BTnodes - Applications and Architecture Compared

Jan Beutel, Oliver Kasten, Matthias Ringwald *
Swiss Federal Institute of Technology (ETH) Zurich
8092 Zurich, Switzerland
beutel@tik.ee.ethz.ch

Abstract

We motivate a prototyping platform for ad-hoc networking research showing some requirements and constraints. The architecture of the BTnodes, each of which can store information, compute and communicate, is explained in conjunction with some demo applications that have been implemented. Important requirements and design trade-offs to be able to support multiple, compatible communication interfaces, to handle limited resources, for power-aware operation and for efficient testbed deployment are discussed.

1 Introduction

By using standardized communication interfaces, wireless sensor networking nodes can interact with these everyday appliances, peripheral devices, sensors and actors alike. According to [3, 6] and others, services in the network are the dominating factor for future growth. Fostering this interaction are well established and acquainted user interfaces on already common devices such as PDAs and cellular phones that make it possible to reach out into the digital representation of smart everyday objects and interactions.

Typical applications in research are in fast prototyping of demo applications [2, 7], interfacing to other devices (sensors, actors, multimedia and computing devices) [12] and the realization of emerging networking concepts that have so far only been theoretically specified and simulated.

Bluetooth is a connection-oriented, wireless communication medium that assures interoperability between different devices and enables application development through a standardized interface. This Host Controller Interface (HCI) hides most of the lower-layer abstraction from the system developer and leaves host-system resources to higher-layer applications. Networks of Bluetooth devices are organized in Piconets. These are Master-Slave star topologies that can be interconnected to form larger Scatternets. Compared to other media used in low-power wireless research [11, 9, 4], Bluetooth offers a host of high-level link-layer functionality such as multiplexing, integrated audio, different channel characteristics, link keys and encryption. The most apparent difference is that the developer is not dealing with a baseband and MAC interface but with dedicated communication channels. Applications thus need to be wireless-aware (broadcast medium), but no knowledge of digital signal processing and real-time systems is necessary.

Key requirements for a ubiquitous research platform are flexibility, power-aware operation, efficient deployment and the support of standardized interfaces. The following target features have been realized in the design of the BTnode:

- In-circuit programmable Bluetooth platform
- Remote update of system software
- Low component count
- Compact overall system size
- Simple debugging capability
- Sensor and user interface
- Single voltage design with power management

2 BTnode Architecture

The BTnode is an autonomous wireless communication and computing platform based on a Bluetooth radio module and a microcontroller. The benefit of this

*The work presented in this paper was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

platform is having a small form factor of 6x4 cm while still maintaining a standard wireless interface. With its general purpose interfaces the BTnode can be used with many peripherals, such as sensors, actors, DSPs, serial devices (like GPS receivers, RFID readers, etc.) and user interface components.

The BTnode hardware (see Fig. 1) consists of an Atmel ATmega128L microcontroller with on-chip memory and peripherals. The microcontroller features an 8-bit RISC core delivering up to 8 MIPS at a maximum of 8 MHz. The on-chip memory consists of 128 kbytes of in-system programmable Flash memory, 4 kbytes of SRAM and 4 kbytes of EEPROM. There are numerous peripherals integrated as well: JTAG for debugging, timers, counters, pulse-width modulation, 10-bit analog-digital converter, I2C bus, two hardware UARTs. An external low-power SRAM adds an additional 240 kbytes of data memory to the BTnode system.

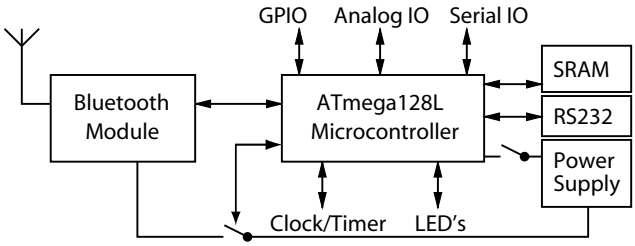

Figure 1. The BTnode system overview

A real-time clock is driven by an external quartz oscillator to support timing updates while the device is in low-power sleep mode. The system clock is generated from an external 7.3728 MHz crystal oscillator.

An Ericsson Bluetooth module is connected to one of the serial ports of the microcontroller using a detachable module carrier and to a planar inverted F antenna (PIFA) that is integrated into the board to further reduce the critical components. The operating and power modes of the Bluetooth module can be controlled by the MCU.

Four LEDs were directly integrated, mostly for the convenience of debugging and monitoring, although they could easily be optional external add-ons. One analog line is connected to the battery input and allows to monitor the battery status. The other peripherals are directly accessible via external connectors. These connectors all have the same setup with 4 signal and additional power and ground lines to be able to flexibly power and control add-on sensor peripherals. The supply voltage for MCU, memory and Bluetooth is fed through separate 0 Ohm resistors. When replaced by a current meter, this is a common way to enable exact in-situ power-consumption profiling for each component.

3 Communication Oriented OS Support

The BTnode system software is a lightweight operating system made up of low-level drivers that are interrupt driven and a simple dispatcher for scheduling multiple threads. This OS is well-suited for the applications of such small-scale networking devices that will consist mostly of simple IO and monitoring tasks and communication.

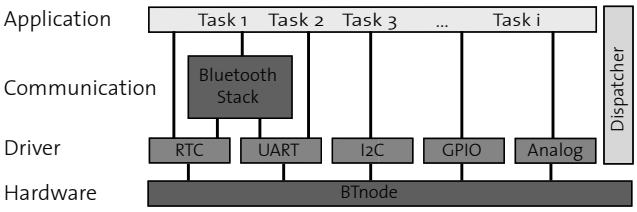

Figure 2. A lightweight communication oriented OS framework for WSN applications

An event-driven programming model provides convenient functions for resource management. The dispatcher is used for the scheduling of tasks; it implements coarse-grained cooperative multithreading. Only one task (event handler) can be active at a time. Events are processed in the order they appear. Every event handler is always completely executed until the next one is scheduled. So every event handler depends on the previous event handler to terminate in time. A software component, such as a driver, can generate an event to notify other components of the occurrence of a change in state that requires further action.

The second part of the system software are low-level interrupt-driven device drivers that allow applications to access the peripherals and interfaces in a standard way. The drivers are designed with fixed buffer lengths that can be adjusted at compile time to meet the stringent memory requirements. Available drivers include memory, real-time clock, UART, I2C, LEDs, power modes and AD converter.

4 Power Aware Operation

Different power-saving modes are available for both the microcontroller and the Bluetooth module. A real-time clock controlled by a separate driver is driven by a separate oscillator to allow to make use of the

low-power modes of the BTnode over longer periods of time. Furthermore, the microcontroller can be run at different operating frequencies controlled by the software.

A simple qualitative sensor application example (see Table 1) with a 10 % duty cycle reveals a quite acceptable average power-consumption of 6.5 mW and a battery lifetime T on the order of weeks on a standard 840 mAh Li-ion battery. Newer Bluetooth hardware is much less power-hungry than our first generation developer hardware, reducing power-consumption in communication mode by a factor of 2-4.

Operation	**t [sec]**	**P [mW]**	**T [h]**
Sensing	4	12	252
Communication	2	160	19
Idle	54	0.5	6048
Total		**6.5**	**421**

Table 1. Power consumption example

5 Platform Deployment and Tools

A software kit consisting of a build environment (avr-gcc cross compiler and standard libraries), source code, debugging support, demo examples and documentation has been assembled for the BTnodes and is available for download. A support mailing list and software repository are also available to developers.

In order to support fast prototyping and debugging without having to download to the embedded target on every design iteration, a separate build tree on Linux with the required interface and hardware emulation has been developed. This allows immediate execution of the identical system software as on the embedded target on a conventional PC with an attached serial Bluetooth device. A further advantage of this emulation mode is the possibility to read and write files on the host system and use the host system's advanced computing resources, e.g. for data gathering and analysis in a network of sensor nodes.

To simplify maintenance and application development with many BTnodes, a mechanism allowing selective network flooding with program-code updates has been developed.

The BTnodes have been developed and distributed in cooperation of the NCCR-MICS [5, 1] and the Smart-Its Project, the latter being a part of the EU Disappearing Computer initiative. The low complexity and small bill of material of the BTnodes results in a unit cost of USD 110 for the initial deployment of currently 200 units that have been distributed among different research groups worldwide.

6 Applications

Many applications have been realized using the BTnodes. Most of them are ubiquitous-sensing and monitoring applications where fast prototyping and ease of deployment are a key concerns [2, 10]. Other applications such as the one shown in Fig. 3 want to interface networks of sensors to commercial devices such as cell phones [12, 13] and PDAs [7]. For the latter type of application, compatibility and adaptability of the interfaces is the most critical issue.

Figure 3. Product monitoring using BTnodes as smart tags and SMS via mobile phones

Other applications are more specific in the networking requirements and deal with multihop schemes, ad-hoc routing, positioning and topology discovery, and hardware-adaptable systems [8].

References

[1] NCCR-MICS: Mobile Information and Communication Systems, Terminodes. http://www.terminodes.org.

[2] S. Antifakos, F. Michahelles, and B. Schiele. Proactive Instructions for Furniture Assembly. In *Proceedings of the The Fourth International Conference on Ubiquitous Computing (UbiComp 2002)*, Goeteborg, Sweden, September 2002.

[3] B. Raman et al. The SAHARA Model for Service Composition Across Multiple Providers. *Pervasive Computing*, August 2002.

[4] J. Hill, R. Szewczyk, A. Woo, S Hollar, D.E. Culler, and K.S.J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[5] P. Hubaux-J, T. Gross, Y. Le-Boudec-J, and M. Vetterli. Toward self-organized mobile ad hoc networks: The Terminodes Project. *IEEE Communications Magazine*, 39(1):118–124, Jan. 2001.

[6] D.G. Leeper. A Long-Term View of Short-Range Wireless. *IEEE Computer*, 34(6):39–44, June 2001.

[7] F. Michahelles and B. Schiele. Better rescue through sensors. In *Proceedings of First International Workshop on Ubiquitous Computing for Cognitive Aids*, Goeteborg, Sweden, September 2002.

[8] C. Plessl, R. Enzler, H. Walder, J. Beutel, M. Platzner, and L. Thiele. Reconfigurable hardware in wearable computing nodes. In *Proc. Int. Symp. on Wearable Computers (ISWC'02)*, pages 215–222. IEEE, October 2002.

[9] J. Rabaey, J. Ammer, J. da Silva Jr., D. Patel, and S. Roundy. PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking. *IEEE Computer*, 33(7):42–48, July 2000.

[10] K. Römer. The lighthouse location system for smart dust. In *Proceedings of the First ACM/USENIX Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*, May 2003.

[11] E. Shih, P. Bahl, and M. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *Proceedings of ACM MobiCom 2002, Atlanta*, September 2002.

[12] F. Siegemund. Spontaneous interaction in ubiquitous computing settings using mobile phones and short text messages. In *Workshop on Supporting Spontaneous Interaction in Ubiquitous Computing Settings, Ubicomp 2002*, September 2002.

[13] F. Siegemund and C. Flörkemeier. Interaction in Pervasive Computing Settings using Bluetooth-enabled Active Tags and Passive RFID Technology together with Mobile Phones. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, March 2003.