uSD: Universal Sensor Data Entry Card

Chunxiao Jiang, Nengqiang He, Student Members, IEEE
Yong Ren, Canfeng Chen and Jian Ma, Members, IEEE

Abstract — *This paper presents an SD-based universal sensor data entry card, named uSD. The prefix "u" of uSD illustrates its most significant characteristic: universality, which means uSD is platform independent and compatible with heterogeneous platforms, truly plug-and-play. In this paper, we relate the design and implementation of uSD, especially about uSD's superiority over SDIO-based method. Additionally, SDK and programmer are provided for third-party to design further applications with uSD. Evaluations about performances of uSD are also developed, like power assumption, CPU and RAM usage. Our practical works including two demos well validate uSD's feasibility and its platform independence. uSD not only can help consumers with portable devices to access WSN conveniently and efficiently, but also are appropriate for researchers constructing testbed.*[1]

Index Terms — Platform Independence; Hardware Expansion; Wireless Sensor Network; ZigBee.

I. INTRODUCTION

A. Background

With the development of wireless communication, sensor technology and low-power embedded system, wireless sensor network (WSN) comes into being and widely applied in various fields. Currently, WSN access is still relatively professional and not approachable to the populace. Consumers should freely access WSN via their own portable devices instead of purchasing special apparatus. However, there are mainly two problems in achieving this. On one hand, the communication schemes in WSN are usually designed according to specific applications, not unified. It leads to the fact that portable devices cannot be equipped with all relevant transceivers in advance. On the other hand, there are various types of CPU architectures and operating systems (we call it as "platform heterogeneity" hereafter) used in portable devices. It is so difficult to develop universal expansion hardware for portable devices to access WSN.

Related Works

Recently, various products of external equipments for portable devices are emerging, as well as relevant research works in consumer electronics area. Secure Digital Input Output (SDIO) is one case of such kind of special purposed hardware expansion via SD card slot [1]. For instances, virtual keyboard, SDIO-ZigBee card and GPS receiver were reported in [2], [3] and [4] respectively. A body sensor network built on phone platform was reported in [5], in which mobile phone can talk directly to the so-called PSI board using SPI serial protocol. However, in such SDIO or SPI method, specific information of Host is required when developing external equipment, such as detail information of Host's processor which is usually not disclosure to developers. Moreover, these methods are platform dependent, which means that for one purpose of expansion, different drivers are requisite varying with different platforms. For example, when releasing the SDIO-ZigBee card, Moteiv [3] also issued drivers of it, which only support Windows CE and XP at present.

B. Motivations

From the analysis above, it can be seen that current consumer electronics area requires a kind of universal and reconfigurable expansion hardware to help common portable devices access WSN. In this paper, we design an SD-based universal sensor data card with RF functions called uSD to expand users' portable devices and enable hosts to conveniently access WSN. uSD is platform independent and reprogrammable to adapt various applications of WSN, well addressing those two problems referred above. The physical and logical interfaces of uSD are completely in accordance with normal SD memory card. Any portable devices with SD card slot can use uSD directly without any added driver, truly achieving plug-and-play. On one hand, mobile terminal inserted with uSD becomes a mobile gateway between WSNs, cellular network and internet. Users can conveniently acquire, upload and share information from WSN around at any moment, which expands the applications of WSN to a large extent. On the other hand, portable device with uSD becomes a personal gateway to conveniently manage all personal electronic devices, making private communication and activity more efficient.

C. Outlines

The rest of the paper is organized as follows. Section II explicitly relates the architecture and implementation of uSD. Then, uSD's platform independence is discussed in Section. In Section IV, circuit design and two demos of uSD are showed. Supports for third party are presented in Section. Then, some evaluations about uSD's performance are developed in Section VI. Finally, conclusion is given.

[1] This work was supported partly by Nokia under National Natural Science Foundation of China (NSFC) grant No.60673178 and No.60873241; National 863 High Technology Projects of China No.2009AA01Z210; and National 973 Basic Research Foundation of China No.2007CB307105.

Chunxiao JIANG, Nengqiang HE, and Yong REN are with the Department of Electronic Engineering, Tsinghua University, Beijing, P.R. China. (e-mail: {jcx08, henq07}@mails.tsinghua.edu.cn, reny@tsinghua.edu.cn).

Canfeng CHEN and Jian MA are with Nokia Research Center, Beijing, P.R. China. (e-mail: {canfeng-david.chen, jian.j.ma}@nokia.com).

Contributed Paper
Original manuscript received 06/18/10
Revised manuscript received 07/09/10
Current version published 09/23/10
Electronic version published 09/30/10.

II. Hardware Design of uSD

A. Architecture

As Fig. 1 shows, the hardware block of uSD consists of two data paths:

- RF part ⟺ Buffer ⟺ Host
- Memory ⟺ Memory Controller ⟺ Host

The first line transports the data from/to WSN; while the second line ensures the memory function of uSD. Through analyzing addresses from Host, the data path selector decides which line to choose. Actually, in normal SD Memory card, there is only one data path: the second memory line.

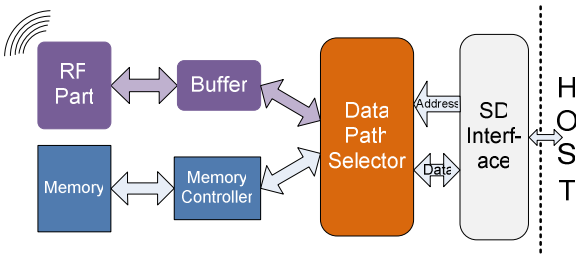

Fig. 1. Hardware architecture of uSD.

SD Interface offers basic SD protocol [6] such as analyzing and translating commands from Host, transferring responses and data. When Host is accessing some files on uSD, **Data Path Selector** receive/send data from/to RF or Memory according to access addresses. As a bridge of data interaction between Host and WSN, **RF Part** helps uSD access WSN. Through modifying the data interface and protocol stack in RF part, the architecture of uSD can support various RF chips, such as Bluetooth, GPS, WiFi, etc. **Memory** is employed to store users' private data, as well as the file system by which uSD communicates with Host. **Memory Controller** is in charge of some basic read/write operations of memory.

B. Implementation

According to the hardware architecture discussed above, uSD is instantiated as Fig. 2 shows below. There are mainly three modules in uSD: FPGA, RF and Nand flash. As the core module, FPGA coordinates the communications between Host and RF chip, Nand Flash. The FIFO designed in FPGA is used to buffer data from/to WSN. MCU in RF can be programmed to run various protocol stacks.

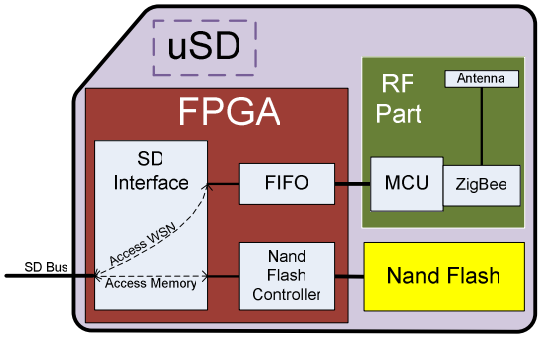

Fig. 2. Implementation of uSD.

III. Platform Independence

As we know, digital devices with an SD card slot support expansion of storage capacity. Files in SD memory card can be directly operated by heterogeneous platforms. Hence the plug-and-play quality of normal SD memory card has already been realized regardless of the low-level details of these files operations. uSD's platform independence is just based on this point. The data interactions between Host and RF chip are all via reading or writing some special files in uSD. This can be done by some interesting tricks as discussed in following two subsections. Firstly, a special file system including some special "virtual files" is designed to differentiate flash operation and RF operation. Secondly, a special cache system is designed to circumvent cache mechanism through skillfully setting the attribute of those "virtual files". This platform independent method has been applied for the U.S. patent.

A. Special "File System"

uSD is formatted with our special file system, the process of which is unlike Host's operating system formatting normal SD memory card. As Fig. 3 shows, some special virtual files are stored in uSD in advance, where "virtual" means these files are only registered in FAT region [7] with names and addresses but without any concrete content. Although they are virtualized, the Host can also see them and access them like normal files. Since the addresses of virtual files are specific; once Host accesses those files, uSD can translate the files operations into RF operations. Hence, such virtual files are just key vehicles of communication between Host and WSN.

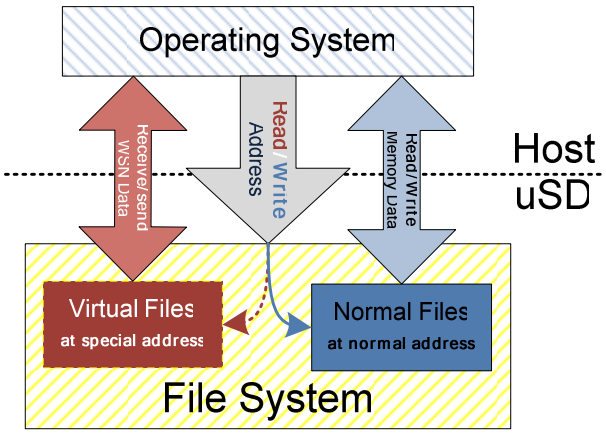

Fig. 3. Special file system in uSD.

When Host wants to receive some new information from WSN, it should read those virtual files. uSD will detect this read operation according to the specific access addresses, then triggering the RF part to acquire data from WSN and putting the data into a FIFO in uSD. Finally, Host will be responded with the data in FIFO. In the eyes of Host, these data are "contents" of the virtual file it accessed. Actually, the data are only new information from WSN. If Host intends to send some information to WSN or adjust some parameters of RF chip, it should write those virtual files. Similarly, the write operation can be detected by uSD, which will then be translated into

corresponding operations according to the content written by Host. Therefore, the data interactions between Host and WSN are all based on those files operations, which is truly platform independent and plug-and-play like normal SD memory card.

B. *Special "Cache System"*

In normal SD memory card, any file access is controlled by OS (operating system) and file system according to the data flow of CPU<->RAM<->SD memory card. When users open some file stored in SD memory card, OS will load the file into Host's internal cache (RAM) and display it via user interface. If users open or modify this file again after a while, OS will execute the corresponding operation to the file in cache instead of directly opening or modifying the file stored in SD memory card. The files in cache will be written back to the memory card after some constant period controlled by the OS or before the card will be disconnected from Host.

However, real-time communications between Host and WSN are via file operations, which require bypassing the Host's cache/RAM. In other words, the Host should directly read/write virtual files in uSD in order to receive/send data from/to WSN timely. Otherwise, if the cache mechanism of OS makes the Host only access the stale file in cache/RAM, uSD will not detect the read/write operation from Host and not trigger the corresponding operations of RF part. This is vividly shown in Fig. 4; the first one illustrates the cache mechanism, in which virtual files in uSD will be loaded into cache/RAM first. The second figure shows the problem brought by cache mechanism, where virtual files updated by WSN cannot be loaded into Host promptly.

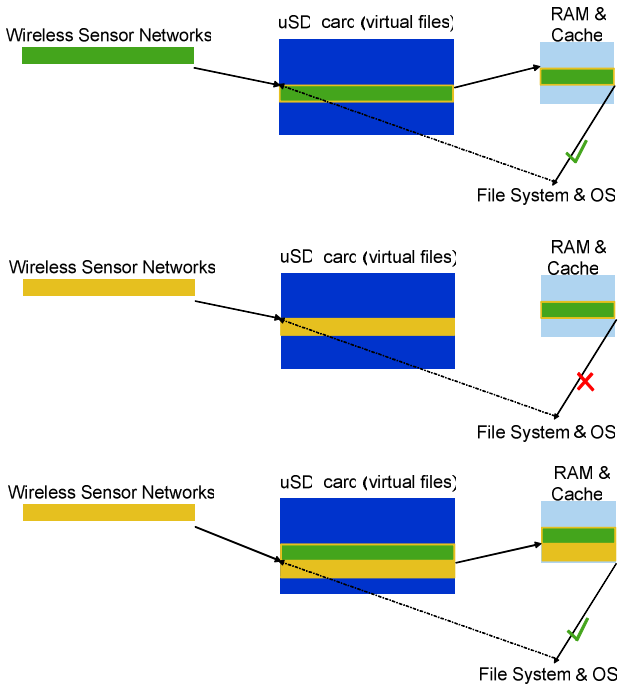

Fig. 4. Cache through description.

The cache in Host is constituted by cache blocks and Host read/write its cache by block. Thus, we can forces the OS to directly read/write virtual files from uSD instead of from cache via carefully setting some attributes of virtual files according to the size of each block and total numbers of blocks:

1. Address interval between two virtual files is larger than the size of each cache block;
2. Number of virtual files is larger than number of cache blocks.

Then, circularly accessing all those virtual files in turn will assure that the OS must fetch files from uSD each time. This is because, the 1st condition prevents Host from simultaneously loading two virtual files into one cache block at one time; while the 2nd condition prevents Host from loading all virtual files into its cache. This is shown in the third picture of Fig. 4 which illustrates that cache updating is not under the control of OS but as a result of special configuration of virtual files in uSD.

To sum up about uSD's platform independence, the expanded RF in uSD is mapped as virtue file in file system, and Host's operations of virtual files are all translated into operations of RF Chips in uSD to accomplish data interactions.

C. *Timing*

We will show by timing diagram below about how to access WSN by operating RF chips through user interface of application in Host, in two aspects: refreshing the Host's user interface and exerting operations to uSD.

Fig. 5 illustrates the procedure about how certain operation to RF Chip is performed. Command inputted from Host's user interface are packed into some request parameters and sent to application in Host, which further interprets them as request of writing files and sends these parameters to SD interface in the form of one file/document. Finally, uSD will control the RF chip according to the content of that file/document, such as transmitting data or adjusting working channel and power.

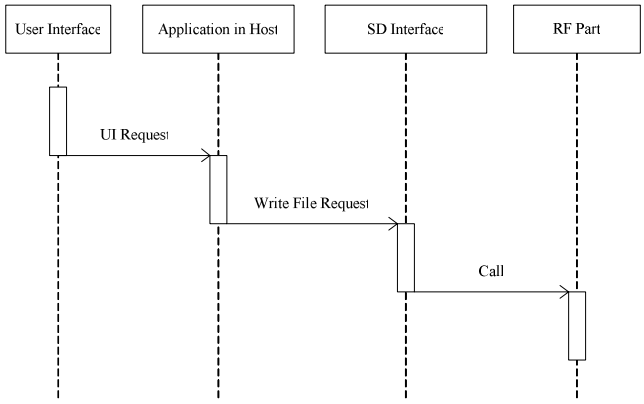

Fig. 5. Timing of operating RF part.

Fig. 6 illustrates how the user interface is refreshed. Application in Host will be activated when Host needs to update its user interface. Firstly, read file request will be sent

to SD interface via file system; then uSD will interpret this request as calling RF chip. After RF chip completes relevant operations, the execution results will be returned to SD interface; finally those results will be transferred into Host and displayed in the user interface.

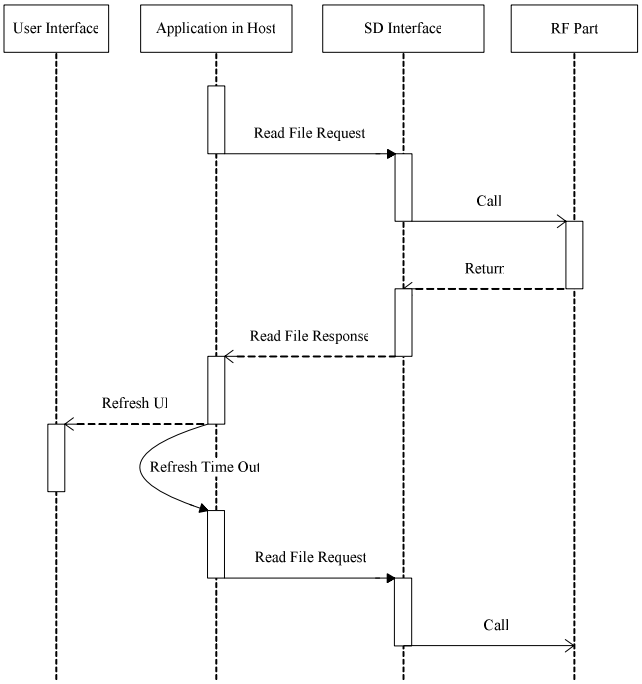

Fig. 6. Timing of user interface being refreshed

IV. PRACTICAL WORK

A. Circuit Design of uSD

Two versions of uSD have been designed: Standard uSD and Mini uSD, as showed in Fig. 7. According to the implementation method described in Section Ⅱ, circuits including FPGA, Nand Flash, and ZigBee transceiver were assembled in standard, mini SD shaped cards. Furthermore, the third version of uSD is under development, in which technology of COB (Chip On Board) will be adopted. TABLE I shows some basic operation parameters of uSD.

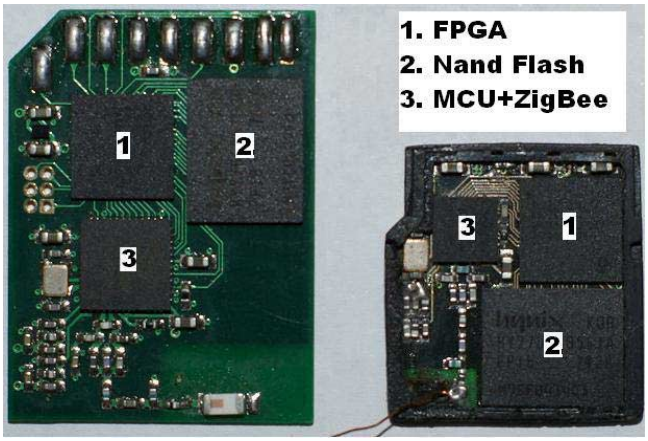

Fig. 7. Picture of standard and mini uSD: top view.

TABLE I
BASIC OPERATION PARAMETERS OF USD

Parameters	Values
Operating Clock Frequency	25M
Operating Voltage	2.7~3.6V
Operation Current	According to transmission power
Data Rate of Interface with Host	50Mbps
Data Rate of RF Communication	128Kbps
Communication Range of RF	According to transmission power
UI response delay	According to the file reading frequency (Typically 1ms)

B. Demo 1: Environmental Information Push System

We design a demo to validate uSD's feasibility and platform independence, called environmental information push system. In the system, the sensor nodes (uNode that will be discussed in SectionⅤ) broadcast temperature information via ZigBee chip in 802.15.4 protocols. While Host receives such information or send command to control sensor node via uSD. We separately developed experiments on three different kinds of Host as listed below.

- Cell phone + Symbian*
- PDA + Linux
- PC + Windows XP

* *Hardware Platform + Software Platform*

As for user interface, an application is also developed with our software development kits (uSDK that will be discussed in SectionⅤ) by Qt which is a cross-platform application and UI framework [8]. Through this application, users can acquire temperature information, control wireless sensor nodes, or talk with other portable devices with uSD. Additionally, the acquired temperature can be uploaded via WiFi or cellular network. The results of this experiment tell that uSD can be directly used in these different platforms. Fig. 8 shows the UI of the application.

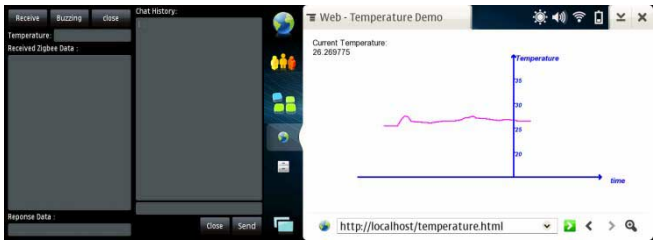

Fig. 8. Screen snapshot of UI.

C. Demo 2: Topology Monitor of WSN

We have also developed an application demo for terminals with uSD to monitor the topology of WSN around, in order to validate uSD's function in sensor network. This demo is also developed based on uSDK, and the mobile phone with uSD card plays a role of sink in the sensor network. The sensor nodes in Fig. 9 are Micaz nodes with XMesh routing protocol

provided by CrossBow Company. To support the networking applications, we develop XMesh routing protocol stack in the phone. Through analyzing the XMesh routing updated messages from sensor nodes; we can monitor the topology of whole network dynamically. The following Fig. 9 shows the picture of user interface of topology monitoring demo. The status of each sensor can be obtained when clicking on the virtualized node in the user interface. Meanwhile, the attributes of sensors in WSN also can be adjusted through this demo, such as transmission channel, power and rate.

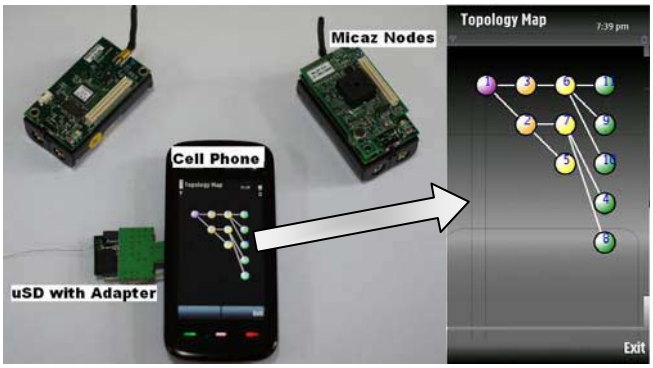

Fig. 9. Topology monitoring demo based uSD.

V. Support for Third Party

In order to consummate the design of uSD for user friendly, some supports for third party are brought forth at the same time, including a programmer for uSD named uProg, an easy-to-use software development kit (SDK) of uSD and an integrated sensor node as discussed below.

A. uProg for uSD

uProg is a programmer, as well as a debugger, which is in a compact size and needs no extra device added. It is designed for downloading and debugging the codes of FPGA in uSD. Fig. 10 shows the architecture of uProg. The MAC and PHY layer of USB is in charge of low-level communication with PC. As USB device class modes, as well as the upper-level protocols, HID (Human Interface Device) protocol [9] is used for simple transportation of control signaling and Mass Storage protocol [10] is used for transportation of amount of data. The Staple Player interprets the STAPL files and controls the JTAG pins of FPGA in order to download codes into it. MMC/SD Controller accomplishes the SD Host protocol with the purpose of successfully accessing uSD. GPIO is multiplexed by STAPLE Player and MMC/SD Controller, fulfilling the data interaction with uSD.

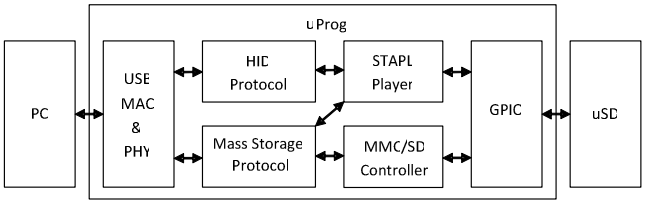

Fig. 10. Architecture of uProg.

According to the architecture described above, we design uProg as Fig. 11 shows. After inserting uSD into the SD card slot of uProg and connecting uProg with PC via USB interface, users can download and debug the codes of FPGA in uSD.

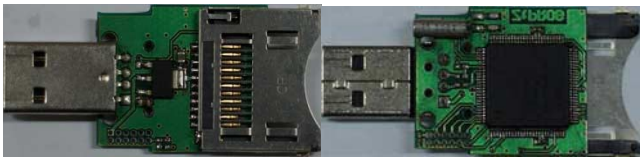

Fig. 11. Picture of uProg: top and bottom view.

B. uSDK: SDK for uSD developers

uSDK is designed by C++ language for users who want to develop uSD centered universal sensor applications. The architecture of uSDK is shown in Fig. 12 below. uSDK lies above uSD card hardware as a middleware in order to provide external sensor data to enable novel local applications or mobile web-service innovations. Since uSD card is recognized as a normal SD memory card in Host, it can be accessed by uSDK through API of file system access provided by operating system. With uSDK, the upper applications can discover the available external sensors, control the uSD card (set RF channel, transmission power, data rate and energy detection level of uSD), communicate with external sensors (send data to external sensors, receive and extract sensor data from these external sensors).

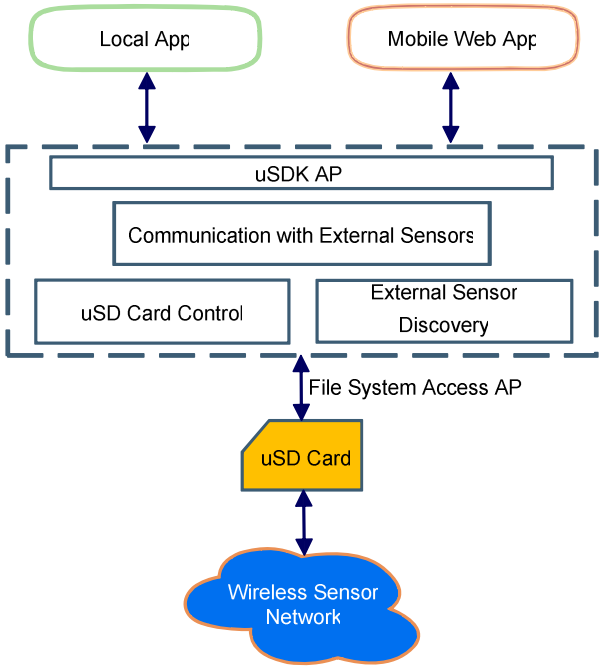

Fig. 12. Architecture of uSD centered universal sensor applications.

C. uNode: Integrated Sensor Node

We also develop a sensor node called uNode, on which compass, accelerometer, alcohol, temperature and humidity sensors are integrated. There is also a low-power ZigBee transceiver in uNode, so that mobile terminal users can communicate with it through uSD card. Additionally, the developers can program MCU on uNode to realize their own protocols. Thus, uNode and uSD can constitute a testbed for researchers to develop experiment of WSN, in which uNode is a sensor of WSN, and uSD can be a mobile gateway or sink point. Fig. 13 shows the picture of uNode. The first demo discussed in Section Ⅳ just hires uNode as the monitor and broadcaster of environment information. Recently, we are developing another demo with uNode embedded in back-cover of cell phone or PDA to detect whether users are drunk before they drive a car, using the alcohol sensor on uNode.

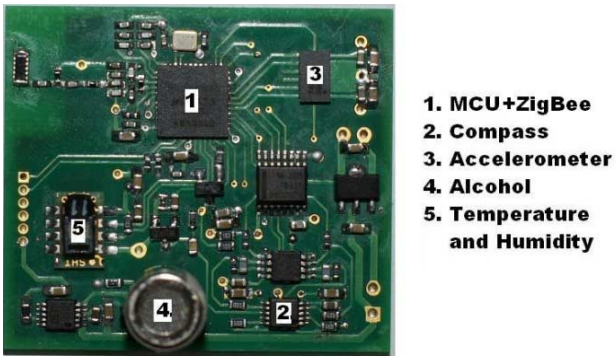

Fig. 13. Picture of uNode: top view.

VI. Evaluations

A. Power Benchmarks

Power measurements of uSD are made using the Energy Profiler, a standard software tool provided specifically for measuring energy use of applications and external hardware. The profiler measures parameters like battery voltage and current approximately every half of a second, storing the results in memory card. Fig. 14 shows the evaluation results of power consumption by uSD. The idle state means that testing phone is without any application on and without any external equipment plugged in. While the evaluation state means the phone is with uSD plugged in and running application designed for evaluation.

The average power consumption is 0.62Watt when the phone is in idle state. Three different states are tested, in which uSD's transmission is with +8dbm. Firstly, it should be mentioned that the net power consumption by uSD is the value in evaluation state subtracts that in idle state. It can be seen from the figure that uSD's receive state (averagely 0.25Watt) consumes more power than transmission state (averagely 0.11Watt). This is because when uSD is receiving data from WSN, continuous files reading operation is needed to realize cache through as discussed in Section Ⅲ.B. In order to avoid repeated data transmission, continuous file writing operation is not allowed when sending data outside.

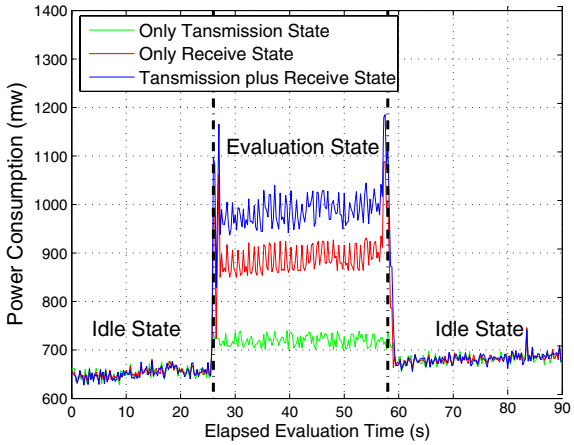

Fig. 14. Power consumption evaluation.

We develop an experiment to measure the transmission radius of uSD varying with the transmission power of ZigBee chip on uSD, as the dotted black line shows in Fig. 15. In our experiment, the transmitter's TX power is tuned at 18 different points from +8dbm to -42dbm, and the corresponding receiver's RX sensitivity is set as -98dbm. The receiving ranges of uSD are also recorded in the process, which are basically same with its transmission ranges. From the figure, we can see that the lower limit of transmission radius is 1m, while the upper limit is 45m. The blue line in Fig. 15 shows the corresponding energy consumption recorded by Energy Profiler: from 32mw to 100mw. The combination of those two lines shows the tradeoff between energy use and transmission range under different transmission power. There is no optimal solution since different users should adjust the transmission power according to their own requirements.

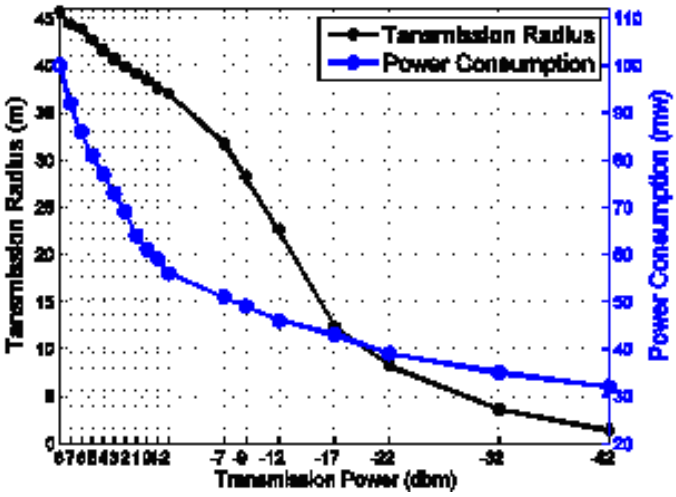

Fig. 15. Transmission radius and Power consumption *VS.* Transmission power.

B. CPU and RAM Benchmarks

We also carried out benchmark experiments to quantify the CPU and RAM usage of uSD and application running on the testing phone also using the Energy Profiler tool, as shown in Fig. 16 below on next page. The experiment also starts with measuring the amount of RAM and CPU usage when the phone is in idle state as defined in last subsection. In this idle state, the phone is working with around 18% of CPU and 65MB RAM. It can be calculated

communicating with WSN by testing application. The CPU and RAM usage are principally derived from virtual file read/write operations as discussed in Section III, as well as the packets analysis and some basic operations in application.

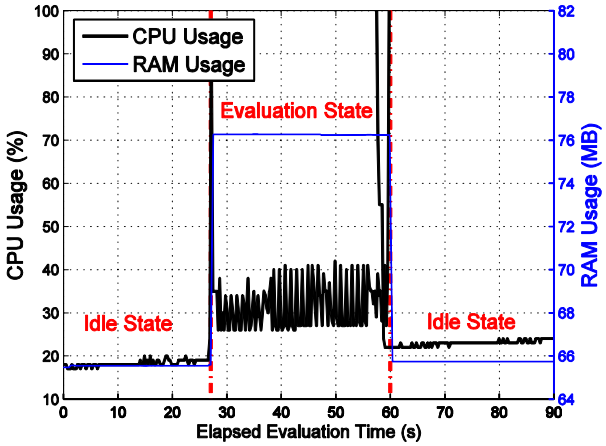

Fig. 16. CPU and RAM benchmark.

VII. CONCLUSION

This paper has presented the design, implementation and performance of our uSD card. uSD has two significances: 1) it gives a new structure of hardware expansion for embedded systems without any hardware or software upgrade, making the external equipment more compatible with Hosts; 2) it not only can help consumers with portable devices to access WSN conveniently and efficiently, but also are appropriate for researchers constructing testbed. In our future work, we will go on developing SDK of higher levels like application layer.

ACKNOWLEDGMENT

The authors would like to thank the whole uSD Research Group, including Master Xin MA and Yang YU, Dr. Shuai FAN from Tsinghua Universtiy; Libing XIONG from Beihang University; Dr. Jinfeng ZHANG from Nokia Research Center, this work benefitted greatly from their technological supports.

REFERENCES

[1] SD Card Association, "SDIO Simplified Specification," 2th version edition, 2007.
[2] Hafiz A Habib and Muid Mufti, "Real Time Mono Vision Gesture Based Virtual Keyboard System," IEEE Trans. on Consumer Electronics, vol.52, no.4, pp.1261-1266, Nov, 2006.
[3] Moteiv Corporation, "Tmote Mini," Datasheet, 2007.
[4] SPECTEC Corporation, "SDG-810 SDIO GPS Receiver," Datasheet, 2006.
[5] T. Pering, P. Zhang, R. Chaudhri, Y. Anokwa, and R. Want, "The PSI Board: Realizing a Phone-Centric Body Sensor Network," In IEEE Proc. 4th BSN, 2007, Germany.
[6] SD Card Association. "SD Specifications Physical Layer Simplified Specification," 2th version edition, 2006.
[7] Microsoft Corporation, "Microsoft Extensible Firmware Initiative FAT32 File System Specification," 1th version edition, 2000.
[8] Johan Thelin, "Foundations of Qt Development," Apress, 2007.
[9] USB Implementers Forum, "Universal Serial Bus Device Class Definition for Human Interface Devices (HID) Firmware Specification," 1st version edition, 2001.
[10] USB Implementers Forum, "Universal Serial Bus Mass Storage Class Specification Overview," 1st version edition, 2008.

BIOGRAPHIES

Chunxiao JIANG received his B.S. degree in information engineering from Beijing University of Aeronautics and Astronautics (Beihang University) in 2008 and was the candidate of Beihang Golden Medal Award. He is currently a PhD candidate in wireless communication and networking at Department of Electronic Engineering of Tsinghua University. He won the 1st prize scholarship of Tsinghua University in 2009 and became a candidate of Microsoft Research Asia Fellowship in Jun. 2010. His research interests include consumer electronics, Ad Hoc network, cognitive radio network.. He is currently a student member of IEEE communication society.

Nengqiang HE received the B.S. degree in communication engineering from Beijing Jiao Tong University, Beijing, China, in 2007. From 2007 Fall, he is the Ph.D. candidate supervised by Prof. Yong Ren in the Department of Electronic Engineering at Tsinghua University, Beijing, China. His research interests include wireless networks, multimedia transmission, congestion control, compressed sensing, network coding and erasure code. From 2008-2009, he was a visiting student as Research Assistant supervised by Prof. Jiannong Cao in the Department of Computing at Hong Kong Polytechnic University, Hung Hom, Hong Kong. He is currently a student member of IEEE and IEEE communication society.

Yong REN received his B.S, M.S and Ph.D degrees in electronic engineering from Harbin Institute of Technology, China, in 1984, 1987, and 1994 respectively. He worked as a post doctor at Department of Electronics Engineering, Tsinghua University, China from 1995 to 1997. Now he is a professor and the director of Complexity Engineering System Lab (CESL) in Tsinghua University. He holds 5 patents, and has authored or coauthored nearly 100 technical papers in behavior of computer network, P2P network and wireless communications. He has served as editor of Chinese of Journal Electronics and so on. His research interests include artificial life, complexity systems theory and performance evaluation of computer network.

Canfeng CHEN received his Bachelor of Engineering in Information Engineering and Ph.D. of Engineering in Signal and Information Processing from Beijing University of Posts and Telecommunications in 2000 and 2005, respectively. Since July 2005, he has been working for Nokia Research Center as a Post-doc Researcher, and joined Nokia Research Center as a Member of Research Staff in May 2007. He is a member of IEEE, and his main research interests cover wireless sensor networks, mobile Internet, and context aware services.

Jian MA received his Bachelor and Master of Telecommunication Engineering from Beijing University of Posts and Telecommunications, and Ph.D. of Communication Technology from Helsinki University of Technology in 1982, 1988, and 1994, respectively. Since August 1994, he has been working for Nokia Research Center, and now as a Principal Member of Research Staff and Guest Professor in several universities. He holds 11 patents and tens of patent applications. He is the co-author of 3 books, and has authored and coauthored more than 160 technical papers in the areas of IPv6, Mobile IPv6, IP traffic engineering, and QoS in IP networks. He has served as General Co-Chair of IEEE PCAC'07, Industrial Liaison Co-Chair of IEEE AINA'07, Workshop Co-Chair of IEEE ICC'08 and Panel Co-Chair of ChinaCom'08. His current research interests include grid services, converged services and solutions of wireless network and broadcasting network, new applications and services in digital broadcasting, Ad-hoc networking and P2P networking.