Embedded ZigBee Design Considerations

By Andy Wheeler

With the ratification of the ZigBee wireless networking standard, companies are integrating complete wireless SoCs on semiconductors smaller than a shirt button, enabling systems designers to embed low-cost, low-power sensor and control capabilities to everyday devices. Today, ZigBee-compliant networks are already being built into home monitoring and control products such as smoke detectors, so that the unit in the basement can wirelessly notify the occupants on the third floor in case of trouble, or even dial the homeowner's telephone. Paintings in art galleries each have their own networked disturbance sensor, discretely informing the guards if any painting has been touched. Room air conditioning units for hotels can be controlled remotely &#151 and their efficiency monitored from a central location &#151 saving hotel owners thousands of dollars every week. And ZigBee-enabled utility meters will potentially save public utility companies millions of dollars by eliminating the need to manually read electric, gas and water meters at homeowners' premises.

But although the ZigBee v1.0 specification has finally been ratified, the protocol is not a one-size-fits-all blueprint for companies wishing to tap into this promising market. At its most basic level, ZigBee simply ensures interoperability with other standard-compliant products. The deeper application, architectural and platform issues that designers must carefully weigh are as wide-ranging as the potential ZigBee applications themselves.

ZigBee-compliant embedded networks have several characteristics that set them apart from conventional wireless local area networks such as IEEE 802.11. A node in a ZigBee network has the following attributes:

Low power&#151 Many applications must run unattended for years using battery

power. Commercially available ZigBee system nodes consume approximately 50 mW when active and less than a microwatt in sleep mode. With a battery source of three watt-hours, such systems will last two and a half days if active 100% of the time, but with a duty cycle of 0.1%, the same system will last more than six years without a battery change.

Relatively low data rates&#151 An average light switch communicates about four bits per day, or 46 microbaud. In many applications, high data rates are not a requirement. The ZigBee v1.0 specification defines a communication rate of 250 kb per second, which is more than adequate for low-rate sensing and control applications.

Low costs&#151 ZigBee networks are engineered to be built into a broad spectrum of products, many of which are cost-sensitive. It is now possible to purchase ZigBee-compliant SoCs for less than $5, making them suitable for many commercial and consumer applications. Naturally, Moore's Law holds for RFICs as well as for other silicon devices, so the price of ZigBee semiconductors will continue to fall over time and enable ever-broader applications.

Reliable, autonomous operation&#151 Because a ZigBee network typically has many more nodes than human attendants, nodes must be easy to deploy, operate reliably without human intervention and be self-healing. A failure of a single node should not interrupt the operation of the entire network.

Based on these attributes, the following is a high-level &quotpunch list&quot of some of the less-understood application considerations systems designers face before designing embedded ZigBee solutions, including issues such as network topologies, interoperability options, security levels, platform differences and development environments.

Architectural Considerations

The ZigBee standard provides network, security and application support services operating on top of the IEEE 802.15.4 MAC and PHY wireless standard. It employs a suite of technologies to enable scalable, self-organizing, self-healing networks that can manage a variety of data traffic patterns.

Although ZigBee is often thought of as synonymous with wireless mesh networking architectures, the standard actually supports several network topologies, including star, cluster tree, or star/mesh hybrid networks. Consequently, the first consideration a developer should address is: &quotWhich network configuration best suits my application needs?&quot

In any network &#151 wired or wireless &#151 the reliability of communication between any pair of nodes is degraded by three phenomena: attenuation, interference and multi-path. In wireless networks, data travel over an uncontrolled medium &#151 the airwaves &#151 and signal degradation becomes an important consideration in the design of the network.

Attenuation&#151 Radio waves become attenuated with distance, just as candlelight becomes dimmer as you move away from the candle. In free space, the attenuation follows an inverse square law &#151 doubling the distance results in a 4X reduction in amplitude &#151 but in physical spaces with lots of clutter, path loss exponents of three or four are common. Radio waves are further attenuated when passing through solid materials such as doors or walls.

Interference&#151 When electromagnetic sources generate signals in the same frequency band as the transmitted radio signal, the radio receiver experiences interference. Interference can come from intentional radiators, such as other radio transmitters, or from unintentional radiators, such as microwave ovens. Spread spectrum techniques, whether frequency-hopping or direct-sequence, offer some degree of immunity from interferers.

Multipath&#151 When a transmitted radio signal arrives at a receiver by means of two or more paths (e.g., due to reflections off nearby objects), the multiple signals combine at the receiver's antenna. The wavelength for 2.4 GHz ISM band is about 4.8 inches, so when direct and reflected paths differ by multiples of 4.8 inches plus 2.4 inches (one half wavelength), the signals cancel one another.

If data reliability is key, mesh network architectures offer the best protection against these signal degradation phenomena. By placing ZigBee receivers and transmitters closer together, the destructive effects of all three conditions are reduced. The redundant paths of mesh networks ensure alternate data path routes and no single point of failure should a node on the network fail. A standard home automation network controlling proximity lights, access and HVAC systems is a

typical example.

But other applications may require ZigBee routers to extend the range of the network by acting as relays for nodes that are too far apart to communicate directly. Moreover, the deployment may rely on battery-powered routers that need a significant amount of &quotsleeping&quot downtime to preserve their lifetime. Consider a crop monitoring network or similar agricultural application. Here, a cluster tree configuration would work best. It can aggregate multiple sub-networks into a long distance ZigBee WAN in which low-data-rate traffic trickles along the tree, awakening the battery-powered routers only when needed to pass or receive data between the appropriate subnets.

Conversely, some shorter-range applications may be better suited using a star topology where the communications overhead of mesh network routing may not be necessary. Raymarine, a leader in marine electronics, developed a remote autopilot system for sports fishing boats using a ZigBee-compliant system in a star topology. The application has primarily one hop between the ZigBee controller and an end-device. Using the beaconing feature specified within 802.15.4 is one way of meeting Raymarine's strict requirements for long battery life.

Security

Most ZigBee solutions will require some level of security to be designed into the devices. Fortunately, ZigBee provides a standardized toolbox of security specifications and software. It is based on a 128-bit AES algorithm and incorporates the strong security elements of 802.15.4. ZigBee stack profiles define security for the MAC, network and application layers. Its security services include methods for key establishment and transport, device management and frame protection.

If developers choose to use a public ZigBee profile, then the security decisions for their applications have already been made for them; they are pre-defined in the profile. And chances are that even if a developer intends to build a private profile application, the choice will be the security mode in one of ZigBee's pre-defined stack profiles.

At this level, developers need to tackle issues such as their need to encrypt the payload of the data frame, as well as the length of the authentication code (8-bit, 16-bit, 64-bit, etc.) tagged on to the end of the frame. Some basic applications may

not require authentication, and thus can benefit from a smaller packet payload. These data integrity options allow developers to make tradeoffs between message protection and message overhead. A thermostat message passing to a controller, for example, will probably have a limited range of valid values (such as a number between 60&#176 and 80&#176) and thus not require encryption, but it would be important to ensure the source is authenticated.

Developers must also decide where to apply security: at the MAC, network or application layer. If the application needs the strongest security possible, secure it at the application layer. Security implemented here uses a unique session key that can only be authenticated and decrypted by another device possessing the key. This approach protects against both internal and external attacks, but it requires more memory to implement.

Security at the MAC and network layers serves essentially the same purpose: to secure single-hop transmissions. The MAC layer inherently mediates access to the shared medium and controls single-hop transmissions between neighboring devices. The ZigBee Alliance added a network layer security option to include additional functionality not available at the MAC layer, including the ability to reject data frames if their freshness cannot be verified. But both these security layers use a global key that all ZigBee devices on the network share. MAC and network layer security is good for applications that need protection against certain kinds of infrastructure attacks; for example, protection against a nefarious device maliciously inserted into the network. If a developer needs to establish a route between two devices and the network layer frames are not secure, that device could intercept the packets.

ZigBee security also introduces the concept of a &quotTrust Center,&quot which allows devices into the network, distributes keys and enables end-to-end security between devices. Here, developers must choose between two Trust Center modes based on their application. *Residential mode* is lightweight, but does not establish keys and does not scale with the size of the network. *Commercial mode* establishes and maintains keys, and scales well, but at the cost of significantly more memory.

Interoperability

Although ZigBee is an open standard, it grants systems designers great latitude in how much of their product they want open to third-party vendors. As previously noted, the ZigBee standard defines only the network, security and application interface layers. Designers can choose to license the entire ZigBee stack, which can

include application profiles for a specific product, or merely license the networking layer for basic network level interoperability.

At the application layer, systems designers must decide whether to use a public application profile or develop their own private profile. ZigBee v1.0 has already defined basic public profiles for lighting, with application profiles for HVAC, industrial sensors and other sensors in the pipeline. Any company may design products compatible with products sporting public profiles. A fluorescent lighting ballast vendor using the public ZigBee lighting profile can interoperate with third-party light-switch dimmers that also use the profile. Developers can simply add their own look and feel to the public profile.

Although it may seem contradictory to ZigBee's spirit of openness, some systems designers may choose to develop products that do not provide open interoperability at the application layer. They may choose to design private application profiles to create a closed &quotecosystem&quot of single-vendor devices, or select third-party devices. ZigBee defines an abstract interface while platform vendors provide APIs that define rules for how applications integrate into the ZigBee stack. For instance, vendors of commercial HVAC systems, who must make significant investments in the installation of their customers' HVAC infrastructures, may want to protect that investment from erosion to third-party thermostats and other environmental controls. A developer should look into the functionality of the platform vendors' APIs before choosing their solutions.

Yet, the &quotclosed&quot HVAC system would still benefit from third-party product manufacturers at the network level. Required interoperability at ZigBee's lower stack provides data routing as well as medium access control, network formation and maintenance, and device and service discovery. So a company installing new ZigBee wireless light switches in the building, for example, would be able to take advantage of the existing transport network provided by the HVAC controllers to help carry traffic.

How those data are transported is another key consideration for developers because ZigBee does not define a transport layer. Developers must decide whether to build the transport mechanism themselves, or simply build their application using a ZigBee chip with a built-in transport layer. For example, Ember provides a transport layer with its ZigBee stack as a way to simplify application development as well as ensure reliable end-to-end messaging. The built-in transport layer provides the framework on which developers can define private ZigBee profiles.

Platform Considerations

ZigBee provides a standardized network and application framework upon which wireless systems designers can build applications without having to worry about the intricacies of networking and RF issues. Yet ZigBee's standardized framework does not by itself ensure easy product development. The market is flush with a diverse vendor mix of components needed to build ZigBee-compatible applications, including RF transceivers, microcontrollers, Flash ROM, vendor-specific protocol stacks and application development tools. Consequently, designers must choose between &quotrolling their own&quot ZigBee solution from multi-vendor components or building their applications upon an integrated hardware/software platform. Lacking both the expertise and inclination to tackle the tough integration effort, most systems designers would probably opt for the latter approach.

The first decision they face stems from the complexities and incompatibilities between hardware and network software. Problems found in development of a new application are rarely confined to just one layer of the stack. For example, a bug found in the MAC layer of a prototype can rarely be detected and debugged by network layer providers. So designers need to carefully consider the integration depth of their chosen platform.

A ZigBee system requires an RF transceiver, a microcontroller or digital signal processor for application processing, and a ZigBee network stack. Until recently, most were multi-chip, multi-software solutions born from multi-vendor partnerships. But today a new generation of fully integrated, single-vendor platforms are starting to hit the market. A multi-vendor platform sometimes offers developers lower upfront costs. But a single-source platform, while sometimes more expensive, lets designers avoid the headaches, finger-pointing and product development delays inherent in solutions built from different vendors' hardware and software products.

Wireless systems designers also need to consider the depth of the networking stack. Typically, the deeper the stack, the easier the development effort. A stack that provides everything from the physical network layer to the transport layer all the way up to the ZigBee profiles will shield developers from the inner workings of the network, freeing them to focus 100% of their engineering efforts on application development. For example, a ZigBee application built with a mixed vendor solution would require developers to factor in how their application handled dropped network packets between nodes. This kind of networking complexity would be already accounted for in a single-source, full-stack platform.

Tool choices can also make or break a successful ZigBee development effort. Developers need to carefully consider the breadth and functionality of their platform's development environment. Does the vendor provide a full development kit complete with multi-node wireless semiconductors, networking software, software tools, training and technical support? Are the tools part of an integrated environment with universal interface, or a collection of stand-alone tools? Does the development environment include sufficient tools for debugging, such as over-the-air monitoring, boot loading and Sniffer-like packet testers?

Conclusion

ZigBee standardization has opened up exciting new markets for wireless sensing and control applications in home and building automation, automatic meter reading, industrial and environmental controls, medical devices and even toys. Wall Street conservatively estimates the market will exceed $8 billion over the next few years. But ZigBee's standardization should not be considered a universal development panacea. The standard was designed to give wireless systems designers great flexibility in developing highly differentiated applications, even to the point of creating &quotclosed&quot solutions. The onus is on the ZigBee developer to carefully consider all the application, architectural and platform issues before committing to the application design.

About the Author
Andy Wheeler is chief technology officer at Ember Corporation.