# Software Development Tools Optimize ZigBee Performance

**By Anders Lundgren, IAR Systems**

*As ZigBee gains traction in the home automation and light industrial arenas, the right development tools make the difference in power savings.*

The ZigBee wireless protocol has been around for a number of years, but is just beginning to make a significant impact on the market. The technology offers an economical, robust solution for short-range applications like home automation and utilities metering. One of the core value propositions for ZigBee is its ability to provide very long term operation—more than a year, in some cases—on a single charge. Achieving this kind of performance requires more than well-designed hardware, however. It requires concise code that has been optimized to guide the system through the required tasks with minimum power demand. Given this, it’s time for a review of the protocol, device trends, and a closer look at what software development tools can do to yield a successful product.

**ZigBee 101**

The ZigBee technology, based on the IEEE 802.15.4 standard, has a potentially huge market in the area of wireless control of industrial and home automation. Like Bluetooth, ZigBee is a personal area networking (PAN) protocol for short-range applications, but, unlike Bluetooth, it has focused on very low power consumption to enable equipment to run unattended without requiring a battery change for up to a year or more.

Like WiFi and Bluetooth, ZigBee operates in the 2.4 GHz ISM band and uses spread spectrum modulation. Additional frequency allocations are at 868 MHz in Europe and 915 MHz in the Americas, the traditional frequency bands for short-range wireless applications. Table 1 summarizes the main characteristics of the systems in each band.

| Band | Coverage | Data rate (kb/s) | Channels |
|---|---|---|---|
| 2.4GHz | Worldwide | 250 | 16 |
| 915MHz | Americas | 40 | 10 |
| 868MHz | Europe | 20 | 1 |

**Table 1:** A ZigBee band shares established short-range frequency allocations with Bluetooth and WiFi, with the characteristics summarized here.

Table 2 shows a comparison of the ZigBee specification with that of WiFi 802.11b and Bluetooth. As you can see, given the characteristics of these three systems, WiFi best suits applications that demand high data rates, have access to a ready power source, and for which a star topology is practical—ideal for airport lounges filled with bored travelers browsing the Internet. Bluetooth suits simpler devices communicating over smaller ranges and with fewer nodes per master, with less frequent need for power input—typified by “hands free” mobile phone headsets. ZigBee’s key attributes of very low power demands and very flexible networking topology mean that it lends itself to communications between simpler devices where limitations such as slow data rates and small stack size are not critical.

| Feature(s) | IEEE 802.11b | Bluetooth | ZigBee |
|---|---|---|---|
| Power Profile | Hours | Days | Years |
| Complexity | Very Complex | Complex | Simple |
| Nodes/Master | 32 | 7 | 64000 |
| Latency | Enumeration up to 3 Seconds | Enumeration up to 10 seconds | Enumeration 30ms |
| Range | 100 m | 10m | 10m-300m |
| Extendibility | Roaming Possible | No | YES |
| Data Rate | 11Mb/s | 1Mb/s | 250kb/s |
| Stack size | 100+ kbyte | 100+ kbyte | 8-60 kbyte |
| Topology | Star | Star | Star, cluster, mesh |
| Security | Authentication Service Set ID (SSID), WEP | 64 bit, 128 bit | 128 bit AES and Application Layer user defined |

**Table 2:** A comparison of the different wireless technology specifications highlights how the characteristics of each makes it a “stand out” choice for a particular set of circumstances.

## ZigBee in the home

With these characteristics of low power consumption, multiple nodes, short range, low data rate and simplicity of specification, it is no surprise that typical ZigBee profiles can be found in applications such as home automation devices. These include on/off switches on mains electrical outlets; remote control of audio and video equipment; hands-free manipulation of curtains or blinds; lighting operation, including dimming; occupancy sensors; temperature sensors; and remote control of heating and air conditioning systems.

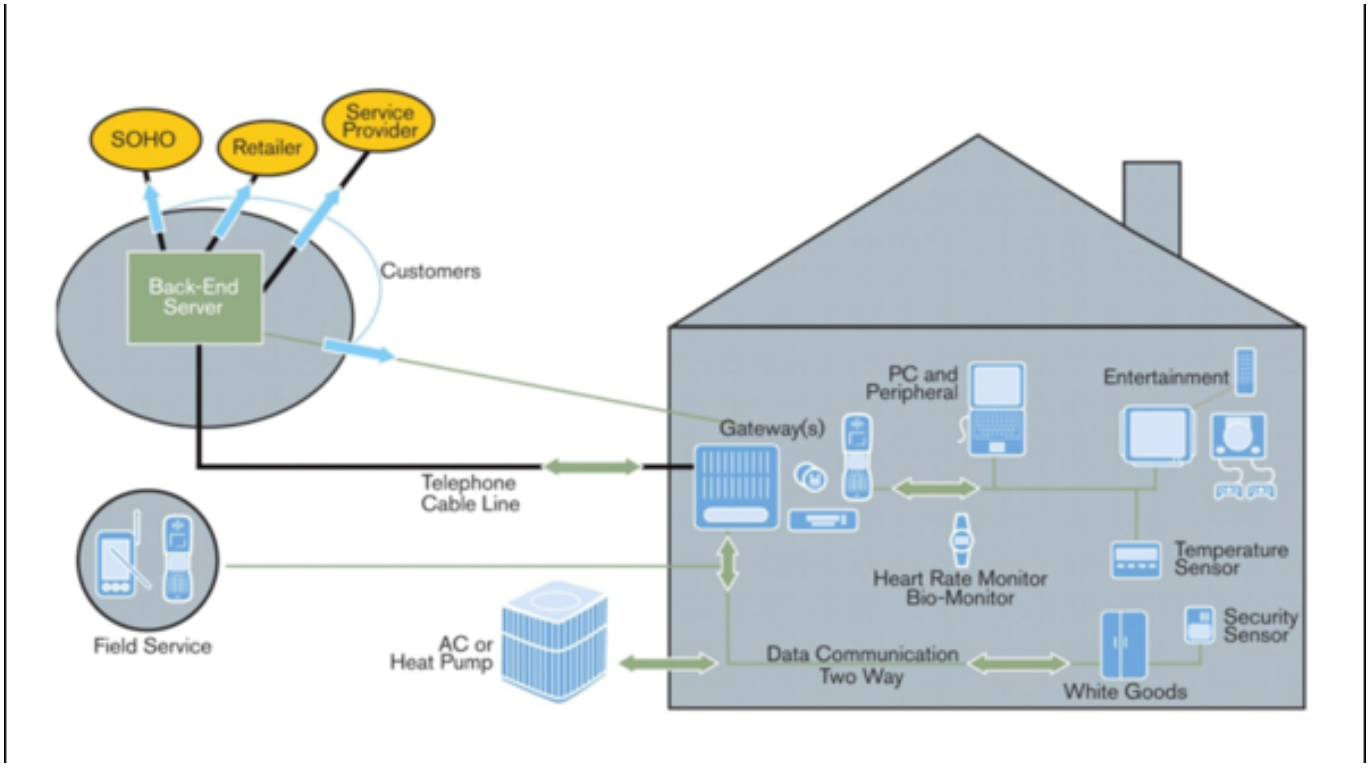


**Figure 1:** ZigBee applications in a domestic environment include devices concerned with medical, security, entertainment and climate control. Several of these applications and more can be found in light industrial environments.

Other present and likely applications include traffic management systems and commercial or retail lighting where a mesh network is used to transmit data over longer distances, passing data through intermediate devices to reach more distant ones. ZigBee is also prominent in plug-in electric vehicle (PEV) technology, serving an expanding industry through applications for charging, installation, configuration and firmware download, prepay services, user information and messaging, load control, demand response, and common information and application profile interfaces for wired and wireless networks.

**Development tools for efficient software**

Although the ZigBee protocol holds enormous promise, systems need to be properly developed and deployed in order to reap the benefits. The specification of an optimal microcontroller is important, and many providers of ZigBee solutions use 8-bit architectures such as 8051, AVR, MSP430, or most popularly the ARM7 or ARM Cortex-M0/M3 32-bit cores. Given that the optimal controller might vary over the lifetime of the product, development teams might prefer to opt for a development environment that is independent of the device vendor to ensure full support of a

wide range of architectures. That way, the selection of a different device for each application does not require mastery of a new integrated development environment (IDE).

Traditionally, minimizing power consumption has fallen to the hardware engineers. In an active system, power consumption depends not only on the design of the hardware but also on how it is used—which, in turn, is controlled by the system software. Implementation complements the attributes of the protocol in that it should be compact and implement an efficiently developed code base. While this is partly the responsibility of the source code developer, the right development tools will go a long way toward helping deliver the most compact, power efficient code. Not only do the microcontrollers being used in ZigBee devices have relatively small memory sizes, but also it takes longer to execute larger code, resulting in more power consumption when the code size is larger.

Writing concise code is not always the solution in and of itself, however. Concise code can still contain power-siphoning errors such as initialization of ports that won't be used, overly high clock rates, etc. Using a technique called power debugging, software developers can map the power consumption of their systems to the program’s instruction sequence, allowing them to discover and remove errors in source code that increase power consumption. These debugging capabilities will aid in writing result code that is as energy efficient as possible without compromising the performance of the application.

In power debugging, a probe connected to the target system monitors power consumption as a function of the instruction trace or in the case of the ARM Cortex-M3/M4 cores, for example, the program counter (PC) sampling facility. This setup allows the developer to download the application and execute while monitoring power consumption graphically or in a log window. Clicking on consumption peaks allows the developer to trace back directly to the specific section of code. Sometimes the offending piece of code may be the initialization of a port that is not being used, or the failure to properly leverage idle time or sleep/low-power modes. Power debugging tools make it possible for developers to find these errors, modify the code, and check to see the results of their changes.

The critical determinant of battery life for a ZigBee system is the beacon interval. This is the time interval between successive bursts of communication between the sensor and the controller—analogous to the ‘handshake’ in cellular systems. In between the communication bursts the device should be put in a low power mode, ideally for the duration of the beacon interval. The longer the beacon interval, the less power is consumed, as shown in Figure 2.

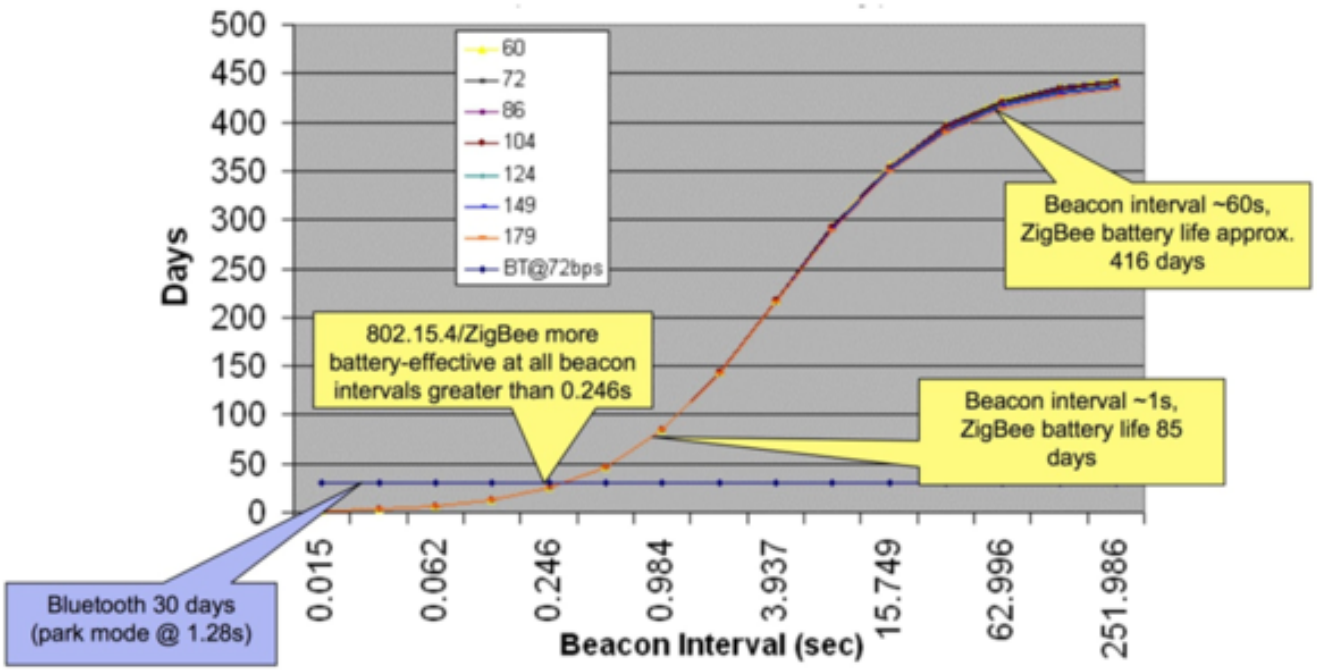


**Figure 2:** This comparison of ZigBee and Bluetooth battery lifetimes under different beacon intervals explains why power saving is such a positive ZigBee attribute.

The length of the beacon interval that can be tolerated depends on the individual application. For a passive monitoring system such as a temperature sensor, 60 s might be perfectly acceptable, but for adjusting lighting levels, for instance, this latency would be intolerable as less than one second would be required. Power debugging allows the developer to not just monitor power usage over shorter or longer beacon intervals but helps highlight sections in which the processor can be put into power saving mode.

**Club ZigBee**

No protocol can be successful in our complex engineering environment without an advocacy group to oversee the standard, coordinate the revision and update process, and advance interoperability. The 280-member ZigBee Alliance is a diverse ecosystem of companies providing everything from radio semiconductor chips to finished products, also encompassing service, support, tools and testing.

A related initiative, the ‘IPSO Alliance,’ promotes the internet protocol as the network technology of choice for connecting smart objects around the world. This will entail all kinds of everyday objects that can be controlled by a computer, each being assigned their own IP address, and compatible with remote monitoring and interrogation using a variety of interoperable technologies including ZigBee.

In today's fast-moving marketplace, time-to-market is every bit as important as quality of product. By leveraging the ZigBee and IPSO Alliances to find vendors positioned to deliver development products that are integrated with other offerings in the food chain, OEMs can ensure that the development process as efficient and rewarding as possible.

**Conclusion**

The ZigBee standard is now being adopted for home and industrial automation applications, bringing with it the benefits of low power, flexible topology, and support for a high node count. Part of the ZigBee value proposition is extreme low power consumption that allows remote monitoring to take place without the need for battery maintenance over months or even more than a year. To fully leverage the promise of the standard, OEMs need to optimize software as well as hardware, though. That's where software development tools can make an enormous difference. Power debugging provides a powerful resource to maximize efficiency while maintaining full functionality in the most appropriate and cost-effective device possible. With the proper toolset, ZigBee OEMs can rapidly deliver systems to market that provide the performance, convenience, and efficiency that their customers expect.

**About the Author**

Anders Lundgren has been with IAR Systems since 1997. Working initially with compiler design and development, he has played a key role in architecting the IAR ARM toolchain and infrastructure. Mr. Lundgren is a regular speaker at seminars and conferences covering coding and debugging methodology. He has an MSCS and has been working with electronics, software and embedded systems since childhood.

September 24, 2012

**Source URL (retrieved on *09/17/2014 - 12:30am*):**
http://www.wirelessdesignmag.com/blogs/2012/09/software-development-tools-optimize-zigbee-performance?qt-digital_editions=0