# A Network-Centric Approach to Embedded Software for Tiny Devices

David E. Culler, Jason Hill, Philip Buonadonna, Robert Szewczyk, and Alec Woo

University of California at Berkeley, Intel Research at Berkeley, Berkeley CA 94720, USA

**Abstract.** The ability to incorporate low-power, wireless communication into embedded devices gives rise to a new genre of embedded software that is distributed, dynamic, and adaptive. This paper describes the network-centric approach to designing software for highly constrained devices embodied in TinyOS. It develops a tiny Active Message communication model and shows how it is used to build non-blocking applications and higher level networking capabilities, such as multihop ad hoc routing. It shows how the TinyOS event-driven approach is used to tackle challenges in implementing the communication model with very limited storage and the radio channel modulated directly in software in an energy efficient manner. The open, component-based design allows many novel relationships between system and application.[1]

## 1 Introduction

The emergence of compact, low-power wireless communication, sensors, and actuators in the technology that supports the ongoing miniaturization of processing and storage is giving rise to entirely new kinds of embedded systems and a fundamentally new genre of embedded software. Historically, embedded systems have been highly engineered to a particular task. For example, a disk drive controller sits between a standardized command/response channel and the disk head assembly, with its rotation sensors, positioning actuators, and read/write heads. The controller software is a highly orchestrated command processing loop to parse the request, move the heads, transfer the data, perform signal processing on it, and respond. The system is sized and powered for the particular application. The software is developed incrementally over generations of products and loaded into a device for its lifetime. An engine ignition controller is even more specialized to performing a particular sense/actuate loop autonomously.

The new kinds of embedded systems are distributed, deployed in environments where they may not be designed into a particular control path, and often very dynamic. The fundamental change is communication; collections of devices can communicate to achieve higher level coordinated behavior. For example, wireless sensor nodes may be deposited in offices and corridors throughout

[1] This research was supported by the Defense Advanced Research Projects Agency, the National Science Foundation and Intel Corporation

T.A. Henzinger and C.M. Kirsch (Eds.): EMSOFT 2001, LNCS 2211, pp. 114–130, 2001.

a building, providing light, temperature, and activity measurements. Wireless nodes may be attached to circuits or appliances to sense current or to control usage. Together they form a dynamic, multihop routing network that connects each node to more powerful networks and processing resources. Through analysis of radio signal strength and other sensory nodes, nodes determine their location. They tap into local energy sources, perhaps using photovoltaic cells or nearby telephone or AC lines, to restore their energy reserves. Nodes come and go, move around, and are affected by changes in their environment. Collectively, they adapt to these changes, perform analysis of usage patterns and control lighting, temperature, and appliance operation to conform to overall energy usage goals.

The new genre of embedded software is characterized by being agile, self-organizing, critically resource constrained, and communication-centric on numerous small devices operating as a collective, rather than highly engineered to a particular stand-alone task on a device sized to suit. The application space is huge, spanning from ubiquitous computing environments where numerous devices on people and things interact in a context-aware manner, to dense in situ monitoring of life science experiments, to condition-based maintenance, to disaster management in a smart civil infrastructure. A common pattern we find is that the mode of operation is concurrency intensive for bursts of activity and otherwise very passive watching for a significant change or event. In the bursts, data and events are streaming in from sensors and the network, out to the network and to various actuators. A mix of real-time actions and longer-scale processing must be performed. In remaining majority of the time, the device must shutdown to a very low power state, yet monitor sensors and network for important changes while perhaps restoring energy reserves. Net accumulation of energy in the passive mode and efficiency in the active mode determine the overall performance capability of the nodes.

To explore the system design techniques underlying these kinds of applications and the emerging technology of microscopic computing, we have developed a series of small RF wireless sensor devices, a tiny operating system (TinyOS), and a networking infrastructure for low-power, highly constrained devices in dynamic, self-organized, interactive environments. The hardware platform grew out of the 'Macromote' developed in SmartDust project as a demonstration of the current analog of what might be put into a cubic millimeter by 2005 [8]. Our first experimental platform (Figure 1) had a 4MHz Atmel AVR 8535 Microcontroller, 8 KB of program store, 0.5 KB of SRAM, a single-channel low power radio [9], EEPROM secondary store, and a range of sensors on an expansion bus [4]. It operates at about 5 mA when active and 5$\mu$A in standby, so a pair of AA batteries provides over a year of lifetime at 1% active duty cycle. The severe resource constraints put this platform far beyond reach of conventional operating systems. TinyOS is a simple, component-based operating system, which primarily is a framework for managing concurrency in a storage and energy limited context. A collection of modular components build up from modulating the radio channel and accessing sensors via ADCs to an event-driven environmental