Plug-and-Play Sensor Node for Body Area Networks

Adnan Saeed, Miad Faezipour, Mehrdad Nourani and Lakshman Tamil
Quality of Life Technology Lab
The University of Texas at Dallas, Richardson, TX 75083
{axs055200, mxf042000, nourani, laxman}@utdallas.edu

***Abstract*— Advancements in low power wireless communication and system on chip design have opened up the possibility of developing miniaturized sensor nodes. These sensors can either be deployed on the human body (non-invasive) or implanted inside it (invasive) to collect vital physiological information and wirelessly transmit it to the system database. At the system backend this data is stored, processed, analyzed, and taken action if required. In this paper we propose the hardware architecture of a low-power, low-cost, small footprint, plug & play sensor node that is suitable for monitoring the vital signs. An event driven operating system is built for the sensor node to efficiently run the house keeping tasks and communicate with the gateway. As a proof of concept, an Electrocardiogram (ECG) monitoring application for body area networks using this sensor node has been developed.**

***Index Terms*— Body Area Network, Vital Sign Monitoring, Telemedicine, Ubiquitous Computing.**

I. Introduction

Body Area Network (BAN) consists of miniaturized sensor nodes attached to the human body (non-invasive) or implanted inside it (invasive) to collect vital physiological and non-physiological information and transmit it to the BAN controller; often called the Gateway. BAN is a special form of Personal Area Network (PAN) [1]; only being a little more personal than the PAN itself; with its roots in Imperial College where Prof. G. Z. Yang coined the term Body Sensor Network (BSN) in 2002 [2] in order to bring people together from different disciplines as medicine, electronics and computer science.

Figure 1 gives a generic example of such body area network where several non-invasive sensors are worn on the body to collect data and pass it on (via gateway) to system database where that data is stored, processed, analyzed, and taken action if required [3]. The sensors in this example can be classified in two categories; medical sensors [4] to monitor vital physiological signs (heart rate, oxygen level in the blood, blood pressure, rate of respiration and body temperature); and motion sensors [5] to collect information about the current state of human body (walking, running, standing, sitting, falling, etc.).

The hardware of sensor node usually consists of a microcontroller, few kilobytes of memory, ultra low power RF transceiver, antenna, sensors and actuators, analog signal conditioning circuitry, data converters, and battery module to power the node [6]. The sensor node needs to run an operating system which under the control of application logic is responsible for (i) moving data between data converter and memory, (ii) formatting and encrypting it for transmission, (iii) and reliably transferring it through the RF transceiver [8]. In addition, the OS is also responsible for task switching and managing system power.

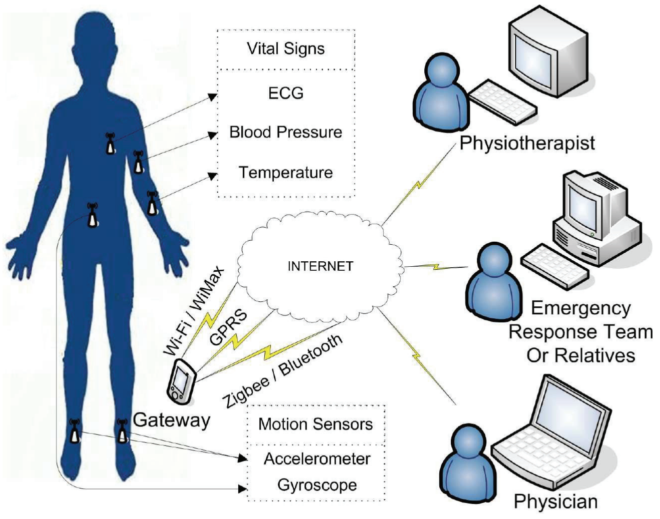

Figure 1. An Example of Body Area Network.

Sensors communicate with the BAN controller, or gateway, which is the main interface between the body area network and the internet. It is responsible for collecting data from sensor nodes; storing them in the local memory in case of no connection with the internet; and forwarding it on its outgoing port to internet for eventual storage on the system database. Due to the less stringent power requirements on the gateway (they can have large or rechargable batteries) some of them might have the ability to process different data streams and pass only relevant events to the system backend.

The system back-end consists of a database to store data and processing and analyzing software to deliver the services for which the system is intended for. Figure 1 shows a system where for example the physician, by looking at ECG signals of a user, may suggest a further detailed diagnosis in hospital if the need be. Or, a physiotherapist monitoring the recovery process of an accident patient who has fractured a limb, or an emergency response team providing immediate help in case of an elderly falling in her / his home.

II. Prior Work

A number of research [1], [7] and commercial [10], [11] sensor nodes have been reported for wireless sensor and body area networks.

Smart Dust [7] is probably the most ambitious project of all for it aims to build a sensor node of millimeter size.

Potential applications envisioned for this sensor are battlefield surveillance, treaty monitoring, or transportation monitoring. Aim of the project is to fabricate all necessary blocks of sensor node including MEMS, CMOS, and battery on a single chip. Optical line-of-sight communication is preferred in this design over RF because it consumes less power as no modulators, demodulators, or active band pass filters are required. A solar cell is also integrated to charge the 2 mm thick film battery present in the node.

iBadge [5] is a sensor node designed for Smart Kindergarten to be worn by children and provides the opportunity to teachers and parents for investigating students learning processes. MicaZ [10] is a commercially available sensor node designed for indoor building monitoring and deploying large scale (1000 plus) sensor nodes. It operates using two AA batteries and has an outdoor range of around 100 meters.

The most widely used operating system for sensor nodes is TinyOS [8] which is an event driven component based OS written in nesC programming language. nesC (Network Embedded Systems C) is a derivative of C designed to build TinyOS and applications for it. Programs are build as components which are wired to each other through interfaces.

III. Hardware Units & Implementation

Factors like wearability, flexibility, power consumption, and cost have influenced the design for a new sensor node. Wearability is the foremost important factor for monitoring different parameters over a long time. To the best of our knowledge, none of the commercially available sensors mentioned above [10], [11] were designed to be worn on body. Most of the other sensors reported in academia [1], [5], [6] also have large footprints. Placement of sensor nodes on different parts of the body imposes different size and orientation requirements. An ECG sensor worn on the chest can have a longer length but must not protrude too much out of the chest imposing height requirements on them. Whereas a sensor worn on the arm to measure blood pressure can have lenient height requirements but can not be too wide because of the curvature of the arm. The flexibility to re-arrange components on the sensor node in order to adjust the length, width or height as and when needed is the other important factor in the study of body area networks.

Figure 2 shows the block diagram and actual implementation of the proposed sensor node which we used as a base for building sensor nodes for body area network.

- **Digital Block:** The digital block consists of an 8051 compatible microcontroller running at 32 MHz, 8 KB SRAM and 128 KB flash memory, DMA controller, UART and SPI ports for serial communication, IRQ controller, reset circuitry, and watchdog and general purpose timers [12].
- **RF Block:** The RF block consists of a digital signal processor to perform automatic gain control, image suppression, channel filtering, demodulation, and frame synchronization for reception of data and implement data spreading and modulation for transmission of data in the ISM 2.4 GHz band using DSSS coding and O-QPSK modulation scheme [12]. Also present are the data converters, frequency synthesizers, mixers, variable gain amplifiers, and power and low noise amplifiers.

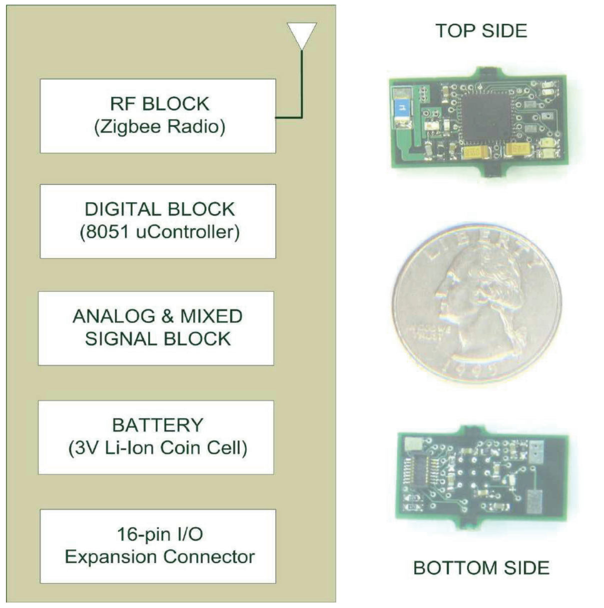

Figure 2. Block Diagram and Actual Implementation of Sensor Node

- **Analog and Mixed-Signal Block:** This block consists of a low dropout voltage regulator to regulate supply for rest of the system, brownout detector to continuously monitor the voltage supply and generates system wide reset when needed, full and low speed oscillators for full and low power operations, and data converters to interface with the real world.
- **IO Expansion:** 16-pin IO expansion connector is used to connect to application specific sensor boards. There are four ADC channels, UART and SPI interfaces multiplexed on the same lines for serial communications, and debug interface. Any unused pin can be used as a general purpose digital IO.
- **Battery:** Because of their high energy and reliability [13], a 3V lithium coin battery (1 cm radius and 165mAh rating) is used to power the node.

IV. Software Modules & Development

The sensor node runs a simple yet efficient operating system which is responsible for initializing and controlling multiple hardware blocks and moving data among them. The OS also provides synchronization between tasks and runs a MAC protocol stack for communication with the gateway. Operating systems can be categorized as multithreaded or event driven. In multithreaded systems different software tasks are implemented as threads and a *Scheduler* is responsible for multiplexing the execution time between different threads. Software tasks in event driven architectures are implemented as modules with a single entry point (event handler) called by the *Dispatcher* (only thread running in the system) based upon the occurrence of particular events of interest.

Figure 3 shows the Event Driven Message Passing architecture that we implemented as the operating system for our

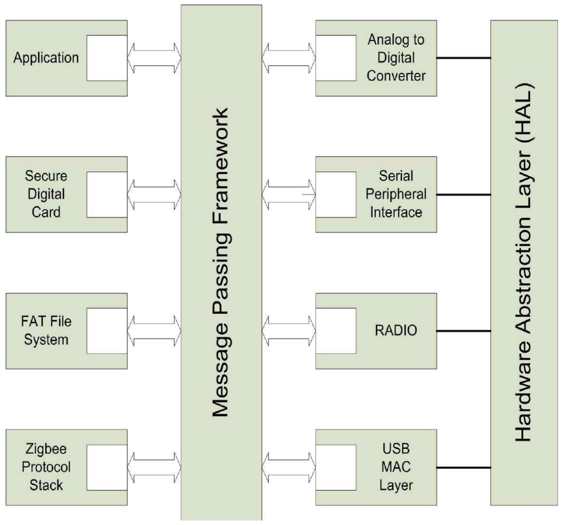
Application
Secure Digital Card
FAT File System
Zigbee Protocol Stack
Message Passing Framework
Analog to Digital Converter
Serial Peripheral Interface
RADIO
USB MAC Layer
Hardware Abstraction Layer (HAL)

Figure 3. Proposed Event Driven Message Passing Architecture

```
forever loop                          (Pseudo Code)
{
        if (Tail != Head)
        {
                get msg from Tail
                check msg destination
                call message handler
                free msg buffer
                increment Tail circularly
        }
}
_____________________________________________
MSG *pMsg;
                                      (C Implementation)
for (;;)
{
        if (qOut != qIn)
        {
                pMsg = MsgQueue[qOut++];
                MPF_Tasks[((pMsg->Dest_Src >> 4)
                & 0x0F)](pMsg);
                pMsg->Rsvd = 0;
                qOut &= (MPF_MAX_MSG – 1);
        }
}
```

Figure 4. Message Dispatcher

sensor node. Modules implement specific tasks and communicate with each other by sending messages through the Message Passing Framework (MPF). These modules can either abstract a particular hardware (Radio, ADC); or implement a particular software task (protocol stack, application logic). Factors like low power, concurrency, resource limitation and reusability have influenced the design of Message Passing Framework. MPF has a very small footprint (355 bytes of memory) and requires fewer CPU execution cycles (9 + 6 statements in C) to perform the main task of queuing and dispatching messages from and to different modules. No module specific interfaces are exposed by any module, thereby precluding the possibility of any module calling a blocking or time consuming function in any other module, making our framework truly concurrent and event driven. As the MPF is not responsible for abstracting hardware; it makes the MPF hardware independent and portable without any change. Also the device drivers are treated just as any other module in the system making it easier to add or remove them during compilation. Main components of MPF are described next.

- **Message Types:** There are four types of messages that are sent and received by the modules. They are the *Request-Response* message pair and *Indication-Confirmation* message pair. A module sends a *Request* to another module to do something and that module sends back the appropriate *Response* based on success or failure. Similarly, a module *Indicates* the occurrence of an event to another module and that module *Confirms* the indication to the originating module.
- **Message Structure:** The data structure of a message needs 10 bytes of storage space (for 8051 architecture) and is the only interface between a module and rest of the system. This data structure is designed in a way to facilitate the implementation of zero copy protocol stacks.
- **Message Queue:** A circular FIFO with rotating head and tail pointers is used as the data structure to queue and dispatch messages. When a module sends a message, a pointer to that message gets stored at the head of queue and head pointer gets incremented. The Dispatcher keeps track of the difference between head and tail pointers and upon finding one will dispatch the message from the tail of queue. Adequate size of the queue remains an open question as it may vary depending upon the application of the sensor node.
- **Message Dispatcher:** Figure 4 shows the pseudo code and C implementation for the dispatcher. When the head and tail pointers of the queue are not equal, a message from tail of the queue is picked up; its destination is checked; and the message handler for that module is called with that message as a parameter. When the module has processed the message, the message buffer is marked as free so it can be used later by other modules for sending messages. The tail pointer is incremented circularly and the entire process is repeated. It is the only thread running in the system and new messages enter the queue either during the execution of message handler or during interrupts; both of which are module specific implementations.

V. System Integration: An ECG Sensor Node

We have implemented an example application for body area networks using the sensor node that we have designed. Electrical waves passing through the body due to pumping action of the heart can be measured with electrodes attached to the skin. Figure 5 shows the block diagram and actual implementation of a 3-lead ECG sensor. ECG leads are connected to the instrumentation amplifier that amplifies the signal from electrodes (usually a few hundred microvolts to a few millivolts). It is conditioned before being passed on to the sensor node base via IO expansion connection, where the signal is digitized and transmitter wirelessly to the gateway. From the gateway the signal goes to the PC via USB port where it is conditioned digitally to remove the noise and

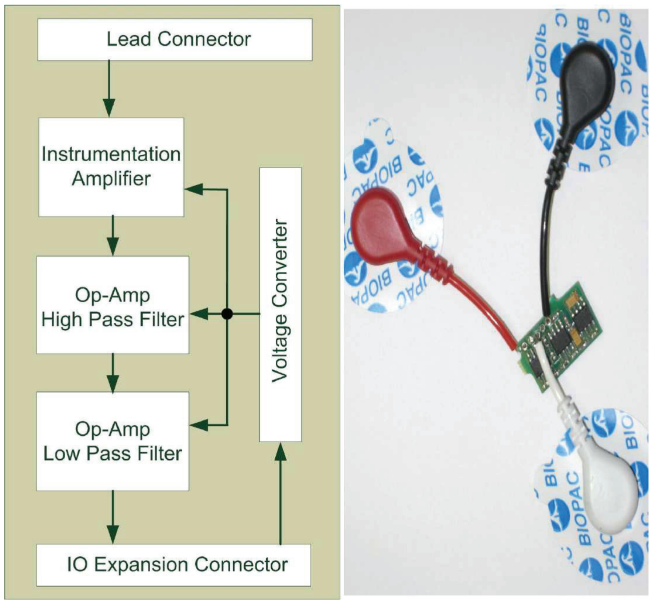

Figure 5. Block Diagram and Implementation of ECG Sensor Node

amplify it further. Lastly the QRS complex is computed from the ECG signal.

The ECG signal is conditioned before digitization with a tunable high pass filter to remove the baseline wander and an anti-aliasing low pass filter. The signal is sampled at 250 Hz and 25 samples are accumulated before transmission to the gateway at 10 packets per second. After the signal is received on the PC via gateway, it is passed through a series of digital filters to remove noise (including the 60 Hz hum) and further amplify the signal. The full analysis of ECG signal and diagnosis was not pursued in this work. The main intention here was to capture ECG signal using a low power and ultra portable sensor node that we designed and transmit that signal wirelessly to the system database where is could be processed and analyzed. Figure 6 shows the ecg waveforms before and after signal processing at the system backend. Matlab code for R-R Interval detection using Pan-Tompkins algorithm was also developed [9] which can be one of the many possible analysis that might be performed on an ECG signal.

VI. CONCLUSION

This work presents the design and implementation of a plug-and-play sensor node to be used in body area networks and Table I compares this node with some of the similar platforms mentioned elsewhere in this paper. Due to its advantage in size, cost and ease of use, it can be connected with different front-ends to realize different sensor types. An example ECG sensor node is designed as a proof of concept. An extension concerning the applications of this sensor node is an intelligent vest containing sensors for monitoring the five vital signs (ECG, blood pressure, blood oxygen, temperature, and breathing rate). Such intelligent vests have applications in hospitals and elderly homes.

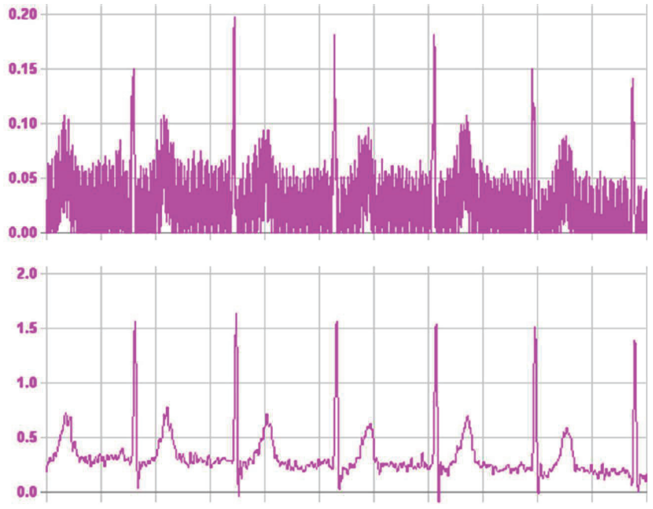

Figure 6. Signal Processing At System Backend

TABLE I
A COMPARISON OF SENSOR NODES

Name	Reference	Controller	Wireless Interface	Size (mm) L x W x H
Smart Dust	[7]	–	Infrared	5 x 5 x 5
XYZ	[6]	ARM (32-bit)	Zigbee	35 x 30 x 12
iBadge	[5]	AVR (8-bit)	Bluetooth	70 x 55 x 18
Pluto	[3]	MSP430 (16-bit)	Zigbee	40 x 28 x 6
MITes	[4]	8051 (8-bit)	Custom	30 x 25 x 8
BSN Node	[1]	MSP430 (16-bit)	Zigbee	46 x 31 x 7
MicaZ	[10]	AVR (8-bit)	Zigbee	58 x 32 x 7
iMote2	[11]	PXA271 (32-bit)	Zigbee	48 x 36 x 9
Proposed	–	8051 (8-bit)	Zigbee	25 x 12 x 6

REFERENCES

[1] Benny Lo, Guang-Zhong Yang, "Architecture for Body Sensor Networks," *IEEE Conference on Perspective in Pervasive Computing, pp. 2328*, Oct 2005.
[2] http://ubimon.doc.ic.ac.uk/gzy/m365.html
[3] M. Welsh, "CodeBlue: Wireless Sensor Networks for Medical Care," Harvard University, www.eecs.harvard.edu/ mdw/proj/codeblue/, 2005.
[4] E. M. Tapia, N. Marmasse, S. S. Intille, and K. Larson, "MITes: Wireless Portable Sensors for Studying Behavior," *In Adjunct Proceedings of the Ubicomp 2004*, 2004.
[5] S. Park, I. Locher, A. Savvides, M. B. Srivastava, A. Chen, R. Muntz, S. Yuen, "Design of a Wearable Sensor Badge for Smart Kindergarten," *IEEE Intl. Symposium on Wearable Computers, pp. 231238*, 2002.
[6] Dimitrios Lymberopoulos, Andreas Savvides, "XYZ: A Motion-Enabled, Power Aware Sensor Node Platform for Distributed Sensor Network Applications," *IEEE Intl. Symposium on Information Processing in Sensor Networks, pp. 449454*, Apr 2005.
[7] J. Kahn, R. H. Katz, K. Pister, "Emerging Challenges: Mobile Networking for Smart Dust," *Journal of Communications and Networks, vol. 2, pp. 188196*, Sep 2000.
[8] J. Hill, R. Szewczyk, A. Woo, P. Levis, K. Whitehouse, J. Polastre, D. Gay, S. Madden, M. Welsh, D. Culler, and E. Brewer, "TinyOS: An operating system for sensor networks," *Ambient Intelligence Book*, Dec 2005.
[9] Gari D. Clifford, Francisco Azuaje, Patrick McSharry, "Advanced Methods And Tools for ECG Data Analysis," *Artech House Publishers, First Edition*, Sep 2006.
[10] MicaZ Datasheet, www.crossbow.com
[11] iMote2 Datasheet, www.intel.com
[12] Texas Instruments Inc., Zigbee Solutions, www.ti.com
[13] Panasonic Inc., Lithium Coin Batteries, www.panasonic.com