A Building Block Approach to Sensornet Systems

Prabal Dutta, Jay Taneja, Jaein Jeong, Xiaofan Jiang, and David Culler
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720
{prabal,taneja,jaein,fxjiang,culler}@cs.berkeley.edu

ABSTRACT

We present a building block approach to hardware platform design based on a decade of collective experience in this area, arriving at an architecture in which general-purpose *modules* that require expertise to design and incorporate commonly-used functionality are integrated with application-specific *carriers* that satisfy the unique sensing, power supply, and mechanical constraints of an application. Of course, modules are widespread, but our focus is far less on the performance of any individual module and far more on an overall architecture that supports the prototype, pilot, and production stages of design, and preserves the artifacts and learnings accumulated along the way.

We present heuristics for partitioning functionality between modules and carriers, and identify guidelines for their interconnection. Our approach advocates exporting a wide electrical interface, eliminating the system bus, and supporting many physical interconnect options for modules and carriers. We evaluate this approach by constructing a family of general-purpose modules and application-specific carriers that achieve a high degree of reuse despite very different application requirements. We show that this approach shortens platform development time-to-result for novice graduate students, making custom platforms broadly accessible.

Categories and Subject Descriptors

B.0 [**Hardware**]: General; B.4 [**Hardware**]: Input/Output & Data Communication

General Terms

Design, Experimentation, Performance

Keywords

Architecture, Mote, Wireless, Sensor Network

SenSys'08, November 5–7, 2008, Raleigh, North Carolina, USA.

1. INTRODUCTION

Sensornet platforms, like most other embedded systems, are tightly coupled to their applications. This coupling can make it difficult for general-purpose platforms to address application-specific needs, forcing platform designers to accept either suboptimal solutions or to repeatedly reimplement functionality. We propose a third way that composes platforms from a two-layer hierarchy consisting of compact, general-purpose *modules* which provide the common functionality and application-specific *carriers* which glue together these modules and also incorporate the sensors, power supplies, and mechanical constraints unique to the application.

Of course, sensornet platforms and modular approaches are widespread. In this paper, however, our focus is on an overall platform architecture for supporting the three phases of sensornet development – *prototype*, *pilot*, and *production*. This focus acknowledges the tensions among design tradeoffs in a rapidly changing field. The desire to tackle new, unexplored problems means that rapid prototyping and "try it and see" experimentation are very important. The wide diversity of valuable applications make realistic pilot studies at modest scale and modest investment essential, and these have to be well-enough executed to gain unprecedented measurements. And the maturing of the field means bringing the technology into production state, reducing cost, optimizing performance, improving manufacturability, and obtaining high reliability, all while preserving the learnings and artifacts accumulated along the way in moving rapidly through these phases of development. Despite the diversity of prior platform design efforts, it is safe to say that none of the available options meet all of these goals, as Section 2 articulates.

This paper presents a building block approach to sensornet platform design represented by the *Epic* family which we believe is the first to support all three phases of sensornet platform development well enough for rapid forward going innovation. The key ideas behind this approach include systematically partitioning functionality, exporting a wide electrical interface for modules, eliminating the system bus, and supporting multiple ways of physically interconnecting modules and carriers, from hand-soldering to machine-assembly. The specifics of this approach are presented in Section 3.

At the heart of the Epic family is a core module that integrates a state-of-the art microcontroller, IEEE 802.15.4 radio, and flash memory onto a small, inexpensive, single-sided board with excellent RF characteristics. Following the architectural principles of *exporting a wide electrical interface* and *minimizing logical interface constraints*, the core

exposes essentially all the pins that might possibly be useful, including internal signals, and does not hide any of these signals behind a multiplexed system bus. The core can be snapped into a standard socket for prototyping, easily soldered to routine carrier boards for pilots, or inlined for production, according to the principle of *supporting many physical interconnect options*. Despite this architectural focus, we recognize that modules can be only ϵ-suboptimal if they are to be enthusiastically adopted. Therefore, module designers must go to some lengths to ensure that the *basic building blocks exhibit competitive performance*. Section 4 describes the Epic core module and its internal subsystems, introducing the key characteristics and revisiting part selection with these in mind, looks at new alternatives since the core was designed, discusses manufacturing and mechanical considerations, provides a quantitative analysis of core module performance, and outlines future development directions for the core.

The case for a core module is clear: effective RF engineering requires *deep expertise* to design high-frequency circuits and *specialized equipment* to assemble, test, and tune them. These reasons are not limited to the core module, however. For example, a solar harvesting circuit can present a range of design options and subsystem choices that a designer unfamiliar in the art would find difficult to navigate. There are other reasons to build modules as well. Some functions are so common that *reuse in modular form* is inevitable. Many platforms, for example, require a USB host interface or battery charger, so this is an obvious module candidate. Finally, sometimes it is *simply more convenient* to group a set of related chips together on a board, like a handful of different memory technologies to create a memory hierarchy module. Collectively, these principles provide some guidance for partitioning functionality between modules and carriers. Complementing the core module is a supporting cast of specialized peripheral modules that offer a handful of choices for complete systems, and a framework for forward going innovation, as Section 5 describes.

The glue for these modules are breakout and development boards or application-specific carriers. For prototyping, breakout and development boards expose a wide array of pins and allow modules to be socketed, enabling novice system builders to compose platforms using simple jumper wires in a "try it and see" fashion and module developers to debug otherwise complex systems with complete freedom to access all exposed and intermediate signals. Section 6 explains our overall vision and approach for prototyping using the Epic family and presents some development hardware designed to support such prototyping efforts.

For pilots, inexpensive two-layer carriers are typically designed to fit a particular enclosure and a set of mechanical constraints with Epic modules being treated just like chips. For production, the modules are eliminated by incorporating their contents directly into the underlying board through *hardware inlining*. Section 7 evaluates the architecture by illustrating how these building blocks are used to build several simple, cost-effective, and application-specific carriers are designed using freely available CAD tools, inexpensively manufactured, and hand-assembled by novice graduate students. Carrier board design is so simple that it can be used even in an undergraduate classroom setting where students do application-specific design, fabricate the boards, and assemble a final solution in just weeks.

The final sections reflect on how effectively the Epic approach meets the various contraposing design goals of the three phases of sensornet platform development. Our experience shows that the building block approach leads to greater reuse, more compact designs, increased simplicity, and lower overall part count. Not only do modules become true building blocks, but so do other components created like CAD parts and scripts. An important benefit of viewing hardware in this way is that modules capture working hardware designs. In the future, we envision others will create many new modules make them available to the wider research community.

2. BACKGROUND

In the early stages of wireless sensor network research, the architecture and the form factor of the platform were wide open questions. The UCLA WINS project developed WinCE-based devices about the size of a shoebox [33]; USC developed PC/104 devices and proposed a tag that would have a small motherboard with slots for a radio board, a power board, and sensor boards [34]; the UCB SmartDust project developed the WeC mote with two microcontrollers, a radio, and a couple of sensors on a disk the size of a half-dollar [12]. Numerous other projects developed a variety of ARM-based systems. The Berkeley René mote [25] began a sea change by integrating the core elements of the low-power WeC design into a simple board with an array of common analog and digital interfaces organized like a conventional system bus on a 51-pin connector.

The René design reflected a key understanding that the common elements across sensor network applications are sampling, processing, storage, and communication, while the parts that are application-specific are the sensor suite, the power subsystem (which can support the application's sample and communication rate), and the mechanical design which holds the three together, exposes the sensors to the phenomena they need to sense, and protects the rest. This 51-pin "AT Bus" of the sensornet world carried forward to the MICA [25], MICA2 [3], MICAz [5], IRIS [2], and many, many other designs. Numerous sensor boards and power boards were designed to stack on it. In many ways, it shaped sensor network research activities for over five years.

Unfortunately, the 51-pin connector proved to be unsound for long-term deployments in harsh conditions, and it was expensive relative to the other components in the system. It began to fail the Goldilocks test – instead of being "just right" it was often too general for simple applications and too limited for demanding ones. New microcontrollers, new radios, and new flash chips led to a variety of new mote designs, such as the Mica2Dot [4], Telos [32], iMote [6], BTnode [15], Eyes, TIP [7], TinyNode [18], Sensinode [8], IRIS [2], MICAz Stamp [5], and kMote. Each with different form factor, connectors, power requirements, and interfaces.

In hindsight, this chaos was a symptom of an underlying tension among design tradeoffs. The rapidly changing nature of the field and the desire to explore novel applications meant that prototyping and experimentation were very important. Meanwhile, realistic pilot studies at modest scale were essential to gain unprecedented measurements, leading researchers to either use commercial offerings that were often not quite right or design their own platforms from scratch at great opportunity cost. And while the maturing of the field meant bringing the technology into production state, none

of the commercial offerings addressed the unique challenges in moving through the design phases, critical for preserving the accumulated learnings and artifacts.

Despite the diversity of efforts, earlier approaches remain inadequate because they rarely address the spectrum of needs for prototype, pilot, *and* production usage. We classify these approaches into three broad categories, *bus-based*, *highly-integrated*, and *assembly-optimized*, and explore their drawbacks.

The modular, stackable approach of bus-based architectures like WINS [33], MICA [25], iMote [6], PASTA [34], Stack [14], MASS [20], and mPlatform [29] make prototyping mechanically simple but their busses can present barriers to interfacing peripherals and also result in signal conflicts if not multiplexed, and their backplanes and board stacks can be too bulky, expensive, or fragile for realistic pilot or production use.

The highly-integrated approach, advocated by Telos [32], bundles a mote core with sensors, antenna, and host interface into a single circuit board, which makes software development and desktop experimentation easy. However, with this approach, realistic prototypes and pilots are strained because too few I/O lines are exported, production costs are too high since many unnecessary features are integrated, and onboard sensors are not useful for many scientific purposes.

To address the various shortcomings of the bus-based and highly-integrated approaches, vendors began to offer new, assembly-optimized module versions of their core platforms, like the MICAz [5], IRIS [2], and Tmote Mini [9]. These modules, while ideal for high-volume surface-mount assembly, are challenging to integrate into prototype and pilot projects because their packaging makes hand-assembly and socketing difficult or expensive, and their relatively narrow interfaces hide many internal signals useful for research.

3. BUILDING BLOCK APPROACH

This section presents the architecture and principles that support the prototype, pilot, and production stages of platform design, and preserves the artifacts and learnings accumulated in their implementation. At the heart of our approach are two architectural elements: the *module* and the *carrier*. Modules are reusable, self-contained subsystems in a multi-chip module (MCM) package. Modules are composed of one or more packaged ICs and other electronic components typically found on a system board. Carriers are custom circuit board substrates that glue together general-purpose modules with application-specific sensors, power supplies, and mechanical constraints. Heuristics for partitioning functionality between modules and carriers are discussed in Sections 4 and 5, and their effectiveness in Section 8. A sensornet platform constructed using this building block approach is shown in Figure 1.

Several principles focus on the interface between modules and carriers. First, we observe that a bus adds cost and complexity but that effective modularity does not really require one. Therefore, we *eliminate the system bus* from a module's interface specification. This allows modules to be flexibly wired together in whatever way a designer sees fit, rather than being encumbered by the constraints of a generic system bus since it uses precious circuit board space, requires costly or bulky or fragile connectors, complicates integration of peripherals, and reduces generality. Extending this line of thought, modules should *export a wide electrical interface* to maximize generality and reuse potential. Finally, to support prototype, pilot, and production purposes, modules should *support many physical interconnect options* ranging from socketing to hand-soldering to machine-assembly, as § 4.3 explores.

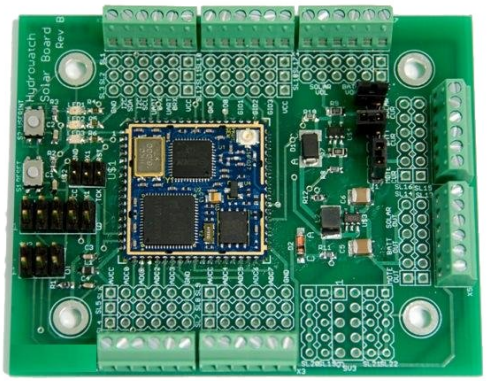


Figure 1: A sensornet platform designed according to the building block-approach. A general-purpose module (square board) is attached to an application-specific carrier (rectangular board). The carrier includes the sensor interface (large 2x3 and 2x5 headers), hosts a solar harvesting circuit (to the right of the square module), and conforms to a standard enclosure (footprint and four mounting holes).

4. CORE MODULE

Epic platforms are organized around a general-purpose core module as well as optional peripheral modules. This section describes the core module, shown in Figure 2, which is essentially the guts of a mote without the constraints on how it can be used. The core module integrates a state-of-the-art microcontroller, IEEE 802.15.4 radio, flash memory, a 48-bit unique serial identifier, and a U.FL RF connector, all attached to a four-layer, 1 mm thick, LCC-68 form factor circuit board one inch on a side, as Figure 3 shows.

Architecturally, the core is very similar to earlier mote designs like Telos [32] and MICAz, but its design is part of a larger framework that seeks to better support the prototyping, piloting, and production of sensornet platforms. To be useful for prototyping, the core module must be easy to use, debug, and profile, and it must provide good performance, sufficient storage, and ample I/O lines. To be useful for pilots, the core module must be easy to design-in at the CAD level, simple to hand solder at bench scales, and flexible when it comes to antenna choices. The core must also be easy to program in-circuit and debug *in situ*, both at the hardware and software levels. To be viable for production, the core must provide performance comparable to commercial modules, have an attractive cost profile, satisfy regulatory constraints like RoHS and FCC, and be open source to allow unforeseen innovation and adaptation.

4.1 Component Choices Revisited

When this study was started over a year ago, a handful of new microcontroller and radio options were available that did not exist when earlier platforms were designed, and to-

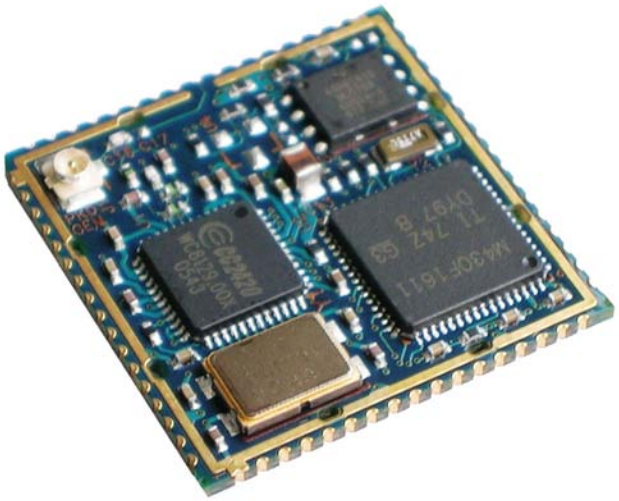


Figure 2: The Epic Core module: a wireless sensornet node ("mote") core that integrates a microcontroller, radio, and flash memory.

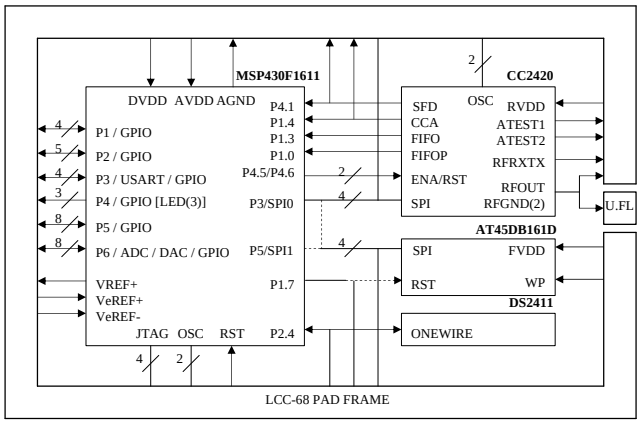


Figure 3: The Epic Core architecture. A Texas Instruments MSP430F1611 microcontroller and CC2420 radio sit at the heart of the core module. An Atmel AT45DB161D NOR flash provides 16 Mbit of storage. A Maxim DS2411 provides a globally unique serial identifier. Nearly all MCU peripherals are exported, including GPIO lines, ADC inputs, ADC voltage references, DAC outputs, USART lines, and the JTAG module. Many internal connections between components are exported as well.

day this list has grown even longer. This situation raises the question of whether earlier component choices still hold given today's offerings. The short answer is that when the core was designed a year ago, the answer was still yes. Today, the answer is still (mostly) yes. Moving forward, the answer is less clear. The rest of this section articulates the long answer to this question.

The opportunity to revisit the core's design raises another architectural question: *what changes are needed regardless of component choice to effectively support prototype, pilot, and production designs?* Addressing this question is a central contribution of this work.

4.1.1 Microcontroller

The microcontroller market includes many new offerings that were not available when earlier generation mote platforms were designed, as Table 1 summarizes. Many of the new offerings, like the TI MSP430F2618 and MSP430F5437 are product line extensions of existing microcontrollers like the MSP430F1611 that offer more memory, better performance, or new features. Other products, like the Jennic JN5139 or Atmel ATmega1281, were not available for consideration until recently. Given these new choices, it is worth revisiting why the MSP430F1611 still makes sense. Several factors influenced the decision to use this microcontroller, but most of the reasons are the same as the ones articulated in the Telos mote design [32]. These include low active current, wide operating voltage range, a 16-bit sleep timer, fast wakeup from sleep, a large amount of RAM, and three direct memory access (DMA) channels that can operate while the CPU sleeps.

By these metrics, the Atmel ATmega1281 (and larger ATmega2561) look more competitive than their predecessor, the ATmega128L. The active current has remained approximately constant at 0.9 μA at 1 MHz, only about twice that of the MSP430F1611. Since the microcontroller does not dominate the system power budget, this difference is not likely to have a large impact on lifetime. The operating voltage of the ATmega1281 matches the MSP430F1611 on the low end with a minimum voltage of 1.8 V and exceeds the MSP430F1611 on the high end at 5.5 V, providing a full 1.9 V wider operating range. This can be beneficial for systems that are directly connected to a lithium battery, which supplies between 2.6 V and 4.2 V, depending on its state of charge. This benefit only accrues if all system components can be operated over this range, which is usually not the case today.

The ATmega1281 offers 8 KB of RAM, only 2 KB less than the MSP430F1611. The memory requirements for many sensornet applications make the 4 KB available on the ATmega128L untenable. Embedded networked devices can use significant amounts of RAM to store message buffers while data collection applications can buffer sensor data in RAM for processing or prior to writing to flash. Therefore, RAM size is an important consideration for mote-class devices. With its 10 KB of RAM, the most among microcontrollers in its size and performance class, the MSP430F1611 remains a competitive choice. And yet, despite this significant amount of RAM, it still has among the lowest of sleep currents (with RAM retention). Today, we see fewer complaints about RAM since many systems with greater RAM requirements use members of the Telos family. We do observe that some applications, like TinyDB [30], require more flash memory than the MSP430F1611 offers, and since the ATmega1281 offers 128 KB and the ATmega2561 offers 256 KB, they are better choices for applications requiring a large code footprint.

Despite the ATmega1281's many improvements over the ATmega128L, there are two important drawbacks that tipped the scale in the MSP430F1611's favor. First, the ATmega1281 low-power mode timer is only 8 bits wide, meaning it has to wakeup every 7.8 ms (using a 32 kHz clock) to service

Mfg	Device	Year	Arch	GCC (y/n)	VCC (V)	RAM (kB)	Flash (kB)	Active (mA)	Sleep (μA)	Wake (μs)	Timer (bits)	DMA (y/n)	Area (mm^2)
Atmel	ATmega128L	2002	RISC/8	yes	2.7-5.5	4	128	0.95	5	6	8	no	81
	ATmega1281	2005	RISC/8	yes	1.8-5.5	8	128	0.9	1	6	8	no	81
	ATmega2561	2005	RISC/8	yes	1.8-5.5	8	256	0.9	1	6	8	no	81
Ember	EM250	2006	XAP2b/16	no	2.1-3.6	5	128	8.5	1.5	>1000	16	yes	49
Freescale	HC05	1988	8-bit	no	3.0-5.5	0.3	0	1	1	>2000	16	no	180
	HC08	1993	8-bit	no	4.5-5.5	1	32	1	20	4	16	yes	305
	HCS08	2003	8-bit	no	2.7-5.5	4	60	7.4	1	10	16	yes	144
	MC13213	2007	HCS08	no	2.0-3.4	4	60	6.5	35	10	16	yes	81
Jennic	JN5121	2005	RISC/32	yes	2.2-3.6	96	128	4.2	5	>2500	16	yes	64
	JN5139	2007	RISC/32	yes	2.2-3.6	192	128	3.0	3.3	>2500	32	yes	64
TI	MSP430F149	2000	RISC/16	yes	1.8-3.6	2	60	0.42	1.6	6	16	no	81
	MSP430F1611	2004	RISC/16	yes	1.8-3.6	10	48	0.5	2.6	6	16	yes	81
	MSP430F2618	2007	RISC/16	yes	1.8-3.6	8	116	0.5	1.1	1	16	yes	49
	MSP430F5437	2008	RISC/16	yes	1.8-3.6	16	256	0.28	1.7	5	16	yes	196
	CC2430	2007	8051	no	2.0-3.6	8	128	5.1	0.5	4	8/16	yes	49
ZiLOG	eZ80F91	2004	ez80/16	no	3.0-3.6	8	256	50	50	3200	16	yes	169

Table 1: Comparison of modern microcontrollers potentially suitable for sensornet platforms. The release year provides a sense of the underlying technology trends. The processor architecture and GCC support affect the cost and complexity of the toolchain. Key design considerations include RAM and flash memory size, active current (at 3 V and 1 MHz if possible) and sleep current, wakeup time from sleep, DMA support, largest width low-power sleep timer, mechanical package, and required circuit board area. For cases in which a manufacturer offers multiple products that are very similar, this table lists those parts with the largest RAM and flash. For cases in which a microcontroller comes in many packages, this table lists only the smaller (or smallest) package.

a timer overflow in sleep mode. Second, the ATmega1281 does not provide DMA support, important for collecting low-jitter samples [23] and high-throughput peripheral communications.

Today, there are many other low-power microcontroller and integrated microcontroller/radio choices available, so we briefly outline them and identify their strengths and weaknesses. The Freescale and ZiLOG microcontrollers are not supported by the GCC toolchain, making them less attractive for a research platform. In addition, the rather high active and sleep currents, long wakeup time, narrow operating voltage range, large footprint, and lack of GCC support make the ZiLOG eZ80F91 especially unattractive as a modern research platform.

Several of the microcontrollers also integrate a radio peripheral. The Ember EM250 integrates a 16-bit XAP2b microprocessor core and radio into a single chip package. An interesting feature of this product is its sleep timer which can operate from either a 32 kHz crystal or a calibrated 1 kHz clock, coupled with a prescaler (up to 2^{10} clock divider), which would let the node sleep for over 18 hours without a clock overflow. Unfortunately, a smaller RAM, higher active current, long wakeup, and uncertain GCC support make this device less appealing as a research platform.

The Jennic JN5121 and JN5139 also integrate a microprocessor and radio into a single package. Their large RAM and flash sizes are attractive but they come with a high cost: a wakeup time of 2.5 ms + 1 ms/kB of program memory when entering and exiting certain sleep states. The CC2430 provides a highly-integrated microcontroller with excellent across-the-board numbers, however its major drawback is a lack of native GCC support due to its 8051-based core.

The MSP430F2618 improves upon the already excellent MSP430F1611 performance numbers, is nearly pin-compatible, and addresses the major weakness of the F1611: limited flash memory. The recently announced MSP430F5437 adds still more flash and RAM but with a slightly lower active and higher sleep current than the F2618. Neither the F2618 nor the F5437 were available when the Epic core was designed but had they been, we would have chosen one, especially since their flash memories can be programmed down to 2.2 V.

4.1.2 Radio

Lacking relevant industry standards, early mote designs used a host of narrowband and wideband radios for their wireless interface. For example, designs employed radios that modulate the signal using on-off keying (OOK), amplitude shift keying (ASK), frequency-shift keying (FSK), and phase-shift keying (PSK). More recently, with widespread consensus on the IEEE 802.15.4 standard, at least at the physical layer, and a number of vendors now offering compliant radios, this choice is a natural one. The diversity in 802.15.4 radio choices, shown in Table 2, once again opens up the design space and warrants a reexamination of the available options with the benefit of hindsight. Although our specific design point focuses on standards-based radios, we do not believe the architectural choices would be different if a another standard (or none at all) were chosen.

For many systems, radio idle listening dominates the system power budget, so receive power is an obviously important metric. By this standard, the Atmel RF230 would be the best radio option since it offers the lowest receive current and best receive sensitivity. However, for CSMA systems employing low-power listening [31], the key to reducing the idle listening cost is to minimize the cost of channel polling since this time establishes the lower bound on duty cycle. The channel polling time is the sum of several factors: the startup time of the radio's crystal oscillator, the time to sample the channel for energy, and the time to convey this information to the microcontroller. Using a low-resistance crystal, the CC2420 is reported to start in 580 μs and detect channel energy in 128 μs [32]. Since the CC2420 exports the clear channel assessment (CCA) signal using a dedicated pin, this allows the host microcontroller to determine if there is channel activity without having to poll the radio over the SPI bus, reducing channel polling time. Since the CC2420 can wake up in about half the time of the RF230 and convey the channel status to the host using hardware lines, the energy cost of polling the channel should be substantially lower on the CC2420 than the RF230.

Another temptation with the RF230 comes from its ultra-low sleep current but this logic is deceiving on two counts. The reasoning would go, since the node is asleep most of the

Mfg	Device	Year	Wake (ms)	VCC (V)	RxSens (dBm)	TxPwr (dBm)	Rx (mA)	Tx (mA)	Sleep (μA)	FIFO (Rx/Tx)	SCLK (MHz)	SFD (y/n)	CCA (y/n)	AES (y/n)	Area (mm^2)
Atmel	RF230	2006	1.1	1.8-3.6	-101	+3	15.5	16.5	.02	128	8.0	no	no	no	25
Ember	EM260	2006	1	2.1-3.6	-99	+2.5	28	28	1.0	128	5	yes	yes	yes	36
Freescale	MC13192	2004	7-20	2.0-3.4	-92	+4	37	30	1.0	128/256	8.0	yes	yes	yes	25
	MC13202	2007	7-20	2.0-3.4	-92	+4	37	30	1.0	128/256	8.0	yes	yes	yes	25
	MC13212	2005	7-20	2.0-3.4	-92	+3	37	30	1.0	128/256	8.0	yes	yes	yes	81
Jennic	JN5121	2005	>2.5	2.2-3.6	-93	+1	38	28	<5.0	16	16.0	yes	yes	yes	64
	JN5139	2007	>2.5	2.2-3.6	-95.5	+0.5	37	37	2.8	16	16.0	yes	yes	yes	64
TI	CC2420	2003	0.58	2.1-3.6	-95	0	18.8	17.4	1	128/128	10	yes	yes	yes	49
	CC2430	2005	0.65	2.0-3.6	-92	0	17.2	17.4	0.5	128/128	4	yes	yes	yes	49
	CC2520	2008	0.50	1.8-3.8	-98	+5	18.5	25.8	.03	128/128	8.0	yes	yes	yes	25

Table 2: Comparison of modern IEEE 802.15.4-compatible radios. The release year provides a sense of the underlying technology trends. The wakeup time (wake) is the time required to transition the radio from sleep to listen. The receive sensitivity (RxSens) is a measure of the minimum signal strength needed for successful reception. The transmit power (TxPwr) is the output power of the radio. Rx, Tx, and Sleep are the receive, transmit, and sleep current draws. The amount of the receive and transmit data path buffering is available (FIFO). The speed of the data bus (SCLK) limits the rate of data input/output to/from the radio from the host microcontroller. The start-of-frame-delimiter (SFD) is a hardware handshake signal that toggles at a well-defined point during packet transmission or reception. The clear-channel-assessment (CCA) is a hardware handshake signal that indicates whether the channel power exceeds the clear channel threshold. The advanced encryption system (AES) indicates whether hardware support for encryption is included in the radio.

time, sleep current matters a great deal. While this may be true in theory, in practice the constant factors dominate. First, the sleep cost must consider sleep currents aggregated over all components, and the lowest microcontroller current is 25 times larger at 500 nA. Second, for systems that operate around 1% duty cycle, but use a radio whose active current to sleep current is 10000:1 or higher like the RF230, energy consumed in the sleep state pales in comparison to energy consumed in the active state. Recent research has demonstrated radio operation at permille (0.1%) duty cycles, making sleep currents more important yet still not among the most important of factors.

The RF230 also offers better receive sensitivity than the CC2420 (-101 dBm vs -95 dBm) and higher transmit power (+3 dBm vs 0 dBm), so its link budget is about 9 dB higher than the CC2420. This translates to either longer-range or lower-power communications since transmit power is adjustable. Finally, a shared send/receive FIFO and the lack of hardware support for AES means this cryptographic function must occur in MCU software, rather than in optimized hardware.

Today, there are many other 802.15.4-compliant radio choices, so we briefly outline some of them and identify some of their strengths and weaknesses. The EM260 appears to offer excellent receive sensitivity and transmit power, at the expense of higher current draws and a constrained development environment. The Freescale family of radios offer an order of magnitude longer wakeup times, in the range of 7-20 ms, than the CC2420 as well as much higher current draws. The Jennic JN5121 and JN5139 are attractive because of their large RAM and 32-bit core, but their 2.5 ms minimum wakeup latency is long, and still longer if RAM retention is disabled and the program must be copied to RAM from flash on each wakeup. The CC2430 appears to be an excellent, highly-integrated system with ample RAM and flash. The only downsides are low receive sensitivity and a lack of GCC toolchain support. Finally, the CC2520 offers nearly all of the benefits of the CC2420 and RF230. If this radio had been available when the core was designed in early 2007, we would have selected it.

For these reasons, the CC2420 still provided the best overall power profile at the time of the Epic core design, cementing our decision to use it in the core module. To ensure a low radio wakeup similar to Telos, the core's radio oscillator circuit is built around a Hong Kong Xtal's C5M family 16 MHz crystal. This decision was inspired by observations that showed the benefits of choosing a crystal with a low series resistance, namely allowing the radio to start up quickly [32]. This crystal's lines are also exported using short traces to allow oscillator quick start circuits to be explored using this module [16]. If such a circuit is added, care must be taken to ensure that capacitive loading of the crystal does not exceed the manufacturer's recommended tolerances. Our evaluation of the Epic core in Section 4.4 shows that its wakeup performance tracks that of Telos.

The CC2420 also provides a pair of test lines, ATEST1 and ATEST2. These lines can be programmed to output a range of internal signals at various stages of the signal processing pipeline. Although normally intended for production testing, these signals can provide the low-level access needed to implement analog network coding [27] or interference cancellation [24]. The radio SPI bus, CCA, and SFD lines are also exported from the module, simplifying external probing and allowing external hardware to count both the number of times these signal are asserted as well as the amount of time they remain asserted. These are important indicators of channel activity, availability, and interference.

4.1.3 Flash

The core uses an AT45DB161D NOR flash [1] that provides 16 Mbit of non-volatile storage. Although this chip has a higher sleep current than the ST M25P80 [11] used in Telos, the dual RAM buffers simplify driver software and allow data to be accessed from one buffer over the SPI interface while the other buffer is busy reading from or writing to non-volatile storage.

There are two core module designs that only differ in the way the flash memory is connected to the MCU. In one configuration, the radio and flash are on the same bus (SPI0), preferable for workloads where the node is connected with another serial device, like a host computer or a sensor with an RS-232 port. In the other configuration, the flash and radio are on different buses, SPI0 and SPI1, respectively, desirable for nodes that do not use their UART, like routers

in a mesh network, since resource contention will not occur and SPI bandwidth does not have to be shared.

The flash memory has a write-protect line that is exported because there is no broadly appropriate default. According to one school of thought, a "boot sector" should always be write-protected unless the module is being reprogrammed through physical connection to a programming board or host computer; however, there is no simple and fool-proof way for the module to determine this unambiguously. According to another school of thought, the default behavior of the module should be to allow the flash to be reprogrammed in its entirety. The issue boils down to a policy decision, so in the interest of end-user flexibility, this line is neither driven nor pulled high or low – the platform developer has the option to pull-up this line by populating a resistor on the core module.

4.2 Implementation Decisions

This section presents several implementation choices that focus on component interconnections, I/O exports, and supply circuitry that have architectural motivations like "export everything" and "minimize constraints."

4.2.1 Component Interconnections and Exports

The MCU communicates with the radio using an SPI bus (USART0), receives status information (CCA, FIFO, and FIFOP) from the radio using three interrupt-capable input lines and packet transmission/reception timing (SFD) from the radio using one timer capture register, and controls and resets the radio using a pair of output lines. The MCU communicates with the flash memory using SPI on either USART0 or USART1 and communicates with the serial identifier chip using a single, interrupt-capable, GPIO line with pull-up to implement Dallas Semiconductor's 1-wire protocol. The MCU exports a byte-wide port to simplify the interface to devices with a byte-wide bus interface like NAND flash memory, FIFOs, and high-speed parallel ADCs.

In addition to the communications and control interfaces shared with the MCU, the radio also exports a wireless interface and some useful test lines. The wireless port passes through a balun and is routed to both a 50-ohm RF port on the LCC-68 module as well as a U.FL connector onboard the module circuit board. A single capacitor selects which way the RF signal goes – LCC-68 pad or U.FL connector. This flexibility allows developers to choose either an external antenna with a U.FL-terminated pigtail – now common because of 802.11 b/g radios – or a board-integrated antenna like a chip antenna or a planar-inverted F-antenna (PIFA). The first choice eliminates low-level RF engineering while the second choice allows for a more compact solution.

4.2.2 Power, Ground, and References

The core exports four different power supply lines for the four major power domains: DVDD supplies the microcontroller core and serial identifier, AVDD supplies the ADC core and reference, RVDD supplies the radio, and FVDD supplies the external flash memory. These signals may be tied together externally, connected to different supplies with slightly different voltages, or individually passed through current sense resistors to allow current profiling per power domain. All of the supply lines are internally decoupled using 0.1 μF capacitors. If long external power traces are used, larger external capacitors should be used. The core also exports several references used by the ADC. The VREF+ line allows the internal ADC reference to be used by external circuitry (with appropriate buffering). The VeREF+ and VeREF- lines allow externally-generated high and low references to be used by the ADC.

In addition to the four supply lines, the core exports four different ground lines. Although three of these ground lines are internally connected, they individually provide the preferential ground return for the microcontroller, radio, and flash memory. The fourth ground line, AGND, connects to an isolated ground plane section and provides the return for the analog section of the microcontroller. The AGND can be connected to the digital grounds externally, but care must be taken to reduce digital noise from coupling with AGND. Finally, the radio ground is divided into a digital section and an analog section with a separate ground, RFGND. The radio digital section shares a common ground with the microcontroller and flash while RFGND provides the return for the RF path. The point where the RFGND lines are exported from the module is the only place where the analog and digital grounds are connected together – the proverbial "ground mecca" – situated on the ground ring along the module perimeter, providing a convenient solder point for an RF shield.

4.3 Mechanical Design

A question that every module designer must confront at some point is *what form factor and connector interface should the module use?* There are nearly as many different answers to this question as there are mote platforms. The Epic core module uses an industry-standard LCC-68 (68-pin leadless chip carrier) form factor that places all parts on one side of the module circuit board and exposes nearly every signal that might possibly be useful along the board edge via perimeter pads. This packaging wastes no connector space since the board edge is otherwise unused, allows a seamless transition from prototype to production since modules can socketed, hand-soldered, or machine-assembled, and a single-sided board makes signal probing easy.

Several considerations played a role in the choice of perimeter pads. First, since the package is leadless, no costs are incurred on connectors. Second, since the package land pattern is essentially JEDEC-compliant (except for pin numbering), an off-the-shelf prototype or production socket can be used to program the device or break out the signal lines for debugging. Third, since the 68 pads around the module perimeter are actually plated-through semi-holes (also known as castellations or routed vias), they are easy to solder by hand which greatly simplifies prototyping. Fourth, since the plated-through semi-holes are concave, an oscilloscope or voltmeter probe tip rests easily in them, making debugging just a bit easier. Fifth, since the plated-through semi-holes are actually vias that connect all layers of the circuit board, they reduce the number of vias that might otherwise be necessary, potentially reducing cost and providing more circuit board real estate.

Superficially, the Epic core's LCC-68 footprint might seem similar to the the MICAz [5] and IRIS [2] OEM modules or the Tmote Mini [9], but there are some important differences that make Epic well-suited to pilot studies: it can be hand-soldered, it has a wide interface that exports nearly every internal signal, and it can be socketed. This design consideration raises an important architectural question: *should*

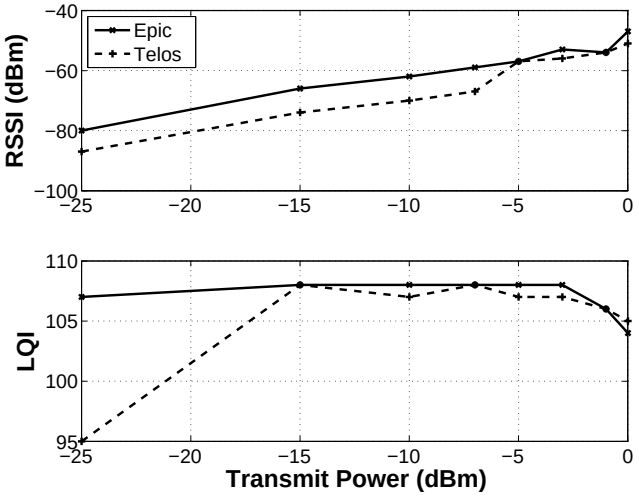


Figure 4: Radio reception performance (RSSI and LQI) of Epic and Tmote Sky over the same channel as the transmit power is swept from -25 to 0 dBm.

the number of pins a module exports grow linearly with its area or as the square root? A ball grid array (BGA) allows a linear relationship between area and pin count while the perimeter pins of a leadless chip carrier (LCC) grows as the square root of the area. We chose an LCC-68 package with plated-through semi-holes to allow hand assembly, but a side-effect of the decision is that modules are more limited in their I/O width. We also experimented with different module thicknesses and found that an 0.5 mm board was too flimsy (without a structural shield) and that the standard 1/16 in circuit board was unnecessarily thick, so we settled on a module thickness of 1.0 mm.

Other mote designs, like the MICA family including the MICA, MICA2, and MICAz often waste circuit board real-estate unnecessarily making them too large to comfortably design into enclosures, require expensive and fragile connectors, and do not export many I/O lines useful for research and experimentation. The highly-integrated Telos suffers from many of these same problems. The MICA2Dot [4] is more space-optimized and integrates the core pieces better, buts its limited I/O lines reduce choice, its connector is difficult to attach, and its antenna connector is poorly matched.

4.4 Evaluation

Modules will only be adopted if their performance is at most ϵ-suboptimal to other alternatives, and we show here that Epic compares favorably to earlier work. One of the key metrics for a platform is the radio wakeup time. We measured the wakeup time of both Epic and Telos by monitoring the state of the CC2420's CCA pin in the same way that the TinyOS 1.x and 2.x stacks use to determine when the oscillator has stabilized. In our experiments, Epic wakes up in $629 \pm 3\mu$s while the Telos wakes up in $619 \pm 3\mu$s (95% confidence).

Sleep current is another important performance metric which for Epic is 7 μA at 3 V. In comparison, we measured the Telos sleep current to be 6 μA at 3 V when running the TinyOS `Null` application. Although the Epic sleep current is comparable to Telos, the constituent currents are different: most of the Epic current draw comes from the flash chip while most of the Telos current draw comes from its host interface, which Epic removes for reasons of generality.

To evaluate radio reception, a transmitter node (Telos B) is positioned 3 m from a fixed antenna. In the first experiment, a Sentilla Tmote Sky [10] is connected to the antenna. In the second experiment, an Epic is connected to the same antenna. During each experiment, 20 packets are transmitted from the sender to the receiver. The received signal strength indicator (RSSI) and link quality indicator (LQI) are logged. This experiment is repeated at eight different power levels. These results, along with tests over a range of channels and distances, confirm that the RF performance of Epic is commensurate with a mature commercial system.

As a cautionary note, we point out that achieving this performance required months of design, evaluation, tuning, and redesign. This work was carried out using expensive test and measurement equipment including high-speed digital oscilloscopes, spectrum analyzers, and network analyzers. In the final analysis, ten different RF section layouts, three different inductor choices, and two different RF ports were evaluated. All of our designs are open-sourced and available online.

4.5 Future Directions

In hindsight, the choice of the MSP430F1611 microcontroller and CC2420 radio have stood the test of time, and product line extensions like the MSP430F26x, MSP430F54x, and CC2520 promise a simple migration pathway forward. An obvious next-generation core module will be an evolutionary one that integrates these much improved but still backward-compatible parts. This path will allow the community to leverage existing investments in software yet allow new research efforts by moving more functionality into the radio, and making the processor-radio interface richer and more flexible. At the same time, new products from other vendors are quickly closing, or have already closed, the gap in wakeup latency, RAM size, low-power timer support, direct memory access, and operating voltage range.

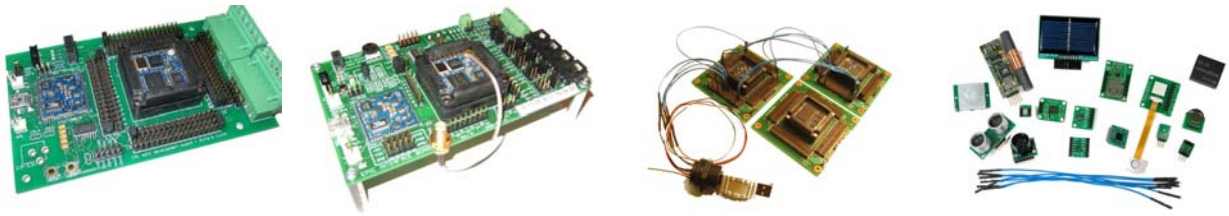
5. EXPERT PERIPHERAL MODULES

Complementing the core module are a family of peripheral modules that provide specific functions, such as power supply conditioning, high speed host communication interfaces, bulk storage, or analog signal conditioning. Figure 6 shows the modules currently in the Epic family.

(a) Core (b) Storage (c) USB

Figure 6: Core, storage, and USB Epic modules.

Since a key aspect of the architectural approach is a systematic partioning of functionality between modules and carriers, we identify four cases when modules make sense: when their design requires *deep expertise*, when their assembly or tuning requires *specialized equipment*, when their function is so common that *reuse in modular form* is inevitable, and when it is *simply more convenient* to group a



(a) Development Board (b) Interface Board (c) Breakout Board (d) COTS Parts

Figure 5: The Epic family includes hardware specifically designed for (a) making platform prototyping possible in a classroom setting by novice designers (b) interfacing with the popular Phidgets analog and digital sensors (c) empowering module designers to construct, probe, and debug intricate circuits on-the-fly, both only using (d) off-the-shelf parts such as jumpers, sensors, solar power packs, and surfboards.

set of related components. Collectively, these principles provide some guidance for partitioning functionality between modules and carriers and they address the question: *where do modules come from?* Following these heuristics, Table 3 traces the genesis of the modules currently in the Epic family, and the remainder of this section discusses their functions.

Module	Deep Expertise	Special Equip.	Modular Reuse	Simple Convenience
Core	yes	yes	yes	no
USB	no	no	yes	yes
Storage	no	yes	no	yes

Table 3: The genesis of core and peripheral modules in the current Epic family. Modularizing a component is beneficial if it demands deep expertise to design, requires specialized equipment to assemble or tune, is general enough that reuse in modular form is inevitable, or just as a way to group together related parts into a subsystem as a matter of simple convenience.

The USB module provides four functions: host interface, reprogramming, JTAG over USB (requires additional host software), and battery charging and management. The first three offer the same functionality as the Telos [32] in that the host interface and reprogramming functions are multiplexed using the same I/O lines and JTAG over USB is possible (but not supported). The battery charging and management can recharge a Lithium battery whenever the module is plugged into a USB port and arbitrate between USB power and an attached Lithium (or alkaline) battery. This module was built because it was perceived to be quite useful to a number of platforms in modular form and was a convenient container for related functionality.

The storage module integrates four different non-volatile memory chips – a 1 Gbit NAND flash, two 16 Mbit NOR flashes, and one 512 Kbit FRAM. These memory chips have very different read, write, and erase characteristics and so they represent a useful collection of chips integrated on a single module for simple convenience when researching storage systems. Additionally, some of the included flash chips are in packages that are either leadless or with extremely small pitch, making them difficult to hand solder and warranting specialized manufacturing equipment. This module was built both for experimentation and as a storage subsystem for motes.

6. PROTOTYPING

In our vision for prototyping, platform developers are able to pick a handful of components like sensors, motes, battery packs, and solar harvesting modules, and literally wire them together in whatever way is most appropriate. Writing the corresponding system software would follow a similar pattern; most components would have associated drivers that could simply be declared and wired to the hardware resources they use, like GPIO lines, ADC channels, or an SPI bus. We envision the emergence of platform construction kits that include an assortment of building blocks, their associated driver software, and the glue to assemble a wide variety of prototype nodes. In this section, we examine how Epic supports prototyping approaches for both novice and advanced system designers.

6.1 Try It And See

Many projects begins with experimentation and rapid prototyping inspired by a "try it and see" attitude. The goal is to demonstrate a basic implementation that showcases an important capability, enables some exploratory data to be collected, or reduces perceived implementation risk through an existence proof. At this stage of the game, maximum impact demands a narrow focus on the essential elements of the system, but the other parts must be good enough to evaluate the prototype. The metric of merit is time-to-result.

Unfortunately, several factors increase time-to-result. Issues like sensor and power supply selection, electrical wiring, and device driver development dominate engineering efforts while more novel aspects like application software, performance characterization, and end-user data collection are routinely back burnered during the initial stages. To improve productivity, we created a Development Board that can be easily and inexpensively integrated with off-the-shelf sensors, displays, and solar packs to improve time-to-result.

Figure 5(a) shows the Development Board, which benefits from the choice of an industry-standard LCC-68 footprint by including an off-the-shelf socket for easily swapping modules. Adhering to the principle that all signals should be available to the platform designer, breakout pins allow access to every signal, simple shorting shunts allow each signal to be individually connected to power or ground, and jumper wires allow a signal to be easily connected to off-the-shelf parts like the ones shown in Figure 5(d).

The Development Board also incorporates a USB module for programming, alkaline and lithium battery connections for supplying power, and LEDs and buttons for feedback,

debugging, and control. This flexible platform enables quick prototyping and exploration of novel development elements while circumventing the complexities of module and carrier design. The board has already been used by undergraduate students to develop application-specific platforms and a second version, shown in Figure 5(b), was used to teach a summer school on wireless embedded systems.

6.2 Debugging

Debugging is an often frustrating aspect of prototyping. Effective debugging requires the developer to probe signal voltages to verify circuit operation and measure currents to identify unexpected draws and verify expected ones. Unfortunately, many systems can make probing signals and debugging painfully difficult: signals are buried under chips, routed through to intermediate layers of the printed circuit board, and never exposed through any header. Measuring currents can be still more challenging since it requires breaking a circuit to take the measurement. In most systems, directly measuring the individual draws of the microcontroller, radio, flash, or other peripherals is impossible since the individual power supply lines are buried in the circuit board and a single, global power supply line is exposed. The result is that developers must write test code that isolates different functions, rather than being able to directly observe the system running application code.

To address these challenges of hardware debugging, we developed a breakout board, shown in Figure 5(c), that includes an LCC-68 socket, pins for easily accessing and jumpering each signal, and an Epic programming port. With access to the full array of signals, hardware developers can easily probe every point in a design, connecting the circuit, multimeters, oscilloscopes, and other monitoring equipment as they see fit.

7. CARRIER BOARD CASE STUDIES

Carriers are circuit boards that act as substrates to glue together general-purpose modules with application-specific *sensors*, *power supplies*, and *mechanical constraints*. To evaluate the utility of our proposed architecture, we designed and implemented several different pilot-stage carrier boards. These case studies illustrate how our decomposition allows new platforms to be designed quickly by novice graduate students (usually in a few days), fabricated inexpensively on typically two-sided circuit boards (for a few hundred dollars), and easily hand-assembled (in hours, by the same students who designed the carriers). Table 4 summarizes these carriers and their differences.

Carrier	Modules	Sensors	Power	Mechanical
HydroWatch	Core	T, H, L	solar	enclosure
ACMeter	Core	V, C	AC	enclosure
BenchMark	Core, USB	T, H, L	USB	Telos-like
Meraki	Core	T, H	Meraki	Meraki

Table 4: Despite their unique application requirements, all carriers incorporate the same mote core. Sensors: temperature (T), humidity (H), light (L), voltage (V), and current (C).

7.1 External Sensor and Solar

The literature on sensornet applications shows many platforms built for monitoring animals and the environment, a subset of which employ a solar power subsystem for sustainable operation including ZebraNet [26, 39] for tracking the location of zebras in African savanna and jungle, Solar Dust [35] for measuring the penetration of light under shrub thickets in former grasslands, and Fleck [36] for measuring soil moisture and tracking livestock behavior on farms.

Building on this line of research, we have developed a new platform to study to the hydrological cycle in forested environments. Each microweather sensing node, shown in Figure 7(a), consists of a small waterproof box containing a carrier board, batteries, and carefully exposed temperature, relative humidity, and light sensors [37]. Additionally, RF requirements in the moist, dense forest require a high-gain antenna; thus, we export the U.FL connector provided by the Epic core module to an SMA connector and use an externally-attached 7-dBi omnidirectional antenna. The carrier board, shown in Figure 1, incorporates an Epic core module, a solar energy harvesting circuit with voltage and current monitoring, the iCount [19] system for measuring system energy consumption, and connections for the sensors. This 2-layer PCB was created using the freeware Eagle CAD Tool in less than two days and fabricated at a unit cost of $10.83 for a 60 piece build with a five day leadtime. The board took under two hours to populate by hand.

Previous incarnations of the HydroWatch node were built around a Telos family mote, resulting in a larger form factor (twice the enclosure size), insufficient exposed GIO and ADC pins (some desired measurements could not be taken), and a significantly higher cost (the Telos cost three times as much as the Epic). The new HydroWatch node design remedies these issues while achieving similar RF and energy performance. In terms of board fabrication cost, the previous 2-layer HydroWatch PCB took about a week to design, took two revisions to become operational, and cost $11.59 per board for a 54 piece build with a five day leadtime. Indeed, the Epic platform design flow has improved design flexibility while reducing time-to-result with comparable fabrication cost to previous methods.

7.2 AC Power Monitoring

Monitoring building energy consumption is an important opportunity for savings in an increasingly energy-conscious era; in fact, many AC power metering and control devices already exist and some are even network-enabled [28, 13]. However, since these are either commercially unavailable or cost prohibitive, we developed a platform for AC power metering, seen in Figure 7(b), to support research in energy-aware decision-making in datacenter and home environments both inexpensively and at scale. This platform includes a TRIAC for switching the AC load on and off, and an Epic core module for wireless communications.

The primary sensor of this platform is an industry standard IC that measures real, apparent, and reactive power by using a manganin current sensing resistor. Though the AC electricity presents a convenient source of power, the high mains voltage must be reduced, rectified, and regulated, for the DC circuitry including the Epic core module and related components. A standard approach is to use a transformer and a bridge rectifier, but this can be bulky. Recognizing the minimal DC current requirements of our design, a more cost-effective and space-saving way is to simply use capacitor dividers and a pair of diodes to shave off a small amount of AC current. This specialized design effort will not need

(a) HydroWatch Board (b) ACMeter Board (c) BenchMark Board (d) Meraki Daughterboard

Figure 7: Platforms for different applications have been built to evaluate the Epic architecture: (a) an environmental monitoring node incorporating solar energy harvesting and application energy metering, (b) an AC electricity meter for measuring building energy use, (c) a platform for sensornet testbeds with a USB interface, application energy metering, and a FIFO buffer for collecting and streaming high-frequency data, and (d) a Meraki Mini daughterboard that connects 802.3 and 802.11-based IP networks to 6lowpan-based sensor networks. Each platform was designed in less than a week using the same generalized core module while satisfying the specific requirements of the application.

to be repeated; the circuit can be replicated in future AC-powered platforms. One drawback to this approach is the floating ground, which may not be ideal for an experimental device, suggesting an isolation transformer may be a better choice for future revisions.

For the enclosure, rather than formulating a custom design, often both costly and time consuming, we selected the enclosure of an off-the-shelf AC power meter and designed our PCB within its restrictions. Thus, the board needed to accommodate not only a standard NEMA 5-15 AC plug and receptacle, but also a number of holes and contact points imposed by the clip design of the off-the-shelf enclosure. Since the Epic core module is a thin single-sided board, we were able to incorporate it easily within our volume constraints without facing the difficulty of connecting and accommodating a separate, larger mote inside the enclosure.

Additionally, we included an optically-coupled TRIAC to enable remote control of the current flowing to appliances connected to this AC meter. The TRIAC can also be used as a dimmer when combined with a zero-crossing output from the AC measurement chip. To save space and cut cost, we used a Planar Inverted-F Antenna (PIFA). Switching from the default U.FL antenna connector on the Epic core module involves switching a single capacitor. More challenging, however, is the RF engineering needed to match the Epic core, microstrip, and PIFA antenna. The antenna and its feedline are available as a script, allowing a simple way to change important parameters.

The design process for this board took one week using Eagle and fabrication of the 2-layer board cost $26.40 each for a quantity of 5 pieces and five day leadtime, while population of the prototype took roughly three hours. The results of this design cycle represent a cost and time commitment that are well within the constraints of most research budgets.

7.3 Testbed Replacement

The bulk of sensornet research over the last decade has largely been conducted on office or laboratory testbeds, fixed and reusable infrastructures of nodes with the network size and extent to enable researchers to investigate link, routing, and transport protocol dynamics without the overhead of constructing a deployment. Existing sensornet testbed architectures vary from flat networks of hundreds of mote-class devices [38, 17] to hierarchical networks interspersing mote-class devices with PC-class devices [22, 21] with backchannels that are USB, Ethernet, or even 802.15.4. However, data collection on current testbeds is generally constrained by limited memory and UART port baud rates on the on the MSP430F1611 and ATmega128L based mote families. The limitation prevents the collection of very high-frequency data such as noise floor information, application program state, scheduler context switches, or other debugging data. We developed the BenchMark mote with these limitations in mind.

The key elements of the BenchMark platform are an Epic core module, an Epic USB module for programming and interfacing, system-wide energy metering using the iCount system [19] with six decades of resistors for calibration over the entire operating range, and a 128 KB synchronous FIFO memory chip. This memory is meant as a high-speed queue for data generated by the application with a read/write time on the order of a few microcontroller instruction cycles. Beyond these components, this platform also incorporates a temperature/humidity sensor.

This platform was developed chiefly for conducting networking research and closing energy-measurement gap on a general-purpose testbed; given this requirement, the primary "sensors" are the iCount energy meter and the radio itself. Further, this platform is intended as a "drop-in" replacement for Telos family devices that already comprise a number of existing testbeds. This goal drove the selection of USB for the interface and power source and an internal PIFA (leveraged from the AC Meter carrier described in Section 7.2) for the antenna. The form factor is nearly identical to Telos.

Design of this platform was among the most recent of the carriers described in this section (only the Meraki Daughterboard was designed more recently). The design of the 4-layer board took roughly three days and fabrication cost $141.30 each for a quantity of 10 boards and a turnaround time of five days. Population of the prototype board took three hours. This time-to-result compares quite favorably with previous motes, which took months to develop and prototype.

8. DISCUSSION

Component reuse is a basic aspect of the building block approach to platform construction and carriers are no exception. The motivation for reuse comes from a desire to preserve the accumulated learnings and artifacts in moving through the phases of development, but this section also traces our experience with unplanned, organic reuse at the level of CAD parts and schematics.

Component	Type	Library	Breakout	DevBoard	HydroWatch	ACMeter	BenchMark	Interface	Meraki
Core	module	Epic	●	○	○	○	○	○	○
USB	module	Epic	●	○			○	○	
Storage	module	Epic	●						
ProgPort	part	Epic		●	○	○			○
Header17	part	Epic	●	○				◐	
LED0603	part	Epic	●	○	○		○	○	○
Socket68	part	Epic	●	○				○	
Headers	part	HydroWatch			●	○		◐	○
MAX1724	part	HydroWatch			●		○	○	
Switch	part	HydroWatch			●	○	○	○	○
Schottky	part	HydroWatch			●				
Zener	part	HydroWatch			●				
ZXCT1010	part	HydroWatch			●				
ADE7753	part	ACMeter				●			
DualPlug	part	ACMeter				●			
R-AXIAL	part	ACMeter				●			
R2010	part	ACMeter				●			
R0603	part	ACMeter				●			
MCP1700	part	ACMeter				●			
RSENSE	part	ACMeter				●			
AC PLUG	part	ACMeter				●			
PIFA Ant	script	ACMeter				●	◐		
74HC138	part	Epic					●		
74V293	part	Epic					●		
74LVC1G00	part	Epic					●		
Trimpot	part	Epic						●	
Phidgets Conn	part	Epic						●	
		Parts Reuse	0%	83%	33%	30%	61%	70%	100%

Table 5: Tracking component reuse over time. The listed components were created specifically for the carrier in question. Components from the Eagle CAD library or other third-party libraries are neither listed above nor included in the reuse statistics. ● identifies the carrier for which a component was originally made and in which it was first used. ○ identifies a carrier that uses a particular component. ◐ indicates a carrier for which a pre-existing component was modified and then used.

We demonstrated the viability of this approach by building a handful of application-specific carrier boards from a collection of modules but, in the process, we discovered two curious things. First, reuse occurs at the CAD parts, schematic, and parts inventory level as well as at the module level. Designers use parts and circuits created by their colleagues or stocked in the lab rather than create new CAD parts themselves or choose parts that must be ordered from distributors. This suggests that we should encourage greater reuse by sharing our niche part libraries more broadly and creating platform development kits that bundle many of these common pieces. Table 5 illustrates the benefits of doing so.

A second observation is that there is little overlap in electronic parts between modules and carriers. Even discrete parts like 10 kΩ pull-up resistors or 0.1 μF decoupling capacitors are different. The module designs, driven by space constraints and anticipating machine assembly (of the modules themselves but not necessarily the carriers), use smaller surface mount parts (e.g. 0402). The carrier board designs, constrained far less by space and anticipating hand assembly (at least for pilot runs) use larger surface mount parts (e.g. 0603 or 0805). This limited overlap in part usage provides some evidence that our modularity hits a design sweet spot; modules and carriers appear well-optimized for their particular purpose. Indeed, the first article of every carrier board presented in this paper was hand-assembled while almost exactly the opposite is true for the modules.

The development of many systems proceeds through the familiar phases of prototype, pilot, and production and while the engineering activities undertaken in each phase are very different, accruing the experiences and intellectual property through the phases is important. The modular architecture proposed in this paper supports such a fluid development model and we believe this approach to sensornet platform design is the first to support all three phases of sensornet development well enough for rapid progress.

9. CONCLUSION

This paper argues for a building block approach to hardware platform design that partitions functionality between general-purpose modules and application-specific carriers. A key principle of this approach is for modules to export as wide an electrical interface as possible rather than a narrowly-defined system bus. Lowering the hardware abstraction level "below the bus" facilitates greater module reuse, more compact designs, increased integration simplicity, and lower overall part count. And, by supporting many physical interconnect options for modules including socketing, soldering, and hardware inlining, this approach supports prototype, pilot, and production system development well enough for rapid progress. An important benefit of decomposing platforms in this way is that modules capture working hardware designs, making hardware libraries a natural extension. In the future, we envision others will create many new modules – like solar harvesting, signal conditioning, or high-precision clocks – and share them broadly to support rapid forward going innovation.

10. ACKNOWLEDGMENTS

Special thanks to Gary Myers, Jonathan Hui, Phil Buonadonna, Lama Nachman, and the anonymous reviewers for their insightful and constructive comments. This material is based upon work supported by the National Science Foun-

dation under grants #0435454 ("NeTS-NR") and #0454432 ("CNS-CRI"), a grant from the Keck Foundation, an NSF Graduate Fellowship, a Microsoft Graduate Fellowship, and generous gifts from Aginova, HP, Intel, Microsoft, and Sharp.

11. REFERENCES

[1] ATMega AT45DB161D Flash Memory. http://www.atmel.com/dyn/products/product_card.asp?part_id=3772.

[2] Crossbow IRIS OEM Module. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/IRIS_OEM_Datasheet.pdf.

[3] Crossbow MICA2 Mote. http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf.

[4] Crossbow MICA2Dot Mote. http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2DOT_Datasheet.pdf.

[5] Crossbow MICAz OEM Module. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_OEM_Edition_Datasheet.pdf.

[6] Intel iMote. http://www.intel.com/research/exploratory/motes.htm.

[7] Maxfor TIP. http://maxfor.co.kr/sub5_1.html.

[8] Sensinode. http://www.sensinode.com.

[9] Sentilla Tmote Mini. http://www.sentilla.com/pdf/eol/Tmote_Mini_Datasheet.pdf.

[10] Sentilla Tmote Sky. http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf.

[11] STMicroelectronics STM25P80 Flash Memory. http://www.st.com.

[12] UC Berkeley SmartDust Project. http://robotics.eecs.berkeley.edu/~pister/SmartDust/.

[13] Watts Up? .NET Electricity Meter. https://www.wattsupmeters.com/secure/products.php?pn=0&wai=32&spec=2.

[14] A. Y. Benbasat and J. A. Paradiso. A compact modular wireless sensor platform. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 56, Piscataway, NJ, USA, 2005. IEEE Press.

[15] J. Beutel, O. Kasten, F. Mattern, K. Roemer, F. Siegemund, and L. Thiele. Prototyping Wireless Sensor Network Applications with BTnodes. In *Proceedings of the 1st European Workshop on Wireless Sensor Networks (EWSN 2004)*, 2004.

[16] S. Blanchard. Quick Start Crystal Oscillator Circuit. In *Proceedings of the IEEE 15th Biennial University/Government/Industry Microelectronics Symposium*, 2003.

[17] B. N. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, and A. Vahdat. Mirage: A Microeconomic Resource Allocation System for Sensornet Testbeds. In *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNets '05)*, 2005.

[18] H. Dubois-Ferriere, R. Meier, L. Fabre, and P. Metrailler. TinyNode: A Comprehensive Platform for Wireless Sensor Network Applications. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN'06)*, 2006.

[19] P. Dutta, M. Feldmeier, J. Paradiso, and D. Culler. Energy Metering for Free: Augmenting Switching Regulators for Real-Time Monitoring. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN'08)*, 2008.

[20] N. Edmonds, D. Stark, and J. Davis. Mass: modular architecture for sensor systems. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, pages 393–397, apr 2005.

[21] J. Elson, S. Bien, V. Bychkovskiy, A. Cerpa, D. Ganesan, L. Girod, B. Greenstein, T. Schoellhammer, T. Stathopoulos, and D. Estrin. EmStar: An Environment for Developing Wireless Embedded Systems Software. *UCLA CENS Technical Report No. 9*, 2003.

[22] E. Ertin, A. Arora, R. Ramnath, V. Naik, S. Bapat, V. Kulathumani, M. Sridharan, H. Zhang, H. Cao, and M. Nesterenko. Kansei: A Testbed for Sensing at Scale. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, 2006.

[23] B. Greenstein, C. Mar, A. Pesterev, S. Farshchi, E. Kohler, J. Judy, and D. Estrin. Capturing High-Frequency Phenomena Using a Bandwidth-Limited Sensor Network. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys'06)*, 2006.

[24] D. Halperin, J. Ammer, T. Anderson, and D. Wetherall. Interference Cancellation: Better Receivers for a New Wireless MAC. In *The 6th Workshop on Hot Topics in Networks (HotNets VI)*, 2007.

[25] J. L. Hill. *System Architecture for Wireless Sensor Networks*. PhD thesis, University of California, Berkeley, 2003.

[26] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '02)*, 2002.

[27] S. Katti, S. Gollakota, and D. Katabi. Embracing Wireless Interference: Analog Network Coding. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '07)*, 2007.

[28] J. Lifton, M. Feldmeier, Y. Ono, C. Lewis, and J. A. Paradiso. A Platform for Ubiquitous Sensor Deployment in Occupational and Domestic Environments. In *Proceedings of the 6th international Conference on Information Processing in Sensor Networks (IPSN '07)*, 2007.

[29] D. Lymberopoulos, N. B. Priyantha, and F. Zhao. mPlatform: A Reconfigurable Architecture and Efficient Data Sharing Mechanism for Modular Sensor Nodes. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN '07)*, 2007.

[30] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks. In *ACM Transactions on Database Systems*, 2005.

[31] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proceedings of the 2nd ACM Conferences on Embedded Networked Sensor Systems (Sensys'04)*, 2004.

[32] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling Ultra-Low Power Wireless Research. In *Proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN'05)*, 2005.

[33] G. J. Pottie and W. J. Kaiser. Wireless Integrated Network Sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[34] B. Schott, M. Bajura, J. Czarnaski, J. Flidr, T. Tho, and L. Wang. A modular power-aware microsensor with >1000x dynamic power range. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 66, Piscataway, NJ, USA, 2005. IEEE Press.

[35] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter. LUSTER: Wireless Sensor Network for Environmental Research LUSTER: Wireless Sensor Network for Environmental Research. In *Proceedings of the 5th ACM Conference on Embedded Networked Sensor Systems (SenSys'07)*, 2007.

[36] P. Sikka, P. I. Corke, P. Valencia, C. Crossman, D. Swain, and G. Bishop-Hurley. Wireless Adhoc Sensor and Actuator Networks on the Farm. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN'06)*, 2006.

[37] J. Taneja, J. Jeong, and D. Culler. Design, Modeling, and Capacity Planning for Micro-Solar Power Sensor Networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN'08)*, 2008.

[38] G. Werner-Allen, P. Swieskowski, and M. Welsh. MoteLab: A Wireless Sensor Network Testbed. In *Proceedings of the 4th international Conference on Information Processing in Sensor Networks (IPSN '05)*, 2005.

[39] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware Design Experiences in ZebraNet. In *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, 2004.