a last mobile enhancement, we chose to explore a robotic add-on module. Robots can bring a unique and powerful dimension in smart space user interaction, i.e. motion. An add-on module would allow a mobile device to physically attach to a robot and communicate with it through the smart sleeve. This would enable networked devices (e.g. cameras, printers, displays) to actually move. The mobile device would act as a computing and communication gateway, through which a user could control a robot locally or remotely.

IV. SYSTEM DESIGN

A. Overview

The Smart Sleeve is a system that serves as the interface between a mobile device and various add-on modules. The different add-on modules attach to the smart sleeve to form a complete embedded device. This in turn is attached to a mobile device such as a cell phone through a USB-based interface. An add-on module consists of the new technology that we wish to integrate, such as a new sensor, a novel wireless communication technology, or an indoor location chipset, along with the required peripherals to interface with the smart sleeve. The add-on module carries also the firmware code in an external Flash memory, which is downloaded to the microcontroller of the smart sleeve upon power up. This added feature provides a solution to the common issue of manually reprogramming a microcontroller every time its peripherals change. Figure 1 depicts the general concept of the smart sleeve.

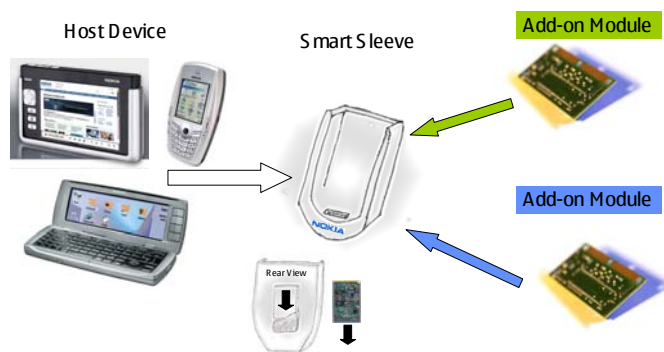


Figure 1 – The smart sleeve concept

A block diagram of our smart sleeve architecture is shown in Figure 2. The smart sleeve itself provides the necessary interface and computational resources for the add-on modules

to connect to the host device. On the host side, the system provides the necessary low-level USB support. This is part of the core system, and once developed, remains the same for all add-on modules. Specific to every add-on module is a server that exposes an API to higher-level applications. No hardware or firmware modifications are required by the user when attaching different modules. New firmware code is downloaded on the smart sleeve from a flash memory on each module as a result of the boot loading capability that the system provides. Therefore, our system has been developed such that the end user only needs to physically switch add-on modules and run a different application on the mobile device.

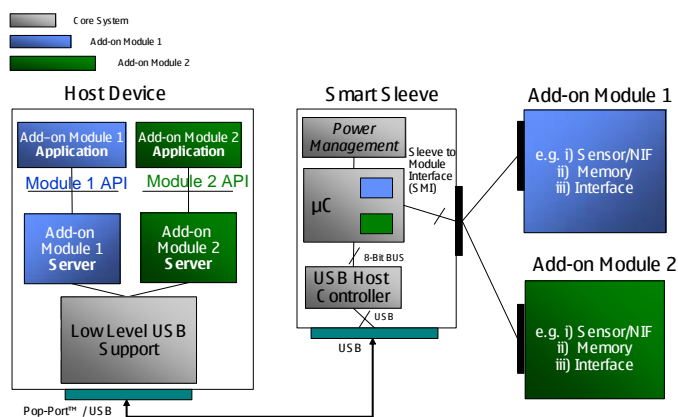


Figure 2 – Smart Sleeve System Block Diagram

B. Core System

The core system is the part of the system that remains the same for all add-on modules. The core system resides in part in the mobile host device and in part in the smart sleeve. In the mobile host device it consists mainly of the low-level USB support and the related hardware to connect to the smart sleeve. In the smart sleeve it consists of the hardware and parts of the firmware.

The hardware core system in the smart sleeve consists of an 8-bit AVR microcontroller (MCU), a USB Host Controller, and other secondary subsystems. The microcontroller that we chose was Atmel’s Atmega128L [12] because we required low power operation, rich peripheral features, boot loading capability, and ample available support. The USB Host Controller we chose was the AT43USB380 from Atmel [13]. The USB architecture is not symmetric [14]. A host needs to be present in order to initiate and control the communication with several USB devices. Since most mobile devices ship as USB Devices that connect to a USB Host (e.g. a PC), the smart sleeve needs to provide the functionality of initiating and controlling the USB communication. The AT43USB380 provides three modes of operation: OTG, host, and device. It comes with 32-/16-/8-bit system bus interface with DMA capability. It is supported by a comprehensive set of USB software suite including standard USB class drivers. A negative point is that it has increased current consumption that may reach 60mA. Finally, the smart sleeve secondary

subsystems include a charging circuitry, a UART interface for low level debugging and quick prototyping, and a JTAG interface [15] used for firmware programming and on-chip debugging. Furthermore, several voltage regulators along with voltage supervisors ensure the integral operation of the system and when directed by the microcontroller, have the ability to shut down elements of the system that are not in use in order to conserve power.

PIN	NAME	DESCRIPTION
1	Vin	Power coming from the 3.7V Li+ Battery
2	GND	Ground pin of the entire circuit, connected to the ground plane
3	PE3	May be used as Enable Pin used to power up the Add-On Module
4	MOSI	SPI interface pin – Master Output Slave Input
5	MISO	SPI interface pin – Master Input Slave Output
6	SCK	SPI interface pin – Serial Clock
7	nSS	SPI interface pin – Slave Device Selector [Active Low]
8	SDA	I ² C interface pin - Pin PD1 on Atmega128L – Interrupt
9	PB5	May be used as a Reset pin - General I/O
10	UART_RX	PE0 – UART_RX
11	PC0	Used as ChipSelect for SPI Flash Memory - General I/O
12	PC1	Pin # 36 on Atmega128L – General I/O
13	PC2	Pin # 37 on Atmega128L – General I/O
14	PC3	Pin # 38 on Atmega128L – General I/O
15	PC4	Pin # 39 on Atmega128L – General I/O
16	PC5	Pin # 40 on Atmega128L – General I/O
17	PC6	Pin # 41 on Atmega128L – General I/O
18	PC7	Pin # 42 on Atmega128L – General I/O
19	PE4	Pin # 6 on Atmega128L – Interrupt
20	SCL	I ² C interface pin - Pin PD0 on Atmega128L – Interrupt
21	UART_TX	PE1 – UART_TX

Table 1 – The Sleeve to Module Interface (SMI)

The Sleeve-to-Module Interface (SMI) that we implemented is described in Table 1. There is an SPI bus, an I²C bus, an 8-bit bus, a UART interface, power from the battery, and several I/Os as well as interrupts for control or communication signals. The SPI bus is the most flexible choice when many different types of serial peripherals must be present, and there is a single controller. It operates in full duplex (sending and receiving at the same time), making it a popular choice for some data transmission systems. A particular strength of I²C is that a microcontroller can control a network of device chips with just two general-purpose I/O pins and software. Peripherals can also be added to or removed from the I²C bus while the system is running, which makes it ideal for applications that require hot swappable components. The UART is a very mature and easy to use interface with plenty of support, and is widely used for quick debugging and communication with a host PC. Finally, the power from the battery is regulated on the add-on module side, so that the developer can choose the voltage level and overall power requirements of the add-on module regardless of the smart sleeve.

The *boot loader* provides a self-programming mechanism for downloading firmware on the MCU. This technique ensures that every time a user needs to replace the current add-on module with a different one, there will be no need for manually programming the MCU through the JTAG interface. Instead, the MCU will automate the process of firmware

update, thus making the process of programming the microcontroller transparent to the end user. To enable the boot loader functionality, the add-on modules carry a 128K Flash chip which stores the application code for the add-on module. The boot loader uses the SPI interface to communicate with the external Flash memory and transfer the resident code to the application section of the microcontroller's Flash memory.

The boot loader can only exist on the boot loading section of the microcontroller's Flash memory. When the microcontroller is powered up, the boot loader instructions will execute first and provide the capability of self-programming the application section of the microcontroller through the SPI interface. Upon reset, the boot loader will check to see if there is any viable code in the application section of the Flash memory. If there is none, the boot loader will start downloading the .hex file that is found on the add-on module's external 128 kBytes Flash memory. After the downloading process is complete, the microcontroller will run the code found on its application section of the memory. If however, the boot loader detects code in the application section, it will check to see if that code matches the one residing on the external Flash memory. This check is performed by applying a CRC (Cyclic-Redundancy-Check) to the contents of the external Flash and comparing it to the CRC stored in the EEPROM memory. If there is a match, then the boot loading is skipped and the microcontroller will start normally. Otherwise, the downloading process will take place and the new code from the add-on module will be downloaded to the microcontroller.

C. NFC Add-On Module

The NFC add-on module that was implemented carries a Philips NFC transmission module [16]. The PN511 is a highly-integrated transmission module for contactless communication at 13.56 MHz. As a stand alone IC, it requires external matching elements as well as an antenna which is used for two purposes. First, the antenna is the medium that transfers the information to an active or a passive NFC device; second, it generates the necessary energizing field that will power up a passive NFC device. Passive NFC devices are not self-powered; therefore they require an external power source. The antenna in an NFC device or tag is a conductive element that permits the tag to exchange data with the reader. Passive NFC tags make use of a coiled antenna that has the ability to create a magnetic field using the energy provided by the reader's carrier signal.

One of the challenges in the hardware design was to ensure that the antenna was perfectly matched to the PN511 chip in order to provide maximum performance and avoid reflections of RF fields back to the receiving circuitry. Another important factor to consider is the fact that mechanically the add-on module PCB board would be placed directly above the smart sleeve PCB board that carried the microcontroller and the USB controller. The distance between the two boards is merely 4mm. Special care in the routing process had to be taken to ensure that the high electric field generated by the antenna would not interfere destructively with the main board at the bottom. Also, no metallic object should be found near

the antenna, especially a ground plane layer, as that would have a detrimental effect on the performance of the antenna itself. Our top-layer PCB layout for the add-on module board is shown in Figure 3.

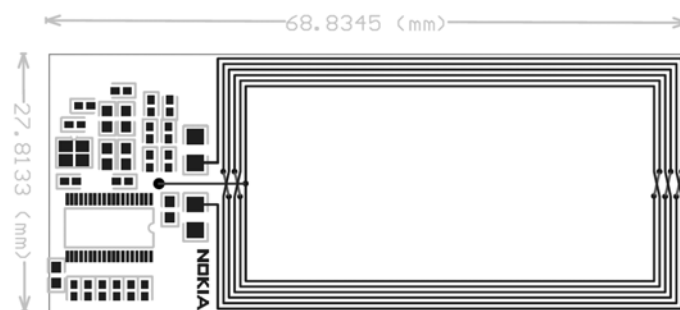


Figure 3 – NFC top-layer (away from mobile) PCB layout

Functionally, the microcontroller found on the smart sleeve communicates with the PN511 chip through the SPI interface and some control signals. Other than the four SPI lines, there is a Reset line used by the MCU, an Interrupt triggered by the PN511 chip, and a control signal to power up or power down the add-on module.

The microcontroller issues commands to the NFC chip such as `send_data` or `read_tag`, the NFC chip executes them and returns the proper data back to the microcontroller. The commands issued to the microcontroller come directly from the mobile device through the user interface of the application. The USB Host Controller initiates and handles all the communication between the sleeve and the mobile device.

The application communicates directly with a server through a well defined API. The API is provided in the form of a class, which has functionality to send commands to the server. Commands include writing data to RFid tags, reading data from RFid tags, sending data through the NFC protocol, and receiving data from a NFC device.

D. Location Add-On Module

The location add-on module combines both indoor and outdoor location-sensing capabilities. For the outdoor part, we used the *ET-301* GPS engine board by *GlobalSat*, which is a small PCB board that includes a SiRF star III high performance GPS chipset. The microcontroller of the chipset is responsible for acquiring information from the satellites, running the algorithms that conduct the timing calculations, and providing NMEA 0183 strings through the serial port [17]. For the indoor part, we have replicated the Cricket architecture [11]. The Cricket uses a combination of RF and ultrasound technologies to provide location information to attached host devices. Wall- and ceiling-mounted beacons placed through a building publish information on an RF channel, much like GPS satellites. With each RF advertisement, the beacon transmits a concurrent ultrasonic pulse. The location module listens for RF signals, and upon receipt of the first few bits, it waits for the corresponding ultrasonic pulse to arrive. When this pulse arrives, the listener obtains a distance estimate for the corresponding beacon by

taking advantage of the difference in propagation speeds between RF (speed of light) and ultrasound (speed of sound).

The PCB board layout is shown in Figure 4. Great care should be taken when laying out and routing a board that carries RF components, especially when two of them need to co-exist without interfering. At the left side of the figure there is the active GPS antenna which needs to remain unobstructed from the rest of the smart sleeve and the host device. This is important since GPS signals are relatively weak signals requiring high performance amplifiers and excellent matching. The ultrasonic transceivers also need to be free of any housing around them. The antenna for the RF part of the indoor location technology is placed on the opposite side of the GPS antenna. It is imperative that no component should be placed near it, especially a ground plane, as it will most likely cause severe detuning.

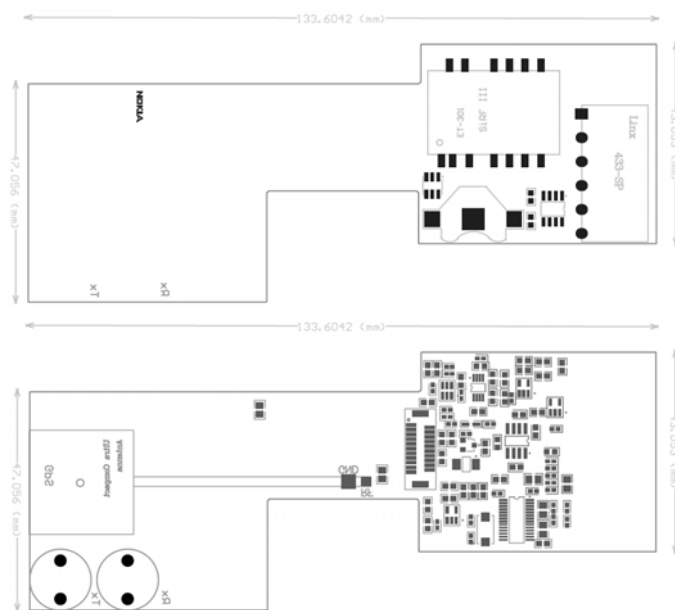


Figure 4 – Location top (away from the mobile) and bottom (towards mobile) layer PCB layout

The microcontroller of the smart sleeve communicates with the CC1000 RF transceiver chip of the cricket through a serial interface for sending and receiving RF data and a 3-wire interface for issuing commands and reading/writing to internal registers. There is also a control signal that powers up or down the entire module and an individual control signal disabling just the ultrasonic part of the cricket. Putting the CC1000 in the low power state is done through firmware. The same is true for the GPS chipset as well. Moreover, there is a line for detecting the incoming RF signal and a line for detecting the ultrasonic signal. Also there is a line used by the microcontroller to generate the 40 kHz square wave and feed it to the amplifier of the ultrasound transmitter. Finally there is the UART interface used by the GPS part and also the programming bus (SPI) that the boot loader utilizes.

The AVR microcontroller in the smart sleeve is constantly listening to the NMEA strings through the serial port. This

makes integration of the GPS module straightforward and leaves room for the more complicated Cricket architecture that requires a number of control I/Os and a separate serial interface. The internal mechanisms and algorithms that are found on the Cricket system are beyond the scope of this paper. It is worth mentioning here that through our common interface we can have both technologies interleaved and function in parallel. However, careful firmware design must be implemented in order to avoid having both subsystems simultaneously operating and dissipating power, even when one of them does not provide useful data.

Similarly to the NFC module, the location module uses a server and an API at the Host system that relays the data from the smart sleeve to the application running on the mobile device. This in turn relates the data to a mapping application similar to the ones found on GPS modules, so that the user knows his position relative to well-known locations. Moreover, the user has the ability to send data, in the form of commands, to the module to control features such as update rate, transmitting power, etc.

E. Robotic Add-On Module

Another proposed module is the Robotic module. In this scenario, an add-on module is developed to communicate with a robot through a standardized interface. This time, the mobile device would be attached to the robot itself and would provide the interface to the user by wirelessly connecting to another mobile device, such as a PDA or a laptop. Figure 5 illustrates the proposed system.

Once the host device is attached to the robot, it will act as the receiving part of a wireless communication scheme. And since most mobile devices support a variety of wireless protocols, controlling the robot would be effortless. If the robot was near the user, the Bluetooth or WLAN protocol could be used to remote control it. Otherwise, for greater distances it could be controlled through Cellular Data such as GPRS or SMS, or through the Internet. This provides the system developer with a great deal of flexibility and variety in the applications that can be developed for that platform.

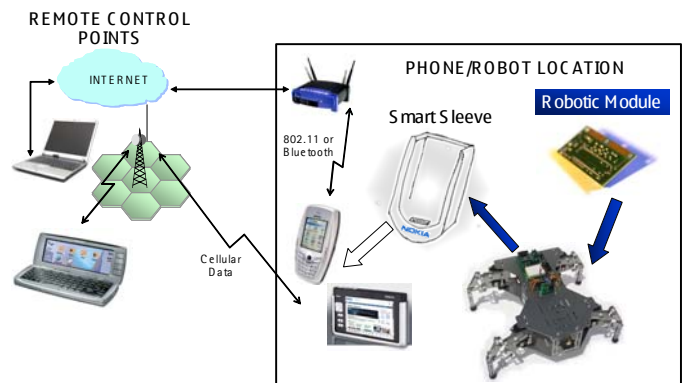


Figure 5 – The robotic add-on module concept

The proposed interface between the robot and the add-on module could be a communication interface or a simple control interface. A communication interface would establish

a serial connection that would issue commands to the robot and get responses back from its sensor system. A control interface on the other hand, would use a number of I/O pins which would change state (high or low), according to the commands issued by the host interface in order to trigger the Finite State Machine of the microcontroller resident on the robot. With this approach there is the inherent limitation of the number of commands that can be issued to the robot and scales as 2^N . However, if it is desirable to transfer the intelligence to the host device instead of keeping it to the robot, only a small number of commands are necessary. That is for example: `move_left_leg`, `move_right_leg`, `read_IR_sensor`, `shutdown_left_motor` etc. More complicated commands such as `avoid_obstacles` or `fetch_tv_remote` can be implemented at the much higher level of the controlling interface, which has the processing power and the user interface for that purpose. An added benefit from the control interface is the inherent immunity to noise generated by the surrounding environment. A serial stream of data is more susceptible to noise generated for example by the motors of the robot or its sensors, than a simple logic high or low.

The AVR microcontroller in the smart sleeve is communicating with the PIC microcontroller of the robot through the defined interface. Therefore, both microcontrollers need to be programmed. For the AVR of the smart sleeve, we need to convert input data packets, received from the USB interface, to control signals (PORTD = 0x7A) that the PIC microcontroller will relate to specific functions. A complete firmware application that serializes both the information obtained by the sensors and the commands sent from the user, according to the USB Host interface protocol, is still pending.

At the application level of the host system, a developer simply writes software according to the API exposed to him without having to worry about reprogramming the microcontroller of the sleeve or of the robot. As mentioned earlier, the API would expose the full robot functionality at the lowest level possible for the developer to have total control of the robot.

V. AN EXPERIMENTAL PROTOTYPE

Currently, we have designed and implemented the smart sleeve along with the NFC and location add-on modules. The robotic add-on module is still in the concept phase. Figure 6 shows the assembled prototypes.

We have also developed a simple mechanical cover to house the PCB boards. The cover is made out of clear plastic and is held together by plastic push-lock screws. This allows for secure tightening of the device, but also for easy access to any electronic component needed. Furthermore, there is an opening at the bottom so that add-on modules that require components to be physically unobstructed, such as ultrasonic transceivers, can be placed externally. It is through this opening that the cable connecting the sleeve to the mobile device passes. The smart sleeve board features a USB mini-

AB connector, to which the cable is connected on the sleeve side. The other side of the cable can have either a Pop-Port™ or a USB mini connector, in order to allow for connection to either the Pop-Port™ of the Nokia 9500 smartphone or the USB mini-B of the Nokia 770 internet tablet. The mechanical design allows the end-user to easily swap these two cable types that we provide, without the need to disassemble the package.

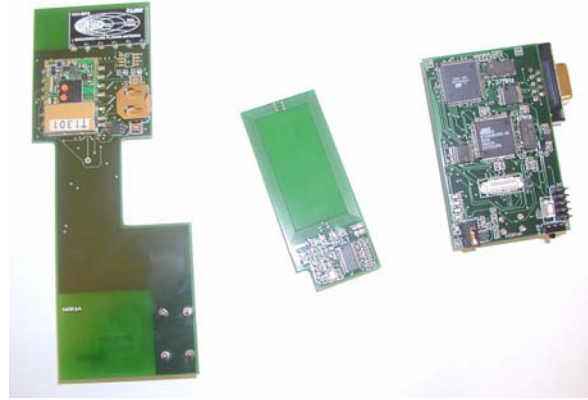


Figure 6 – Actual Assembled PCB Boards. From left to right: Location Add-On Module, NFC Add-On Module & Smart Sleeve.

The hardware part of the smart sleeve has been designed for moderate data rates. Most sensing technologies require only low bandwidth. Nevertheless, due to the wide adoption of mass storage devices and high speed wireless interfaces, it will become imperative in the future to achieve much higher data rates. Our prototype is a system comprised of different interfaces. The USB interface between the mobile host device and the smart sleeve achieves data rates up to 12 Mbps. This dictates the maximum speed achievable between our prototype and any host device. Internally, however, there are other possible bottlenecks in achievable data rates. In our proposed Sleeve-to-Module Interface, the SPI bus achieves data rates up to one quarter of the external oscillator found on board. We have chosen a 7.37 MHz crystal for our purposes. This would give us a maximum of 1.8 Mbps. The I²C bus achieves 3.4 Mbps, according to the High-Speed mode specification. The 8-bit parallel bus could achieve rates as high as 58 Mbps, but with no encoding scheme. Therefore, depending on which interface of the SMI an add-on module uses, the maximum achievable data-rate can be between 1.8 Mbps and 12 Mbps.

On the mobile device side, we need to provide device drivers to enable the communication between the smart sleeve and our host platform. In our case, we chose as a host platform two Nokia devices: the 9500 Communicator [18], a Nokia S80 phone, based on Symbian v7.0s [20] supporting GPRS/EDGE, 802.11, Bluetooth, and Infrared; and the Nokia 770 internet tablet [19] based on the Maemo version of the Debian Linux Operating System [21] supporting 802.11 and Bluetooth.

Regarding power requirements, the total current consumption of our system, which heavily depends on the add-on module, averages at 150mA. We have used a Nokia BL-5C Lithium Ion battery with 850mAh of capacity. This gives the user more than 5 hours of operating time. In

comparison, a typical cell phone usually has talk-time of 4 hours. Low power operation is desirable, however not our main goal in the prototyping phase. As a research platform, the smart sleeve may have relatively high power consumption, but a few hours of operation between charges should be adequate.

Figure 7 shows a snapshot of a demonstration of our smart-sleeve prototype with the NFC add-on module, using the iTouch [4] middleware running on the Nokia 9500 Communicator. The demonstration shows how, through a simple text based application, the user can write a record in a NFC card and then use the same card later to launch a desired application. Part of this process requires the user to acquire data from another device. Once the user selects the read command, the server module that runs on the Nokia 9500 Communicator will transfer the necessary data to the USB driver of the phone. The driver will in turn communicate with the USB Host controller of the smart sleeve and transfer the data to the microcontroller. After all those transactions have taken place, the microcontroller will relate the data sent from the phone to a series of commands that need to be issued to the PN511 chip. As a last step, the PN511 will transmit the data in a specific format to an active or passive device (e.g. a NFC card). Typical dataflow cycles apply to other NFC commands as well.

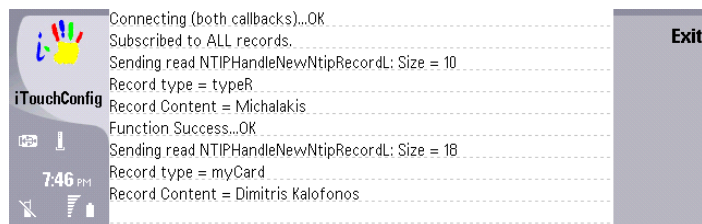


Figure 7 – Demonstration of prototype system

VI. CONCLUSIONS

In this paper, we have presented our experience in building the smart sleeve and some example add-on modules. We have pointed out the importance of rapid prototyping in mobile device enhancements in terms of smart environments and proposed a solution to that issue. In addition, we have chosen to design three prototype modules which will assume a central role in making the interface between smart spaces and non-

expert users more intuitive. Finally, researchers can benefit from our platform and integrate novel technologies to mobile devices with less time and effort.

ACKNOWLEDGMENT

The authors would like to thank Nokia Research Center Cambridge (NRCC) for providing the necessary funding for this collaboration project with the CDSP group of Northeastern University. Special thanks to NRCC members Zoe Antoniou, John Ankcorn, Franklin Reynolds, Brian Avery, Andrew Christian, and Jamey Hicks for contributing to this project with their valuable feedback and ideas.

REFERENCES

- [1] Handhelds.org, <http://www.handhelds.org/>
- [2] Holmquist, L.E.; Gellersen, H.W.; Kortuem, G.; Antifakos, S.; Michahelles, F.; Schiele, B.; Beigl, M.; Maze, R., "Building intelligent environments with Smart-Its," Computer Graphics and Applications, IEEE , vol.24, no.1pp. 56- 64, Jan-Feb 2004
- [3] Rashid, O.; Coulton, P.; Edwards, R.; Bamford, W., "Utilising RFID for Mixed Reality Mobile Games," Consumer Electronics, 2006. ICCE '06. 2006 Digest of Technical Papers. International Conference on, vol., no.pp. 459- 460, 07-11 Jan. 2006
- [4] Z. Antoniou, G. Krishnamurthi and F. Reynolds. "Intuitive Service Discovery in RFID-enhanced Networks". Proceedings of IEEE COMSWARE Conference, January 2006.
- [5] Kanter, T. G., "HotTown, enabling context-aware and extensible mobile interactive spaces," IEEE wireless communications, Vol 9; Part 5, pages 18-27, 2002
- [6] Yilin Zhao, "Standardization of Mobile Phone Positioning for 3G systems," IEEE Communications Magazine, July 2002
- [7] Sven Behnke, Jürgen Müller, and Michael Schreiber, "Playing Soccer with RoboSapien" In Proceedings of The 9th RoboCup International Symposium, Osaka, Japan, paper #97, July 2005.
- [8] Fujitsu Laboratories, MARON-1 press release, October 2002 <http://pr.fujitsu.com/en/news/2002/10/7.html>
- [9] Digital Living Network Alliance, <http://www.dlna.org/>
- [10] Near Field Communication Forum <http://www.nfc-forum.org/home>
- [11] NB Priyantha, A Chakraborty, H Balakrishnan, "The Cricket Location-Support System," Page 1. 6th ACM International Conference on Mobile Computing and Networking (ACM MOBICOM), Boston, MA, August 2000
- [12] Amega128L AVR 8-Bit RISC Microcontroller by Atmel http://www.atmel.com/dyn/products/product_card.asp?part_id=2018
- [13] AT43USB380 USB Host Controller by Atmel http://www.atmel.com/dyn/products/product_card.asp?part_id=3393
- [14] Remple, T.B., "USB on-the-go interface for portable devices," Consumer Electronics, 2003. ICCE. 2003 IEEE International Conference on, vol., no.pp. 8- 9, 17-19 June 2003
- [15] JTAG Boundary Scan Standard - <http://www.jtag.com>
- [16] PN511 Transmission Module http://www.semiconductors.philips.com/acrobat/other/identification/sfs_pn511_rev2.0
- [17] The National Marine Electronics Association (NMEA) <http://www.nmea.org/pub/0183/index.html>
- [18] Nokia 9500 Communicator – <http://www.nokia.com/>
- [19] Nokia 770 Internet Tablet – <http://www.nokia.com/770>
- [20] Symbian OS Mobile Operating System -<http://www.symbian.com/>
- [21] Maemo Development Platform - <http://www.maemo.org/>