# Use A Kernel In Your IoT Designs

Christian Legare | *Electronic Design*

*Jun 3, 2014*



In the Internet of Things, 80% of all devices will be deeply embedded and will require 32-bit processors for communication. The use of a kernel provides the required software architecture and efficiency.

I won't attempt to define the Internet of Things (IoT). Given its current state of development, many technologies and communication protocols still need to be defined. However, that's also why there is room for innovation and definition. We have enough of the building blocks today to make the IoT a reality, so no one should hesitate to start developing products or systems to capture a portion of this emerging market.

RELATED

» Who's Doing What In IoT And M2M?
» The Connected World Awaits
» Understanding The Internet Of Things
» The Internet Of Things Is A Standards Thing

An IoT system comprises many components, including the thing itself (the embedded device), local networking (wired or wireless), the Internet, new IoT protocols, and, finally, some type of cloud-based subsystem. It's clear, though, that 32-bit processors will be prevalent for the embedded devices in IoT systems.

There still will be many 8-bit and 16-bit processors for low-power sensors and actuators.

But because of the IoT's communication requirements, 32-bit processors will surpass any other processor architecture.

Download this article in .PDF format
This file type includes high resolution graphics and schematics when applicable.

While many embedded systems manage well with single-threaded application, networked devices require more capable software. This means the software for an IoT device must be:

• Scalable, to accommodate a wide range of different classes of devices

• Modular, so you can choose only the components you need to meet tight RAM requirements

• Connected, so you can move data in and out of the device via Wi-Fi, Ethernet, USB, Bluetooth, or a wireless sensor networking technology

• Reliable, so your device can be certified for safety-critical applications

These requirements can be met with the use of a kernel. In addition, a kernel allows the establishment of a modular software architecture, facilitating development, troubleshooting, and product upgrades.

**Why Not Linux?**

Linux is a robust, developer-friendly operating system (OS) that has been getting attention as a platform for IoT devices. It has matured into a mainstream embedded operating system for many applications. Yet it has a disadvantage compared to a real-time operating system (RTOS): memory footprint.

Linux can be trimmed down by removing tools and system services that aren't needed in embedded systems, yet it's still a large piece of software. It simply will not run on 8- or 16-bit MCUs, and many newer 32-bit MCUs don't have enough onboard RAM for the Linux kernel.

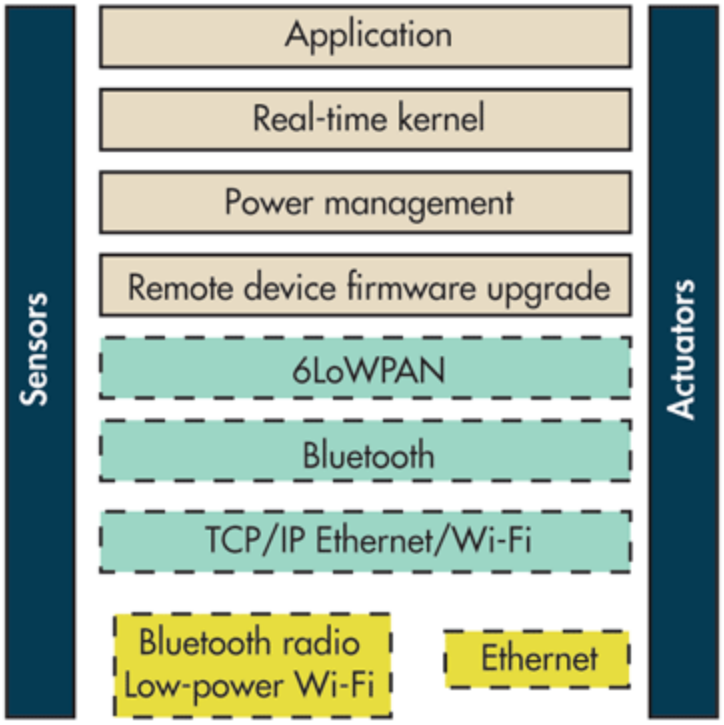
The ARM Cortex-M series is a good example. Hundreds of different MCUs that typically have only a few hundred kilobytes of onboard memory are based on the popular Cortex-M architecture.

Linux will certainly have many uses in embedded devices, particularly ones that provide graphically rich user interfaces. But there are thousands of applications for which Linux is ill suited.

**Industrial Vs. Consumer IoT**

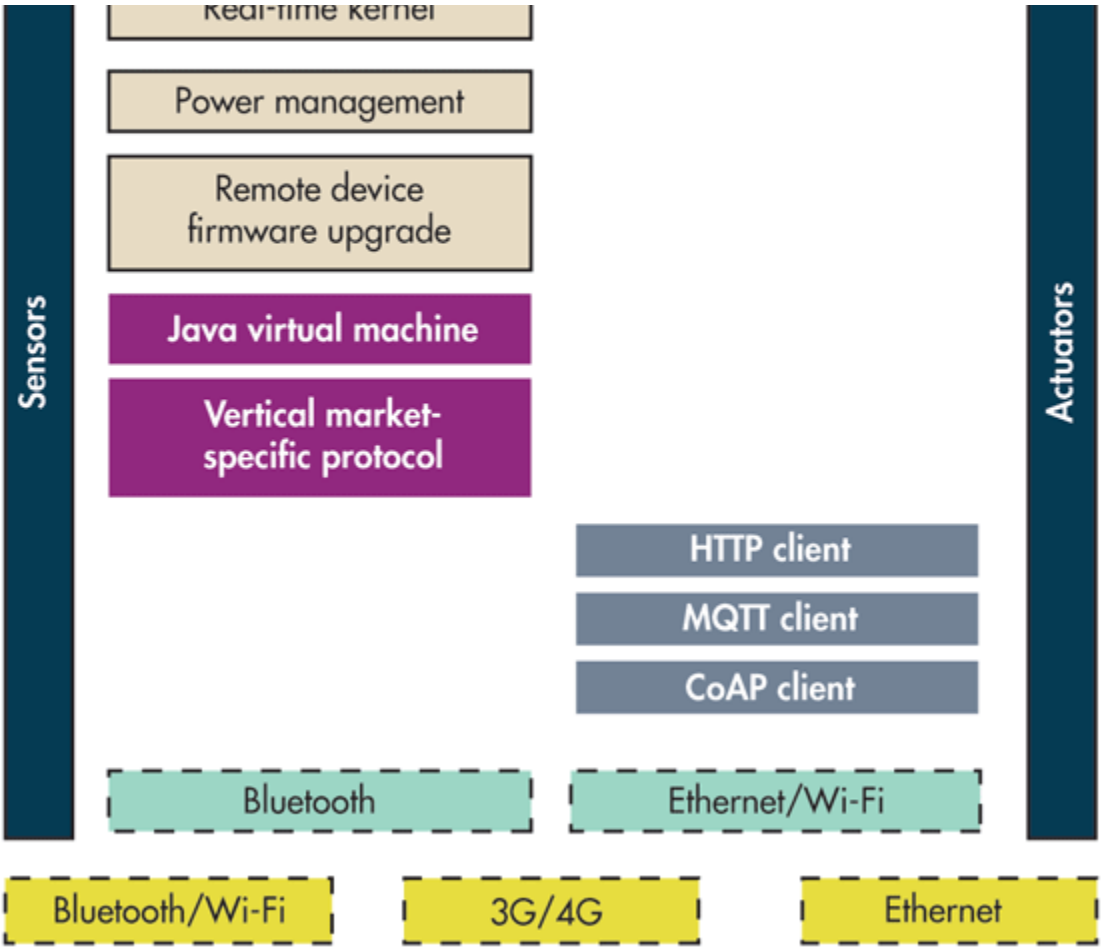
The software requirements for industrial and consumer IoT devices can differ quite a bit. Although they might share a common kernel and low-level services, the middleware required by their applications can be radically different.

Figure 1 represents a software stack for an industrial IoT device, such as a wireless sensor node. This low-power, low-cost device may run entirely on a battery. Such a device might typically use a Cortex-M0 or Cortex-M3/M4 MCU. It would use a highly efficient network protocol such as 6LoWPAN to reduce transmission time and save power. It also would communicate over short distances wirelessly using Bluetooth or low-power Wi-Fi, or else use Ethernet when employed as an edge node.




**1. A typical low-power industrial IoT device may incorporate one or more wired or wireless network interfaces.**

Figure 2 conveys a software stack for a consumer IoT device. Clearly, the software requirements for this device are much greater. It might need a Java VM. It also may well use a vertical market protocol such as AllSeen, HomePlug/HomeGrid, Continua Alliance, or 2net. Such a device typically might use a Cortex-M3/M4 or a Cortex-A processor.




**2. The software stacks for a consumer IoT device may incorporate one or more client profiles.**

Ultimately, these requirements will drive your RTOS choice, so the platform choice shouldn't dictate a device's functionality. In fact, a flexible, scalable RTOS can help increase return on investment, cut development costs, and reduce time-to-market.

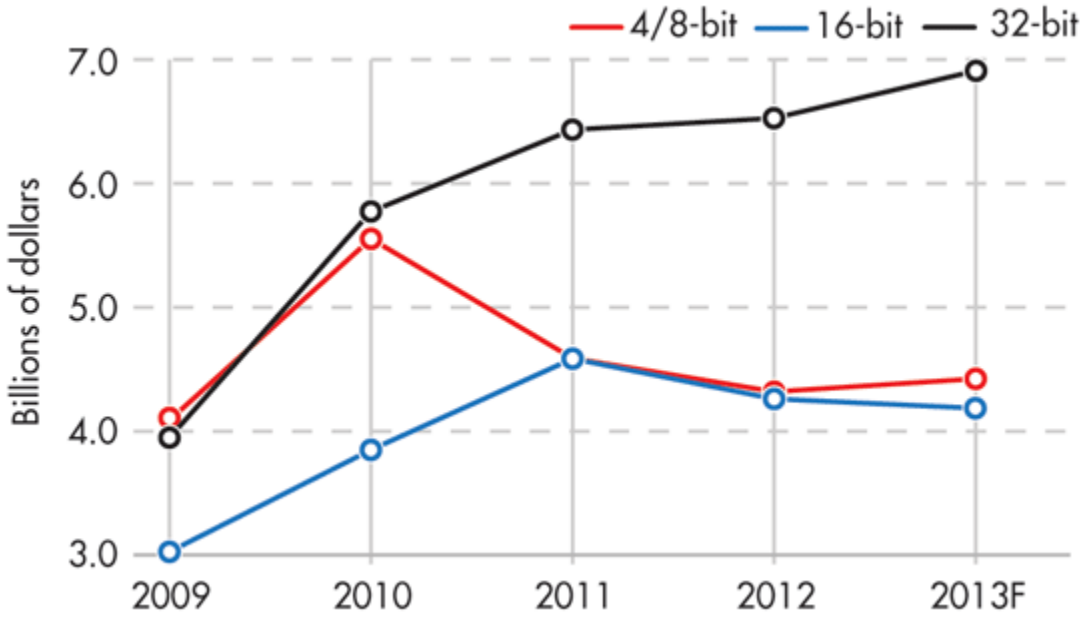
**Scalability**

Although deeply embedded systems historically have been built entirely around 8- and 16-bit MCUs, the price of 32-bit MCUs has been dropping rapidly. As they have become commodity products, their popularity for embedded devices has skyrocketed.

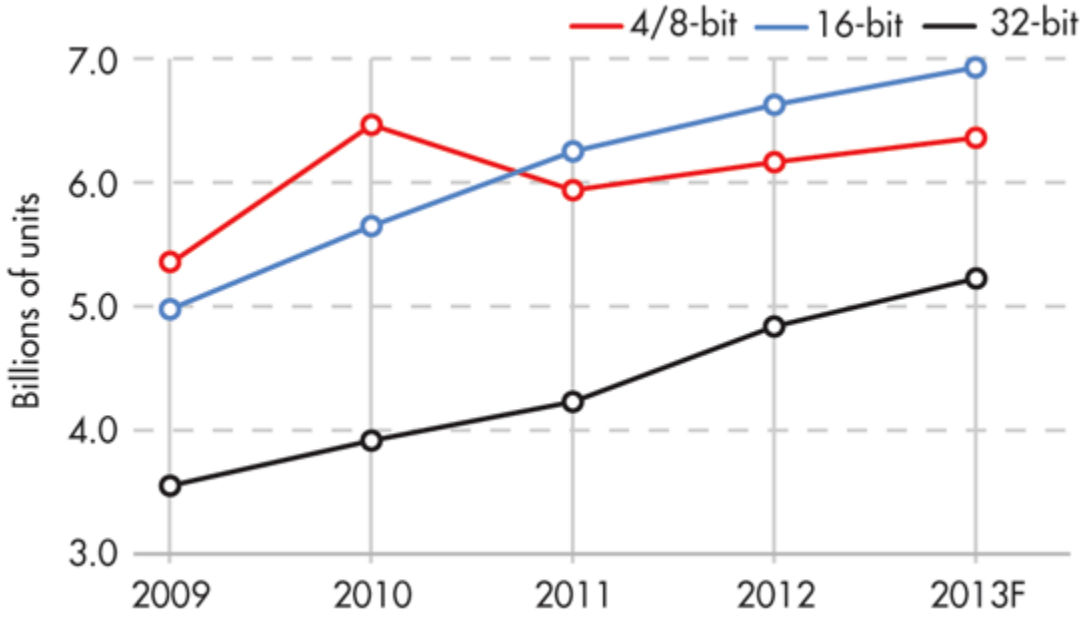
A common engineering solution for networked sensor systems is to use two processors in the device. In this arrangement, an 8- or 16-bit MCU is used for the sensor or actuator, while a 32-bit processor is used for the network interface. That second processor runs a RTOS.

Sales of 32-bit MCUs have exploded in the last several years, and they have become the largest segment of the MCU market. The 32-bit MCU segment alone is expected to grow to

$19.2 billion by 2017 *(Figures 3 and 4)*.




**3. Sales of 32-bit MCUs continue to increase. (courtesy of IC Insights, April 25, 2013)**




**4. All MCU unit shipments are increasing. (courtesy of IC Insights, April 25 2013).**

IoT devices will still include a mixture of small and large MCUs for years to come. A scalable RTOS that runs on a variety of 16- and 32-bit MCUs will meet tight memory requirements and reduce processor demands to save money.

**Modularity**

IoT devices also will require a modular OS that separates the core kernel from middleware, protocols, and applications. The OS will provide ease of development and minimize the software's memory footprint.

Using a modular RTOS simplifies the development process, especially when developing a family of devices with different capabilities. Relying on a common core allows the entire family of devices to share a common code base, while each device is customized with only the middleware and protocol stacks required by the application.

This approach also allows for a smaller memory footprint in the device. Unlike a monolithic operating system that bundles an entire suite of software together, a modular RTOS allows the embedded software to be tailored for the device, requiring less RAM and flash memory while reducing costs.

**Connectivity**

Network connectivity is essential to the IoT. Whether we are talking about wireless sensor nodes in a factory or networked medical devices in a hospital, the industry now expects embedded devices to be connected to each other and to communicate with corporate or public networks.

To achieve this, the RTOS needs to support communications standards and protocols such as IEEE 802.15.4, Wi-Fi, and Bluetooth. The device must be able to connect to Internet protocol (IP) networks using bandwidth-efficient protocols such as 6LoWPAN.

An RTOS will allow you to select the specific protocol stacks needed, which again means saving memory on the device and reducing cost. It also can help retrofit existing devices with new connectivity options without reworking the core of the embedded software.

**Reliability**

Many IoT systems will be deployed in safety-critical environments or in locations where repair and replacement are difficult. IoT devices, then, will need to be faultlessly reliable.

In these situations, an RTOS must have safety-critical certification. This is vital to demonstrate the device's reliability and safety. Certifications that may be required include:

• DO-178B for avionics systems

• IEC 61508 for industrial control systems

• ISO 62304 for medical devices

• IEC SIL3/SIL4 for transportation and nuclear systems

When building products for use in a safety-critical environment, using software that is already certified can reduce certification time and costs. Every part of the device will require certification and extensive documentation. Validation suites and certification kits, typically available from third parties, provide thousands of pages of documentation that otherwise would have to be developed for each product.

Even if certification isn't required for the device, knowing that the OS running within it has been certified can provide confidence and peace of mind that your product will perform reliably.

**Conclusion**

Embedded devices used as things in the IoT will predominantly use 32-bit processors mainly because of the networking requirements. The use of a kernel, especially an RTOS, provides the software architecture and efficiency required for this type of embedded design.

An IoT system may use small sensor nodes running on low-power smaller processors up to large gateways running on application processors. Designers must invest in an RTOS that can run on these types of processors. And while RTOSs may have been seen as a commodity in the last decade, the IoT is giving a new life to them and to embedded systems in general.

***Christian Legare**, executive VP and CTO, joined Micrium as vice president in 2002. Previously, he spent 22 years in the telecom industry as an executive in such large-scale organizations as Teleglobe Canada and in engineering and R&D startups. He was in charge of an Internet Protocol (IP) certification program at the International Institute of Telecom (IIT) in Montreal, Canada. He is a regular speaker at the Embedded Systems Conferences in Boston and Silicon Valley and has published several articles on embedded systems. He holds a BSEE and MSEE from the University of Sherbrooke, Quebec, Canada. He can be reached at* christian.legare@micrium.com.

## Related Articles

The Connected World Awaits

Interview: Ritesh Tyagi Discusses Embedded Trends

Understanding The Internet Of Things

Manage Network Traffic In Your 100-Gbit/s Designs

Integrate The Design, Validation, And Verification Steps In Your Next ZigBee Radio

**Submit Articles**

**View Mobile Site**

## Electronic Design Related Sites

**Machine Design SourceESB Microwaves & RF Power Electronics Hydraulics & Pneumatic**

**Defense Electronics Global Purchasing Electronic Design Europe**