Embedded COMPUTING DESIGN

NOV/DEC 2016
VOLUME 14 | 7
EMBEDDED-COMPUTING.COM

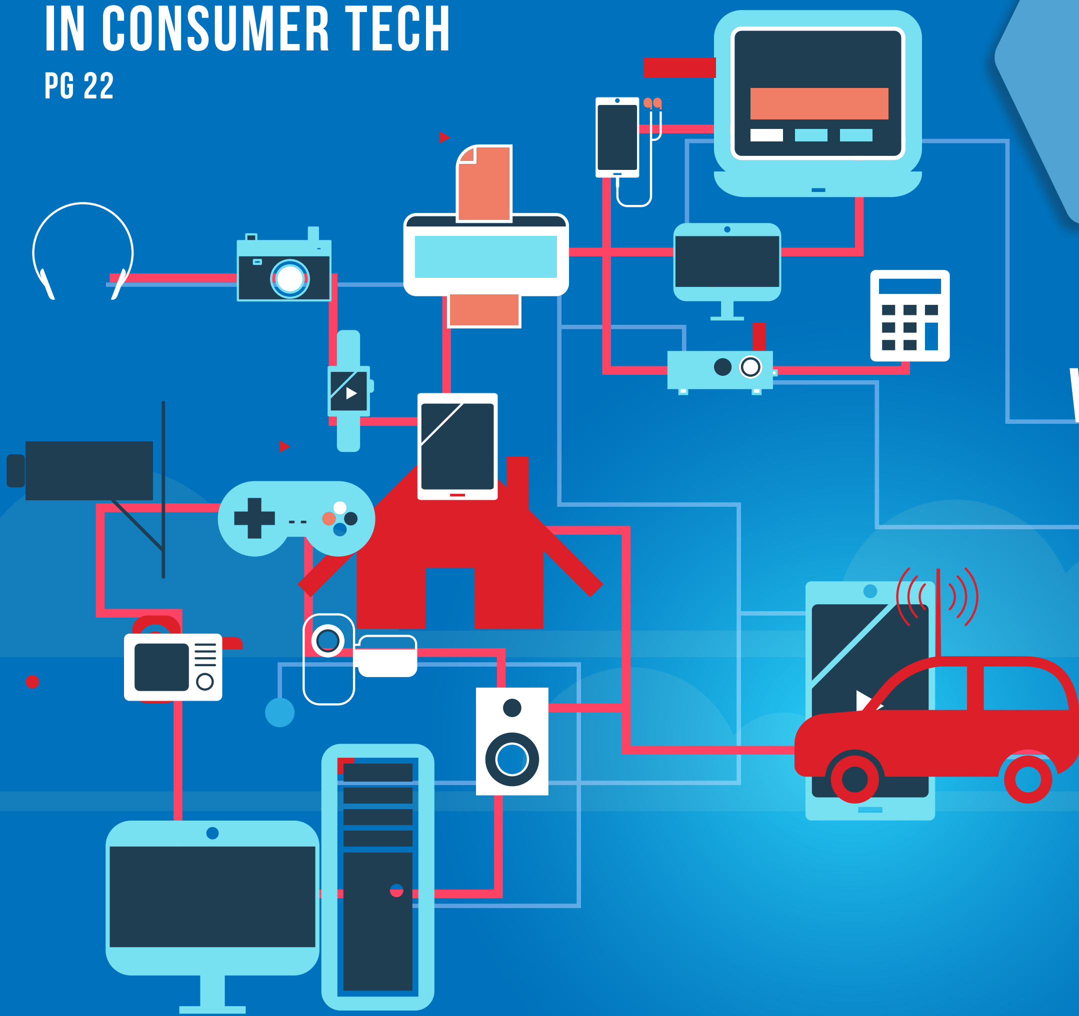
DEVELOPMENT KIT SELECTOR

Digi-Key ELECTRONICS

www.embedded-computing.com/designs/iot_dev_kits

EMBEDDED IS EVERYWHERE IN CONSUMER TECH

PG 22



RTOS IN IoT: WHERE WE'RE HEADED

PG 16

RISC-V: REIGNITING PROCESSOR INNOVATION

PG 10

CONTENTS

Embedded
COMPUTING DESIGN

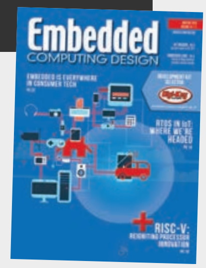
Nov/Dec 2016 | Volume 14 | Number 7

opsy.st/ECDLinkedIn @embedded_comp

FEATURES

COLUMNS

COVER



Embedded technology is everywhere around us, even in our phones, cars, and homes. In the November/December 2016 issue of *Embedded Computing Design*, the embedded technology you might not know is in your consumer devices is revealed, the benefits of real-time operating systems (RTOSs) for the Internet of Things (IoT) are explained, and new processor architectures for innovation and security are introduced.

WEB EXTRAS

DOWNLOAD THE APP

Download the Embedded Computing Design app

iTunes: itun.es/iS67MQ
Magzter: opsy.st/ecd-magzter

Published by:

OpenSystems media

ISSN: Print 1542-6408
Online: 1542-6459

ENVIROINK

blewis@opensystemsmedia.com

TOP TECH TO WATCH AT CES 2017: BUILDING BLOCKS FOR AUTONOMOUS DRIVE

By Brandon Lewis, Technology Editor

It's hard to believe, but once again it's time for the most exciting (and exhausting) show of the year – the Consumer Electronics Show (CES).

CES 2017 will be my third, and while each of the last two has showcased innovative advances in the segments of wearables, drones, and augmented/virtual reality (AR/VR), CES is as much an automotive show these days as anything else. The event allows attendees to see underlying embedded technologies side by side with the vehicles that employ them, and glimpse features that will be incorporated into the model years of the not-so-distant future.

One of the best examples of this from CES 2016 came at the QNX Software Systems booth, where the company showcased an automotive emergency stop simulation that prevented an "accident" on the show floor. There, the QNX team outfitted a Jeep Wrangler, Toyota Highlander, and impromptu traffic signal with Cohda Wireless MK5 telematics boxes based on the NXP/Cohda RoadLINK chipset and running QNX's ISO 26262-certified OS for Safety. The MK5 devices – on-board units (OBUs) in the case of the vehicles and a roadside unit in the case of the traffic signal – communicated wirelessly over the IEEE P1609 protocol for wireless access in vehicular environments (WAVE), and the OBUs were also connected via the CAN bus in order to communicate with other vehicular subsystems. As a result, the Jeep and Toyota believed they were traveling in various directions at certain speeds when fed manipulated timestamp data and GPS coordinates, which formed the basis for several potential accident scenarios surrounding an imagined four-way intersection.

Here, the QNX OS for Safety played multiple key roles. First, the OS for Safety was responsible for managing the networking and connectivity stacks on the MK5 boxes, as well as the corresponding inbound/outbound GPS notifications indicating the speed and orientation of both vehicles. Second, after running GPS data through a thread analysis engine, the OS for Safety needed to reflect that information through the CAN bus to warn drivers of an imminent collision, which could have been performed using haptic or auditory measures but was done using a red warning light indicator on each vehicle's instrument cluster. Given the safety-critical nature of the application, this process had to be conducted within the millisecond and sub-millisecond latency requirements of vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications.

The OS for Safety has two attributes in particular that enabled last year's demo and will usher in the advanced driver assistance systems (ADAS) and autonomous driving technology of the years to come, with one being a microkernel real-time operating system (RTOS) architecture and the other being a feature called adaptive partitioning.

The OS for Safety is based on QNX Neutrino, a deterministic microkernel RTOS that encapsulates each software component (including individual protocol stacks) into separate memory-protected regions. This architecture allows software to automatically restart and execute even in the event of faults, but also facilitates a preemptive design paradigm in which the kernel can be interrupted by external requests and then execute those threads based on a pre-determined scheduling priority. In the context of the V2X demo described previously, this implies that, regardless of the function being performed, the system is able to rapidly switch context and process tasks related to a GPS notification, predictably and on deadline.

Adaptive partitioning extends that preemptive design characteristic to management of the processor itself through a unique capability that permits the full utilization of a CPU while still guaranteeing temporal isolation (in other words, the time and resources dedicated to a particular task). Beyond typical scheduling algorithms and as opposed to fixed static partitioning that reserves part of a processor's core(s) for a dedicated operation, adaptive partitioning allows idle portions of a processor waiting to compute a pre-defined executable to be reallocated to other threads that could use those resources immediately, and then revert back to the original task as soon as it is ready to be processed. This has obvious benefits in terms of efficiency, but also paves the way for the vehicle architectures of tomorrow.

With the consolidation of electronic control units (ECUs) in next-generation automobiles and the possibility that one ECU could be responsible for multiple vehicle subsystems, there will not only be a need for software technologies like adaptive partitioning, but safety-certified hypervisors that can ensure the separation of safety-critical and non-safety-critical functions running on the same system on chip (SoC) as well. These are the embedded building blocks that will take us into the age of autonomous driving, and *that* is what I want to see at CES 2017.

To watch a video of QNX Software Systems CES 2016 booth demo, go to embedded-computing.com/videos/qnx-emergency-stop-ces-2016/.

AD LIST

SOCIAL

Facebook.com/
Embedded.Computing.Design

@Embedded_comp

LinkedIn.com/in/
EmbeddedComputing

Pinterest.com/Embedded_Design/

Instagram.com/
Embedded_Computing

youtube.com/user/
VideoOpenSystems

Embedded
COMPUTING DESIGN

EMBEDDED COMPUTING BRAND DIRECTOR Rich Nass rnass@opensystemsmedia.com
EMBEDDED COMPUTING EDITORIAL DIRECTOR Curt Schwaderer cschwaderer@opensystemsmedia.com
TECHNOLOGY EDITOR Brandon Lewis blewis@opensystemsmedia.com
EUROPEAN EDITOR/TECHNICAL CONTRIBUTOR Rory Dear rdear@opensystemsmedia.com
CONTENT ASSISTANT Jamie Leland jleland@opensystemsmedia.com
DIRECTOR OF E-CAST LEAD GENERATION AND AUDIENCE ENGAGEMENT Joy Gilmore jgilmore@opensystemsmedia.com
CREATIVE DIRECTOR Steph Sweet ssweet@opensystemsmedia.com
SENIOR WEB DEVELOPER Konrad Witte kwitte@opensystemsmedia.com
WEB DEVELOPER Paul Nelson pnelson@opensystemsmedia.com
DIGITAL MEDIA MANAGER Rachel Wallace rwallace@opensystemsmedia.com
CONTRIBUTING DESIGNER Joann Toth jtoth@opensystemsmedia.com

SALES

SALES MANAGER Tom Varcie tvarcie@opensystemsmedia.com (586) 415-6500
STRATEGIC ACCOUNT MANAGER Rebecca Barker rbarker@opensystemsmedia.com (281) 724-8021
STRATEGIC ACCOUNT MANAGER Bill Barron bbarron@opensystemsmedia.com (516) 376-9838
STRATEGIC ACCOUNT MANAGER Eric Henry ehenry@opensystemsmedia.com (541) 760-5361
STRATEGIC ACCOUNT MANAGER Kathleen Wackowski kwackowski@opensystemsmedia.com (978) 888-7367
SOUTHERN CALIFORNIA REGIONAL SALES MANAGER Len Pettek lpettek@opensystemsmedia.com (805) 231-9582
SOUTHWEST REGIONAL SALES MANAGER Barbara Quinlan bquinlan@opensystemsmedia.com (480) 236-8818
NORTHERN CALIFORNIA REGIONAL SALES MANAGER Twyla Sulesky tsulesky@opensystemsmedia.com (408) 779-0005
ASIA-PACIFIC SALES ACCOUNT MANAGER Elvi Lee elvi@aceforum.com.tw
EUROPE SALES ACCOUNT MANAGER James Rhoades-Brown james.rhoadesbrown@husonmedia.com

OpenSystems media®
WWW.OPENSYSTEMSMEDIA.COM

PUBLISHER Patrick Hopper phopper@opensystemsmedia.com
PRESIDENT Rosemary Kristoff rkristoff@opensystemsmedia.com
EXECUTIVE VICE PRESIDENT John McHale jmchale@opensystemsmedia.com
EXECUTIVE VICE PRESIDENT Rich Nass rnass@opensystemsmedia.com
CHIEF TECHNICAL OFFICER Wayne Kristoff
GROUP EDITORIAL DIRECTOR John McHale jmchale@opensystemsmedia.com
VITA EDITORIAL DIRECTOR Jerry Gipper jgipper@opensystemsmedia.com
INDUSTRY EDITOR Jessica Isquith jisquith@opensystemsmedia.com
MANAGING EDITOR Jennifer Hesse jhesse@opensystemsmedia.com
ASSISTANT MANAGING EDITOR Lisa Daigle ldaigle@opensystemsmedia.com
SENIOR EDITOR Sally Cole scole@opensystemsmedia.com
ASSOCIATE EDITOR Mariana Iriarte miriarte@opensystemsmedia.com
CREATIVE PROJECTS Chris Rassiccia crassiccia@opensystemsmedia.com
FINANCIAL ASSISTANT Emily Verhoeks everhoeks@opensystemsmedia.com
SUBSCRIPTION MANAGER subscriptions@opensystemsmedia.com

CORPORATE OFFICE
16626 E. Avenue of the Fountains, Ste. 201 • Fountain Hills, AZ 85268 • Tel: (480) 967-5581

SALES AND MARKETING OFFICE
30233 Jefferson • St. Clair Shores, MI 48082

REPRINTS

WRIGHT'S MEDIA REPRINT COORDINATOR Wyndell Hamilton whamilton@wrightsmedia.com (281) 419-5725

rdear@opensystemsmedia.com

SMART HOME HINDERED BY EASE OF CONFIGURATION, COST

By Rory Dear, European Editor/Technical Contributor

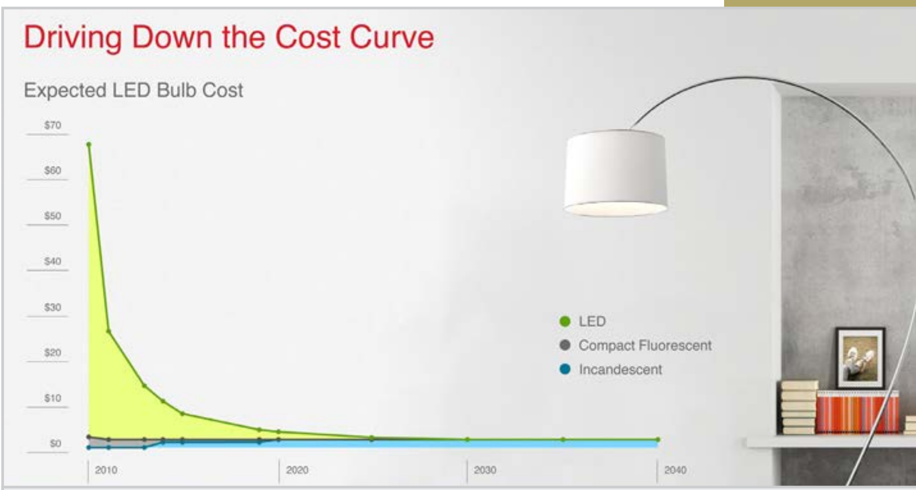
Back in the '90s, before online gaming involved centralized servers, connectivity was either via a local area network (LAN) or the untamable beast that was a direct cable connection. This involved dial-up modems connecting directly to one another over phone lines. I vividly recall running nearly a mile back and forth from my old school friend's house desperately configuring it. It worked for a few minutes, then lost connection – an experience us proud early adopters almost accept as par for the course with (what was) new technology.

Your typical consumer demands the opposite. It was they who Microsoft targeted with their famous plug and play drive in Windows 95 and, to be honest, even we proud early adopters expect a high degree of auto-configuration today. Technology is held back from becoming mainstream until its integration is simplified sufficiently to enable even the most technologically illiterate to "plug and play."

The other barrier is cost. An obvious example today is the stuttering deployment of smart LED lighting, which should be a warning to all of those involved in realizing consumer IoT, the technology's benefits are not enough. Skip Ashton, Vice President of Software at Silicon Labs warns, "The issue of going mainstream with IoT is cost." He cited that a typical U.S. household contains 40 light bulbs. At $70 per smart bulb, that's nearly $3000! Even combining all the current benefits of smart lighting, your typical householder is unlikely to perceive value at these prices. So how can cost be driven down?

Silicon Labs believes that this can be achieved through more highly integrated SoCs with a reduced number of integrated circuits (ICs), as well as economies of scale that will eventually result in smart and non-smart bulbs having near-equal bill of materials (BOM) costs. Its MGM111 module combines a 2.4 GHz Mighty Gecko SoC, high-efficiency chip antenna, crystals, and RF matching and shielding into a complete, ready-to-use mesh networking module supporting ZigBee and Thread protocol stacks. If smart light bulbs suddenly cost the same, or mere cents more than a non-smart bulb, who would buy the non-smart?

That question isn't just one of cost. As per my opening statement, if smart bulb installation demands a potential technophobe to enter a configuration world they aren't comfortable in, they'll stick with what they have. This installation simplification needs to stretch well beyond a linear example of smart bulbs. The smart home needs bulbs to seamlessly communicate with door locks, smoke and security alarms, and much more. There's a real risk that the big players, to protect their investments, will lock down their proprietary formats and consumers risk ending up with disparate systems with non-existent or weak interconnects.



"THE CONSUMER HAS A RIGHT TO DEMAND THAT ANY SMART DEVICE THEY BUY WILL INTEGRATE SEAMLESSLY INTO THEIR EXISTING SET UP ..."

To address this, Silicon Labs is driving the convergence of IP connectivity, enabled at application layer. The consumer has a right to demand that any smart device they buy will integrate seamlessly into their existing set up, and their existing set up is sufficiently future proofed supporting multiprotocol communication. They expect each smart home subset to immediately interconnect, with mere clicks enabling future smoke alarm activation, turning on all their lights, and sending notifications to all of the household's smartphones – those same smartphones we need to manage the entire system, within a single app. ECD

cschwaderer@opensystemsmedia.com

INTERNET OF THINGS PLATFORMS, SYNERGIES, AND KEY CONCEPTS

By Curt Schwaderer, Editorial Director

Internet of Things (IoT) platforms and use cases are abounding in key industries like utilities, communications, and retail. Vendors are emerging with platforms and frameworks for IoT to make end-to-end development and integration simpler. Are these platforms viable? Or is it too early?

The IoT value chain

Every link within an IoT deployment model must add value and perform its function or the entire solution may fail. Network delivery must be reliable and provide the proper quality of service; media and entertainment must be compelling and usable to consumers; and smart home and mobile IoT services must be useful and also provide information and analytics on how customers are using the services for vendors to improve their offerings.

"ONLY BUILDING VERTICAL IOT SOLUTIONS LIMITS WHAT CAN BE DONE WITH THEM. THIS HORIZONTAL APPROACH BECOMES A PLATFORM THAT CAN PROVIDE A WIDE RANGE OF OPTIONS TO OPTIMIZE AND EVOLVE THE IOT SOLUTION."

Greenwave Systems is an IoT software and services company that views the IoT as a broad play that spans the telecom, utilities, and retail segments, and its core business is helping large customers with sizeable consumer bases generate revenue from next-generation services. Jim Hunter, Chief Scientist & Technology Evangelist with Greenwave calls the organization "a software company with deep hardware chops," as their current CEO was previously the CTO of Linksys and held various positions at Cisco, which allows the organization to leverage understanding about the marriage between hardware and software in route to creating effective solutions.

"Sometimes you have to build the hardware if it doesn't exist," Hunter says. "For example, we've built a reference design for a next-generation broadband router for Tier 1 service providers. The reference design has no added bill of materials (BOM) cost, and we license our software so that the entire hardware/software solution works perfectly. In this example, there are over three million units now deployed in the US. This model is what we call 'AXON Engage.'"

Greenwave's AXON Engage model takes a horizontal approach to IoT solutions that ranges from set-top boxes and broadband routers to other core IoT devices. AXON gathers the information from various IoT sensors and devices, then translates them into a common, syntactical language so that a lightbulb, a car, or a set-top box can be controlled with the same application programming interface (API).

A second component of the approach involves software that connects devices to the cloud. This software leverages the syntactical language with the ability to include third-party software to control and manage the IoT system. The software includes a framework to stop, start, manage, or upgrade code for an IoT system. This software is wrapped in a container and can be included in and leveraged by IoT functional components or analytics applications.

"Only building vertical IoT solutions limits what can be done with them," Hunter says. "This horizontal approach becomes a platform that can provide a wide range of options to optimize and evolve the IoT solution."

Self-healing IoT

Another key concept involves "self healing." The ability to set service and availability thresholds across specific products with automated problem scenario detection and correction is extremely valuable. Self-healing systems can result in shorter and less frequent support calls, as well as enable information collection for system degradation indicators. Once the information is obtained, it can be analyzed and acted upon.

Analysis doesn't stop with self-healing, of course. Looking for overall use trends and developing specific policies around these trends provides the foundation for automating things for the user. The key is the ability to map out the relationship between the person and the technology to create the lowest possible friction or remove barriers to effective usage.

As for the single biggest factor in advancing IoT platforms, Hunter says, "We need to be user-centric instead of technology-centric. The problem with technology is that technologists deliver products that tend to be less friendly than they could be from a user interaction perspective. We need to elevate the level of interaction for the best possible user experience."

An example of this is the "Alexa-like" voice-activated communications that learn to tag objects with specific purposes or functions in a "noun/verb" fashion. This moves interaction toward assigning tags to the environment. For example, interacting with a system by saying, "Assign outside to my porch light" creates a tag so when I say "I'm going outside" and it's dark, the system turns on the porch light. This is a much different user experience than opening a home lighting app on a smartphone and touching the porch light icon.

Hunter says another key concept for IoT platforms is that it cannot simply be a primitive messaging structure. Platforms must be designed with a paradigm that can be built upon, and the best platforms can be used horizontally and interact asynchronously (HTML is an example of such a platform for the web); only assuming asynchronous communications is not a good platform characteristic.

"Hiring an IoT platform is similar to hiring good employees," Hunter says. "They must be trustworthy, reliable, and work well with others. To extend this paradigm even further, if I'm a CEO, should every sensor report directly to me? Of course not – too much information and bottlenecks would result. There should be a hierarchy. Sensors report to a controller, the controller publishes the summary output, and specialists can dig into the details."

Hunter has a tempered, pragmatic approach to the evolution of standards within the IoT domain. "Consortiums tend to be good to get the conversation started, but without a single company driving things, good work tends to stall. The real challenge arises when standards commoditize things. When companies and products lose their differentiation and become commoditized, we've reached the point where progress tends to slow down."

To Hunter, industry shouldn't expect to agree on a single standard or language, and rather accept the fact that the IoT is a multi-lingual environment. This amplifies the need to build a platform that understands all the viable elements and has the ability translate those up to a common model.

Additionally, Hunter identified the following key elements as critical to the advancement of IoT:

- Fortifying existing network technology through better network reliability – If you ultimately want to monetize a solution, you have to have a strong communications foundation.
- Extending into new networks and devices like mesh and radio technologies with an eye toward new and emerging topologies.

Smart homes, security, and privacy

Hunter highlights the connected home environment as an example of work Greenwave has done with companies to deliver specific solutions. "Our number one effort tends to be building out the right networks for the application. We fortify these technologies through software, then design elements where the voids exist."

The phased approach Greenwave employs starts with getting the IoT network right, then establishing a data model with a common addressable structure. As long as it's hierarchical, once the data model is defined you can start incorporating application-specific noun/verb tags and search paradigms, then apply analysis, feedback loops, and third-party contributions to adapt a smart home solution as use cases evolve.

IoT security is another key concept Hunter promotes, as Greenwave serves as co-chair on the Privacy and Security Committee within the Internet of Things Consortium. "The consortium is comprised of a broad mix of stakeholders that include networking, hosting, chip, software, and device companies, among others. They all promote the concept of having security at the forefront of any IoT design. To exist, everything should have security before it ever enters the IoT. Security is the mechanism by which we protect data and authenticate actions."

Privacy is a less-considered aspect, but one Hunter feels is still important. "Privacy tends to fall by the wayside in IoT because we have the word 'Internet' in front of this evolution. Mechanisms like cookies, pushed information, and even plugins have been used to gather information without our permission in the Internet world. Many companies have the supposition that things should and will continue this way in IoT. IoT will literally be in, on, and around you 24/7/365. Do the same web-browsing rules apply?

"For example, companies with business models that predicate taking your information without permission or returning value will find themselves facing challenges in the near future. We should look at the data of IoT as content. Because IoT takes large amounts of information about consumers from a wide set of sensors, it will be able to create content about the individual and their actions, communications, and interactions with others. Companies need to be mindful of this. Within five years they may be asked by a court of law to be accountable. We don't have the right density in IoT to see that right now, but as it matures it will become clearer that an alarmingly accurate picture can be painted about consumers, from an individual to groups and demographics." ECD

RISC-V: REIGNITING INNOVATION IN SEMICONDUCTORS?

By Brandon Lewis, Technology Editor

Do you remember when one advantage of the ARM architecture was its limited number of instructions? After all, the "R" in "ARM" stands for reduced instruction set computing (RISC), and the benefit of this smaller instruction set architecture (ISA) was that it was simpler to program than complex instruction set computing (CISC) ISAs such as the x86-64 instruction set (which currently includes more than 2,000 instructions[1]). The more intuitive approach to programming and the fact that instruction sets for all ARM CPUs up to version 2 were available in the public domain helped the architecture gain popularity, and eventually aided in the mass adoption of ARM-based processors in the market today.

However, as ARM IP took hold in segments ranging from mobile to networking to embedded systems, it became necessary to develop new cores tailored to those applications (Cortex-A, Cortex-M, and Cortex-R variants among them), and with them, new instructions. As a result, an ISA that originally included fewer than 100 instructions ballooned to a size now comparable to that of Intel's, and ARM eventually elected to make their instruction sets proprietary in order to maintain them in accordance with their burgeoning semiconductor IP business. Unfortunately, the upfront licensing fees for an ARM IP core can range anywhere between $1 million and $10 million depending on the design[2], making the microarchitectures, and hence, the ISA, impractical for consideration by many academics and small startups.

Realizing the complexity of the Intel ISA and the exclusivity of ARM's, a team of researchers at the University of California, Berkeley embarked on a what they thought would be a short project to develop a new RISC-based ISA. After completing the ISA in 2010 they made it available to the open source community, but received several complaints when attempting to make a change to it shortly thereafter. At that point the team realized the opportunity for an open-standard RISC ISA that was locked down – indefinitely. And so, RISC-V was born.

Fewer and freer

RISC-V is an open standard ISA, but as opposed to other open-source RISC implementations like OpenRISC, it is not standardized around a particular processor microarchitecture. Rather, RISC-V defines four base integer ISAs for 32-, 64-, and 128-bit processors that contain fewer than *50* hardware instructions each: the 32-bit RV32I; the 64-bit RV64I; the 128-bit RV128I; and the RV32E, which is a 16-register subset of the RV32I.

A number of standard extensions also exist, including the general-purpose ISA (classified extension "G"); integer multiply/divide (classified extension "M"); atomic memory operations (AMOs) and load-reserved/store conditional (LR/SC, classified extension "A"); single-precision floating-point (classified extension "F"); double-precision floating-point (classified extension "D"); quad-precision floating-point (classified extension "Q"); and an optional compact subset for reduced code sizes (classified extension "C"). In addition, vector and crypto extensions are currently underway in working groups at the RISC-V Foundation.

RISC-V was designed to operate independently of microarchitectural features, and, as mentioned, once work on an ISA component is finished by the Foundation it is frozen, forever. The ISA also provides clear separation between system software layers, with straightforward communications between an operating system (OS) and the system or applications facilitated through a system binary interface (SBI) and application binary interface (ABI), respectively (Figure 1).

What this means for chipmakers is that, outside of the locked down opcode space reserved for the base ISAs and extensions (both current and planned), the Foundation has guaranteed not to intrude on remaining greenfield opcode space with future modifications. That greenfield opcode space is therefore available for developers to create their own instructions, allowing the standard to be extended in a proprietary way that, as Ted Speers, Head of Product Architecture and Planning for the SoC Group at Microsemi and a member of the Board of Directors at the RISC-V Foundation reveals, allows them to innovate around the "secret sauce that is really in the microarchitecture that you build around it."

"What's important about the ISA at the end of the day is it's not important. What's important is that people can build and innovate a computer architecture with something that is

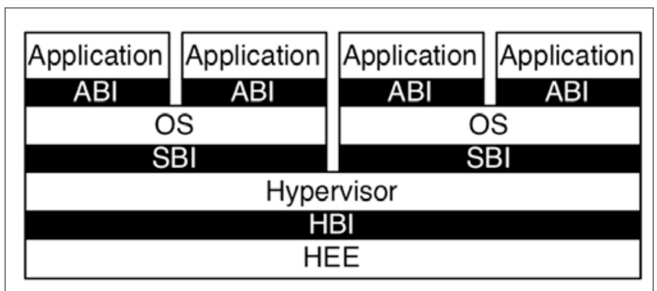


FIGURE 1 The RISC-V ISA communicates with the system execution environment (SEE) through a system binary interface (SBI), with the application execution environment (AEE) through the application binary interface (ABI), and with the hypervisor execution environment (HEE) through a hypervisor binary interface (HBI) in systems that employ a hypervisor.

well done," Speers says. "Ultimately, the goal is that RISC-V becomes the common language of all new computer architectures going forward.

"Although at its root it's just another ISA, a lot of thought has gone into making RISC-V extensible, so there are natural groupings of instructions that you might want to use," he continues. "For a microcontroller (MCU) you might just want to use the compressed instructions, or not the floating-point instructions. The point there is that the language is built and architected with different extensions that are appropriate for your application."

The minimalist flexibility and technology agnosticism of the RISC-V ISA makes it applicable in processor designs ranging from embedded MCUs to high-performance data center processors, as well as in heterogeneous system on chips (SoCs) that include multiple IP cores. Traditionally, the various IP in complex SoCs has required vast amounts of expertise spread out over large teams just to get the assorted cores talking to one another, much of which can be eliminated through comprehensive RISC-V libraries, such as those being developed by NVIDIA.

Equally as important as the technology is the fact that RISC-V is license- and royalty-free, and operates under a BSD open source license, not a GNU license. Under a BSD license the standard can be used and extended without having to return those proprietary extensions to the open source community, enabling companies to leverage the foundations of RISC-V while still protecting their differentiation through special instructions.

Reigniting innovation in semiconductors

Tech industry giants such as Micosoft, HP, and Google have joined the RISC-V Foundation, alongside startups like Codasip and SiFive that are already bringing RISC-V-based solutions to market. RISC-V has been upstreamed by major open source projects including the Free Software Foundation, BeanUtils, the GCC, and Fedora, all of which indicates not only a growing amount of interest in RISC-V as an accessible technology, but also one with real commercial viability. But it's just another ISA, right?

"The whole vision is how to drive down costs, everywhere actually, starting with the ISA," Speers explains. "I view RISC-V as the kernel for reshaping how design is done in semiconductors, and starting to dramatically lower the cost of producing new chips. That's the end goal in mind. It's not that the RISC-V Foundation can proudly say that they have this ISA, it's that they want to dramatically lower costs – reignite innovation in semiconductors."

To learn more about RISC-V, the RISC-V Foundation, or upcoming workshops, visit riscv.org. ECD

References:

1. "How Many X86-64 Instructions Are There Anyway?" Accessed November 18, 2016. http://www.tuicool.com/articles/JJzAj23.
2. Shimpi, Anand Lal. "The ARM Diaries, Part 1: How ARM's Business Model Works." RSS. Accessed November 18, 2016. http://www.anandtech.com/show/7112/the-arm-diaries-part-1-how-arms-business-model-works/2.

SECURITY AND THE CORTEX-M MPU

By Ralph Moore

As embedded systems are drawn more into the Internet of Things (IoT), security in the form of protection of critical resources is becoming increasingly important. Effective protection can only be achieved via hardware means.

The Cortex-M Memory Protection Unit (MPU) is difficult to use, but it is the main means of hardware memory protection available for most Cortex-M processors. These processors are in widespread use in small-to-medium-size embedded systems. Hence, it behooves us to learn to use this MPU effectively in order to achieve the reliability, security, and safety that modern embedded systems require.

Developing a good protection strategy for an embedded system is difficult enough without having to deal with excessive low-level complexity. The latter is not only frustrating, but may result in insufficient system protection. MPU software is needed that overcomes hardware complexity and provides a solid foundation for creating protected, secure systems, and for detecting and dealing with security violations. This is the first in a series of entries presenting a software approach to satisfy this need.

Terminology

Cortex-M processors have three modes of operation:

- **Handler mode** – Privileged mode for ISRs, fault handlers, the SVC Handler, and the PendSV handler. This mode can be entered only via an exception.
- **Privileged thread mode** – Privileged tasks (ptasks) run in this mode. It can be entered only from handler mode.
- **Unprivileged thread mode** – Unprivileged tasks (utasks) run in this mode. It can be entered from either of the above two modes.

In the discussions that follow, the first two modes are collectively called pmode and the third mode is called umode. Similarly, this and subsequent discussions refer to pcode, ucode, pSSRs, uSSRs, etc. These are not industry-standard terms, but rather are introduced here to simplify discussions.

Protection goals

The basic goal of protection is to run trusted software in pmode and to run all less-trusted software in umode. Examples of trusted code are real-time operating systems

"RETROFITTING LEGACY CODE WITH MPU PROTECTION IS POSSIBLE AND IS A PRIMARY OBJECTIVE OF THE MPU SOFTWARE THAT WE WILL BE DISCUSSING."

(RTOSs), instruction set randomizers (ISRs), handlers, and low-level drivers. Examples of less-trusted code are untested code, third-party software, or software of an unknown pedigree (SOUP), and code that is vulnerable to malware such as protocol stacks and high-level drivers. This goal can be broken into subgoals for umode:

- Prevent direct access to RTOS services and data
- Prevent access to restricted RTOS services by utasks
- Protect processor core resources (e.g. SysTick timer)
- Protect peripherals from unintended modification
- Prevent code execution from RAM
- Prevent direct access to critical system code and data
- Restrict utasks and utask groups access to only designated code and data
- Permit isolating utasks and utask groups from each other
- Detect task stack and buffer overflows immediately
- Detect intrusions and bugs and shut them down so critical operations are not imperiled

Obviously, the degree of protection needed depends upon the security and safety requirements of the specific system. Note that protection for a system can be increased in future releases as it becomes more widely distributed and therefore more vulnerable. This MPU software approach fosters progressive protection improvement in this manner.

MPU basics

Cortex-M0/1/3/4 MPUs have eight slots and Cortex-M7 MPUs have 16 slots. Each active slot defines a memory region with its own attributes, such as size, alignment, read/write (RW), read only (RO), execute never (XN), etc. Slots in which the XN bit is zero are inactive and have no effect upon memory accesses. Hence a user is not forced to use all slots. Unused slots are usually filled with zeros to disable them.

Two unfortunate aspects of the Cortex-M MPU are that memory region sizes must be powers of two, ranging from 32 bytes to 4 GB, and memory regions must start on multiples of their sizes. These requirements undermine the utility of the MPU by making it difficult to use without wasting substantial memory. For example, if a protected stack increases from 256 bytes to 260 bytes, the region containing it must be increased from 256 bytes to 512 bytes, and if it is not already on a 512-byte boundary, it must be moved up 256 bytes to the next 512-byte boundary – almost a 200 percent waste of memory. This is a problem because systems using Cortex-M processors usually have limited memory.

An advantage of the MPU is that it allows definition of very small regions (as little as 32 bytes). This compares to a minimum of 4096 bytes for most memory management units (MMUs). Hence an MPU is more appropriate for RTOS-based multitasking systems than is an MMU.

Following initialization, the MPU is in background mode and all of its slots are disabled (i.e. all are filled with zeros). In this mode, operation is the same as with the MPU off or no MPU at all. System initialization is performed in this mode. Then MPU slots are loaded with regions.

Converting legacy code

Hacks and malware attacks are becoming increasingly prevalent. As a consequence, many managers probably wish that products their companies are shipping today had better protection. Retrofitting legacy code with MPU protection is possible and is a primary objective of the MPU software that we will be discussing.

Legacy code will run normally in pmode with the MPU enabled in background mode. This is the starting point. From here, less-trusted tasks and code are gradually moved to umode. This step-by-step process allows dealing with the least-trusted and most-vulnerable code first, while making sure that the system continues to run correctly after each step. If it does not, the step can be reversed and the problem(s) found and fixed. This permits a strategy of security updates to make installed systems more secure as their numbers increase.

Another good time to add MPU protection is when new features are added to legacy systems that have been in use for some time. In this case, the new features are probably ancillary to the main function of the equipment and thus can and should be isolated. This would be particularly true if networking were added to what had been a stand-alone system, for example.

Trusted code and trusted tasks are best left running in pmode because it is simpler and faster. pcode and pdata are in the background region and thus accessible by all ptasks and handlers. This assures that carefully crafted mission-critical code need not be rewritten – it stays the same and runs the same. Furthermore, communication stays the same whether between ptasks or between ptasks and utasks. However, utasks are isolated and may not perform restricted system services, such as power down or deleting other tasks.

Developing new code

Security adds a new dimension to product development. While theoretically sound to "build security in from the start," it may not be an overly welcome added dimension to projects that already have too many dimensions and too little time to achieve them. Thus, there may be a need to postpone security measures until late in the project, or even after the project, when they become more beneficial and less of a distraction.

Drawing protection boundaries before tasks and code have stabilized can waste significant time.

Background mode may be the best starting point for developing new code for new projects since debugging is simpler. However, once code is operating reasonably well, moving it into umode enables the power of the MPU and supervisor call (SVC) to assist in debugging problems such as stack and buffer overflows, wild pointers, restricted operations, etc. This is most helpful during the system-integration phase. Tasks can be moved from pmode to umode and back in order to track down and fix problems caused by umode.

The benefits of MPU error detection during the final project phase can easily outweigh the time required to modify code and tasks to run in umode. However, some projects may prefer to suffer. If so, post-release system security upgrades are possible. Of course this breaks all the rules, but it makes practical sense for projects that are behind schedule and overwhelmed by just getting required features to work.

Once the dust settles, it is possible to step back, look at the system security requirements, and start making the system more secure. During this time, manufacturing and installation problems are being slowly solved by other people, shipments are gradually increasing, and likewise, security can be gradually improving.

Upcoming entries on embedded-computing.com:

- Multitasking and the MPU
- Defining MPU regions
- Software interrupt API for MPU systems
- Structuring MPU applications
- More information can also be found at www.smxrtos.com/mpu.

***Ralph Moore,** President and Founder of Micro Digital, graduated with a degree in Physics from Caltech. He spent his early career in computer research, then moved into mainframe design and consulting.*

ASYNCHRONOUS EVENT-DRIVEN ARCHITECTURE FOR IoT'S EDGE

Dr. Miro Samek, President, Quantum Leaps

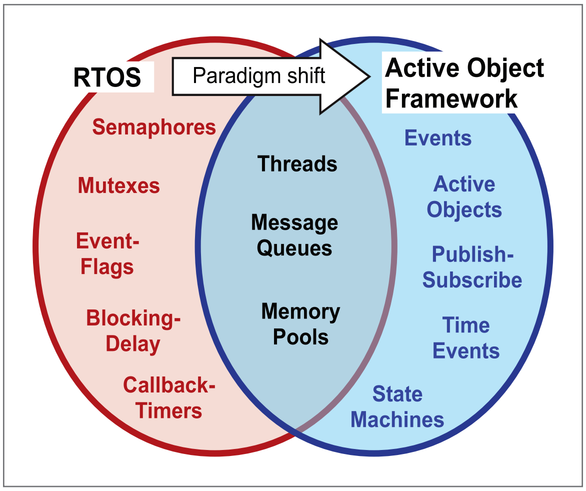
The embedded software industry is in the midst of a major revolution. Tremendous amounts of new development lays ahead. This new deeply embedded software needs a truly reusable *architecture* that is inherently safer, easier to understand, and provides a higher level of abstraction than the usual "free-threading" approach based on a traditional real-time operating system (RTOS).

For years, experts in concurrent software have been pointing out that unrestricted use of threads and various blocking mechanisms of an RTOS often leads to programs that are unsafe and difficult to reason about. Instead, experts from different industries independently came up with the following best practices, which are collectively know as the *active object (or actor) design pattern:*

1. Keep data isolated and bound to threads. Threads should hide *(encapsulate)* their private data and other resources, and not share them with the rest of the system.
2. Communicate among threads asynchronously via event objects. Using asynchronous events keeps the threads running truly independently, *without blocking* each other.
3. Threads should spend their lifetime responding to incoming events, so their mainline should consist of an event loop that handles events one at a time (to completion), thus avoiding any concurrency hazards within a thread itself.

While these best practices can be applied manually on top of a traditional RTOS, a better way is to use an *active object framework*. The main difference is that when you use an RTOS, you write the main body of the application (such as the thread routines for all your tasks) and you call the RTOS services (e.g., a semaphore, or a time delay). When you use a framework, you reuse the overall architecture and write the code that *it* calls. This leads to *inversion of control,* which allows the framework to automatically enforce the best practices of concurrent programming. In contrast, a raw RTOS lets you do anything and offers no help or automation best practices.

The other important difference is that the event-driven active object framework really represents a *paradigm shift* from a traditional RTOS, as illustrated in the following Venn diagram:



In resource-constrained embedded systems, the biggest concern has always been about the size and efficiency of such active object frameworks, especially since the frameworks accompanying various modeling tools have traditionally been built on top of a conventional RTOS, which adds memory footprint and CPU overhead to the final solution. However, our experience at Quantum Leaps shows that frameworks of this type can be actually **smaller** than a traditional RTOS, because active objects don't need to block internally, so most blocking mechanisms of a traditional RTOS aren't needed (or desired) for programming event-driven active objects.

But perhaps the most important benefit of active object frameworks is that they offer a much *higher level of abstraction,* and the *right* abstractions for applying formal design techniques such as hierarchical state machines (UML statecharts), modeling, and automatic code generation.

All of this means the event-driven architecture is not only possible in deeply embedded systems, such as Internet of Things (IoT) edge nodes, but it is actually ideal for such applications.

RTOS IN THE IoT: WHERE WE ARE AND WHERE WE'RE HEADED

By Brandon Lewis, Technology Editor

The Internet of Things (IoT) has brought software developers from all walks of technology into the field of embedded system design, and with them various predispositions concerning the type of operating system (OS) best-suited for device development. General-purpose operating systems (GPOS), bare metal design, and the "free one" all have their place based on the requirements of the system, but increasingly commercial-grade real-time operating systems (RTOSs) are being deployed for their determinism, flexibility, portability, scalability, and support. In this roundtable, Bill Lamie, President and CEO of Express Logic, Andrew Caples, Senior Product Line Manager at Mentor Graphics, and Dinyar Dastoor, Vice President and General Manager at Wind River discuss the current and future value propositions of using an RTOS in the IoT.

How is the advent of the IoT impacting both RTOS development and developing with an RTOS?

CAPLES: Certainly, the landscape of RTOS is changing. In the '90s it was all about networking – slapping a TCP/IP stack on every device possible, and it was very exciting when you saw the first Ethernet controllers built into the 68332 microcontrollers (MCUs) way back then. Now it's all about connectivity, whether it's 802.11, Bluetooth, Bluetooth Low Energy (BLE) – any way to connect devices.

You're now starting to see the introduction of platforms specifically geared towards IoT – mbed, Zephyr, etc. They offer a lot of connectivity and IoT protocols, and that's what's needed essentially when it comes to device design. It's all about connectivity today.

This changes the philosophy of the developer, for sure. There's less of an emphasis on doing RTOS development. It's all about the platform and the ease of use. When it comes to hardening the platform, customers are expecting the platform to already be hardened and the connectivity to already be there. They're expecting out-of-the-box solutions that will work on their platforms of choice and have the IoT connectivity selections that they need. Essentially, the developer is looking to focus on their use case. The assumption is that the RTOS and the connectivity just work.

DASTOOR: One of the key changes that we see from an RTOS perspective is that RTOS initially were single-function devices. You'd program them to do one thing and then they would do that thing extremely well in a very deterministic fashion. Then, they became more sophisticated, so they could do several things – again, in a very deterministic fashion.

Now that same RTOS is also supposed to do all kinds of communications, and with communication also comes the headache of managing communication, storing, logging, and the like – and all of these need not be deterministic, but the overall device should be fairly deterministic. This is really showing up in the industrial/medical world, and creating interesting paradigms because you have very controlled real-time determinism on the operational side so that if you have a machine doing something with a patient, for example, it retains its safety, its security, and all of that. At the same time you also want to extract data from the device and put it in the cloud.

Technology has evolved to address some of these challenges, but as an RTOS developer you have to keep these issues in mind because when you're designing a device you're constantly thinking, "How will I communicate with the outside world in a secure fashion?"

BILL LAMIE
President and CEO of Express Logic

EXPRESS LOGIC
www.rtos.com

LINKEDIN
www.linkedin.com/company/express-logic

ANDREW CAPLES
Senior Product Line Manager at Mentor Graphics

MENTOR GRAPHICS
www.mentor.com/embedded-software

TWITTER
@MentorEmbedded

LINKEDIN
www.linkedin.com/company/express-/mentor-embedded

DINYAR DASTOOR
Vice President and General Manager at Wind River

WIND RIVER, AN INTEL COMPANY
www.windriver.com

TWITTER
@WindRiver

LINKEDIN
www.linkedin.com/company/wind-river

How do you see RTOSs being used in lieu of, in conjunction with, or not at all given other choices available to IoT developers, such as GPOSs, open source options, or bare metal development?

CAPLES: It's all about the requirements of the system. Based on the requirements, bare metal might be completely satisfactory for the design. In other situations, something like a FreeRTOS may be well suited – based on the limited functions and features that it brings, it still may be enough to meet the system requirements.

Certainly an RTOS, by definition, "a real-time operating system," is going to continue to have a significant role. You do see GPOS out there that provide a lot of connectivity and find their way into embedded systems, but those systems they find their way into do not have real-time requirements, and they may lack other types of requirements where an RTOS is very well suited.

When it comes to devices that are more feature rich or have requirements such as the utilization of space-domain partitioning, power management, connectivity, dual networking stacks, file systems, execute in place (XiP), security, and so forth, these historically become more problematic or challenging for some of the free RTOSs out there to be able to introduce. These systems truly require not only the feature set of a commercial offering, but also the expertise of the people that support it in order to be able to deploy the features in a meaningful way to meet the system requirements of the project.

Especially on the end nodes, sensors, and actuators, we do see real-time requirements with low latency, fast interrupt context switching, small footprints, fast boot-up times, and maybe even features like XiP where there are limited system resources. These are features that are better suited for an RTOS than a GPOS or even bare metal.

LAMIE: Development with a commercial RTOS versus in-house, bare-metal development can be beneficial in several ways. First, an RTOS is easier to develop with and maintain, as threads (tasks) can be developed to handle smaller sets of functionality, spread out over a team. Second, it is easier to add middleware with a commercial RTOS as a product becomes more connected or complex due to increased functionality. Third, a commercial RTOS is more portable because it eliminates the need for processor-specific code that otherwise would have to be part of the application. Fourth, most commercial RTOSs provide widely available technical support.

An RTOS provides developers the flexibility to add new features to a given device, as the RTOS manages the processor allocation logic such that real-time performance of a high-priority thread can easily be guaranteed, regardless of whether the firmware is 32 kB or 1 MB in size and regardless of the number of threads in the application. This alone makes it easier to maintain the application and easier to add new features to a device. In addition, most commercial RTOS offerings have an extensive set of middleware that is pre-integrated and ready to be deployed. This enables developers to easily add networking, file systems, USB, and graphical user interfaces (GUIs). A commercial RTOS should have a small footprint, something in the area of 2 kB of instruction area memory (usually flash) and 1 kB of RAM, which means there is more memory available for supporting connectivity and GUI needs.

Applications that use an RTOS are more portable because service functions are accessed through an application programming interface (API). The API makes the RTOS platform-independent, meaning that it's the same regardless of the processor it runs on. This makes switching processors easier since none of the application's service references need to be changed. The application will run anywhere the RTOS can run, and with most popular commercial and open-source RTOSs, that means virtually any 32-bit processor architecture. This gives developers application portability with only minimal changes to their code.

Responsiveness is another area where an RTOS works well for IoT devices. This is because an RTOS invisibly handles allocation of the processor to the threads that perform the various duties of the embedded device. With the proper assignment of thread priorities, application software does not have to concern itself with the processing time required for each thread. Even better, changes in the processing responsibilities of a given thread do not impact the processor allocation (or responsiveness) of higher priority threads. The key is to ensure that your RTOS is priority-based and supports fully deterministic processing.

In addition, a commercial RTOS is more likely to be widely used, thus "field proven," with fewer bugs likely to "pop up." Commercial RTOS vendors can invest in improvements of the RTOS in an organized and customer-focused fashion.

DASTOOR: As far as RTOS being used alongside a GPOS, we absolutely see the two being used in together, and there are two ways this would happen. One is that, with multicore, you have more cores at the same price point than you can put to use. There are many devices today that have four cores, but only one core is used. With multicore, it opens up this possibility that, on the same processor, I can run two heterogeneous applications or heterogeneous OSs doing two different things altogether (Figure 1). That's one classic case of virtualization where you are running VMs, and one example where that is popping up is in gateways.

Gateways are a classic device where you want to protect something that is either deterministic or real time on the operational technology side of it. On the other side, you're connecting to the cloud, which is the IT domain. With the IT domain, of course you are looking at more security, bandwidth, cost of transportation, etc., so a hard separation comes along when trying to put two heterogeneous OSs or two heterogeneous use cases on the same box.

The other concept that comes up is extensibility of devices, which means that you install a device today but you don't know what you'll want to do with this device two years from now because technology changes so fast. So containerization is another big trend that we see in the IoT. We know the advantages of containerization – isolation, it provides you a nice sandbox to write applications, etc. – but we see that, more importantly, looking to the future, if I want to do something with the device, I can do it. I can update the whole firmware. Therefore, you would rather have a framework in your device that allows you to deploy applications remotely and run them. Containerization also allows you now to separate soft real-time from hard real-time from no real-time requirements, and still have the applications mish-mashed together in the same system.

Given the microkernel architecture that many RTOSs employ, as well as the need for portability and scale in IoT devices, is it recommended to begin development with an RTOS so there is headroom for the future?

CAPLES: RTOSs today bring a lot of connectivity capability and IoT protocol support, whether it's CoAP or MQTT or XMPP. Those IoT protocols are routinely supported by leading RTOSs, and much of this stuff is functional in resource-constrained devices that you would find MCUs in. It includes even the crypto you would expect to see, like OpenSSL equivalents from wolfSSL that integrate with our Nucleus RTOS in a small-footprint manner so it can be utilized on memory-constrained devices.

But just because it's on memory-constrained devices doesn't mean these platforms can't transcend up the food chain into Cortex-A devices. What you're finding on the higher end is the multicore, heterogeneous cores, and heterogeneous OSs, and if you already have that type of capability built into a small MCU, many of today's heterogeneous system on chips (SoCs) already include a Cortex-M-type core that perhaps you use as the communications core sitting next to the applications processor. Certainly that's going to give great advantage to the RTOS.

DASTOOR: The decision is what to start with. Should you start with an RTOS and then extend or do you start with a GPOS and run an RTOS as a virtual machine (VM) within that GPOS? Both have their place.

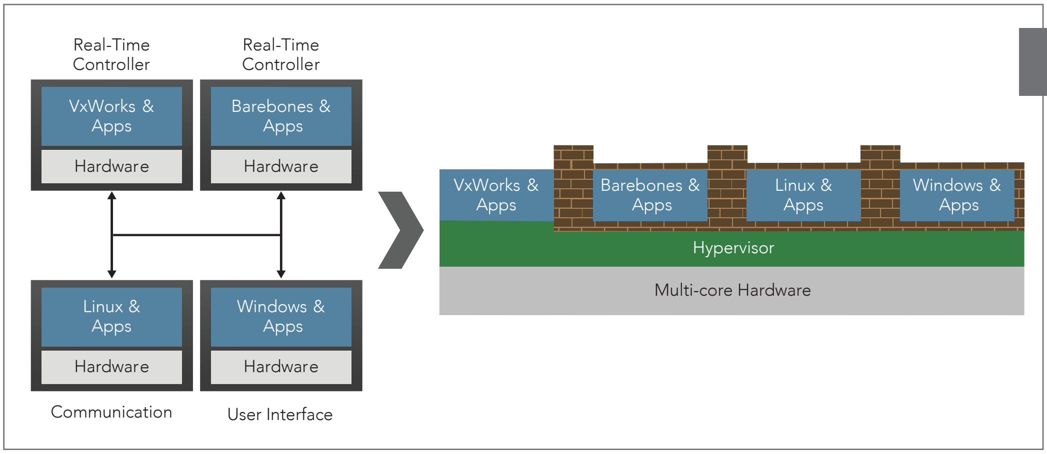


FIGURE 1

The ubiquity of multicore processors, paired with virtualization technology, enables multiple heterogeneous operating systems (OS) to be deployed in the same system, with gateways being a prime example. Shown here are heterogeneous OSs containerized using Wind River's Virtualization Profile for VxWorks.

"… THE INCREASED CONNECTIVITY REQUIREMENTS ALONE NECESSITATE THE EXECUTION OF COMMUNICATION PROTOCOL STACKS ON 32-BIT EMBEDDED MICROPROCESSORS. THAT, IN TURN, NECESSITATES THE USE OF AN RTOS. GUI DESIGN AND RUNTIME SOFTWARE FROM THIRD PARTIES TYPICALLY RELY ON RTOS SERVICES AS WELL. ALL THIS MAKES IT INCREASINGLY LIKELY THAT IOT DEVICE DEVELOPERS WILL CONSIDER USING AN RTOS."

One very interesting thing now is that, as you containerize, you're almost creating a fragment of functionality that is running on the device. Now, what prevents you from running that fragment of the device somewhere else or in another server? Maybe still in the same geographic location, but on another system?

This is where the concept of software-defined automation or the software-defined factory starts bubbling up. As a good example, if you have a programmable logic controller (PLC) controlling a machine, it also does intelligent analytics of whatever it is trying to control. If you start containerizing the analytics portion of it, you can run it on a server in the data center. Then, once you virtualize something, you are into the familiar world of IT virtualization. So software-defined everything (SDX) is showing up in many places.

What else do you see happening in the world of RTOS in the next 5-10 years?

DASTOOR: The "things" that we talk about today – we call them devices – those things will have feelings like humans have feelings, which means the things will have a need to be identified. You put them in a factory and when they power up they should immediately figure out, "Where am I? Who do I belong to? What language should I speak? And who do I trust in this world?" Also, once you have that identification and connectivity, the next element of feeling is, "Can I learn something on my own?" Machine learning becomes a big thing for those devices; self-healing becomes another big thing for those devices.

It's a very complex problem to solve because you may have 1 billion devices waking up one day needing to be identified and find their place in the world. Those kinds of features will become very fundamental to the OS or the RTOS itself.

LAMIE: A study published in Business Insider tells us there will be 25 billion IoT devices shipping by the year 2019. Because such devices will require network connectivity (for example, Wi-Fi, BLE, ZigBee, Ethernet, etc.) and will often include a GUI, they also will require 32-bit microprocessors to provide the necessary address space and processing power. We are already seeing strong migration from 8- and 16-bit microprocessors (MPUs) due to enhanced functionality, as well as the cost versus performance attributes of new 32-bit MPUs.

The predicted IoT explosion promises to sharply accelerate this migration. Moreover, the increased connectivity requirements alone necessitate the execution of communication protocol stacks on 32-bit embedded MPUs. That, in turn, necessitates the use of an RTOS. GUI design and runtime software from third parties typically rely on RTOS services as well. All this makes it increasingly likely that IoT device developers will consider using an RTOS. **ECD**

RTOS IN THE IoT: COMBATTING COMPLEXITY WITH CONTROL

By Johan Kraft

In the near future, almost everything will be connected to the Internet – at least, that's the vision for the Internet of Things (IoT). This vision belies the complexity behind it, because most of the communication will take place between the "things" in the IoT rather than people.

That's relevant because people are really rather good at managing complexity. User interfaces have evolved to support this: keyboards have been such a successful common denominator for people and computers that they remain the default interface, while pointing devices and touch screens work intuitively from the perspective of the operator. Things – even intelligent things – may not be quite so good at managing the complex task of interfacing with other technology, except under strictly controlled conditions.

Invariably those strict conditions will translate into protocols – an approach that only works when all devices understand the protocol being used. The IoT will make use of many different kinds of protocols at the edge, particularly lightweight protocols that are suited to small, resource-limited devices. The job of getting data out of these edge devices and on to the "super highway" will fall to gateways, which are devices intended to manage local-area networks (LANs) of edge nodes including (but not be limited to) intelligent sensors and actuators.

Already the potential for complexity becomes apparent. Gateways may have many hundreds of nodes to manage locally using multiple protocols, while also being part of a wide-area network (WAN). What makes this task even more challenging will be the nature of those nodes. Although imbued with some level of intelligence, it is unlikely that they will be aware of the LAN, and definitely not the WAN that is the Internet; they may well see the world simply as them and the gateway. These 'dumb intelligent' sensors will need to communicate reliably with the gateway, not least because the gateway will have limited resources with which to manage all nodes.

Many nodes will be expected to operate for many hundreds of hours (if not years) on a single battery, perhaps even supplementing their power with harvested energy. To preserve energy, many of those nodes will spend most of their time in a deep sleep mode, waking periodically to check their own status, take measurements, and ultimately exchange data with the gateway.

"PUTTING AN RTOS INTO AN IOT EDGE NODE WILL ALLOW THAT NODE TO MORE ACCURATELY CONTROL ITSELF …"

Coordinating hundreds of resource-limited edge nodes on a network this way will require accurate timing. Even with extended periods of inactivity, this mode of operation describes a typical use case for embedded real-time operating systems (RTOSs).

An embedded RTOS today is less "software overhead" and more "essential component." Even smaller and cheaper microcontrollers are able to run an RTOS, some of which take considerably less code space than engineers may expect. The system management features of a modern embedded RTOS can remove a lot of the complexity involved with developing an embedded device. A modern RTOS on an embedded microcontroller (MCU) would typically be able to manage the device's advanced power saving modes much more efficiently than code written by the developer. This efficiency extends to implementing communication protocols. Many of those being used in the IoT may already be provided as part of the standard middleware from RTOS suppliers, or available through their ecosystem partners.

Putting an RTOS into an IoT edge node will allow that node to more accurately control itself: reacting to network commands, periodically waking to check its status and collect data, or launch functions triggered by external events. Perhaps more importantly, it will be able to communicate with the gateway in a more reliable way, thereby maintaining a high level of network efficiency that will allow the gateway to better manage a large number of edge nodes.

The IoT is going to require a level of intelligence at the edge node that supports its inherent complexity. That intelligence will be best provided by using a small, efficient RTOS running on an advanced embedded MCU, supported by embedded software development tools that can make sure realizing the IoT is achievable, however large it gets.

Johan Kraft *is founder and CEO of Percepio. Based in Sweden, Johan built the company up after his early days as an embedded software developer and even earlier academic work focusing on practical methods for timing analysis of embedded software.*

Percepio
www.percepio.com

TWITTER
@percepio

LINKEDIN
www.linkedin.com/company-beta/341077

FACEBOOK
www.facebook.com/percepio

GOOGLE PLUS
https://plus.google.com/113857908701745133692

IMPROVING SMARTPHONE CAMERAS WITH COLOR SENSOR TECHNOLOGY

By Darrell Benke

The world of digital image capture has evolved dramatically in the past decade. Perhaps most striking and remarkable are the advances in smartphone camera technology. The impact of smartphone cameras has resulted in a significant reduction in standalone digital camera sales over the last several years, as consumers are increasingly satisfied with the features and image quality of their mobile phone cameras.

The quality and accuracy of image capture has become a key area of innovation and differentiation, and OEMs are continually improving the capabilities of smartphone cameras. Ultimately, smartphone OEMs continue to strive to deliver a digital single-lens reflex (DLSR) camera experience on phones while avoiding a heavy camera body/lens and higher cost. Consumers enjoy the benefits of capturing pictures with improved simplicity, resolution, and more life-like color accuracy.

LG raises bar with smartphone color spectrum technology

A great example in the market today is LG, which has a long-standing reputation for delivering advanced camera technology in its smartphones and has received many awards from various photography and imaging organizations. Through close collaboration with LG, ams delivered color sensor technology enabling LG's G5 flagship phone to deliver a color spectrum sensor (CSS) technology for its primary camera. The phone's color sensor is located next to the flash LED on the backside of the phone (Figure 1).

LG is one of the first smartphone OEMs to take advantage of a color sensor to enhance its camera capabilities. The new sensor enables measurement of the ambient light while also determining if the light source is artificial or natural. In addition, it provides the ability to intelligently distinguish if the light source is from the ambient environment or an object within the field of view. By understanding the exact lighting conditions with such precision, the phone can select the optimal white point.

How color sensing technology works

The critical capability of a color sensor is the precise measurement of correlated color temperature (CCT) as well as the infrared (IR) component of the ambient light. CCT is a metric that defines the color appearance of a light source by relating its color to a defined reference. IR is the portion of the electromagnetic spectrum with wavelengths beyond visible light and in the 700 nm to 1 mm range. Light CCT ranges from cool colors (bluish white) to warm colors (yellowish and red).

With accurate CCT and IR measurements, the lighting source can be identified as natural or artificial (i.e. sunlight vs. LED, incandescent, and fluorescent) and used to set the optimal white point for the image capture. The spectral components of various light sources have wide variations in the visible red, green, and blue (RGB) and invisible (IR) spectrum, and therefore accurate sensing of both components is needed to distinguish each light source.

Take, for example, the spectral responseness of the ams TCS3400 color sensor (Figure 2). The on-chip IR-blocking filter minimizes the RGB responses to IR light required for more precise color measurement.

The response from the RGB channels can be used to determine the lighting environment CCT. In addition to RGB and IR sensing, the sensor also has a Clear (C) channel, which provides a reference channel for isolation of the color measurement. The four RGBC channels each have a dedicated 16-bit data converter, allowing simultaneous measurements.

Why use discrete color sensors?

Using a discrete color sensor in conjunction with an image sensor allows for measurement of IR light. Typical smartphone image sensors block this portion of the light spectrum since it is not in the visible range and, more importantly, it can have adverse effects on image capture quality. Therefore, smartphone cameras' image sensors typically implement an IR blocking filter to ensure that the camera does not sense IR. In some situations, IR light can pass through these blocking filters and result in unnatural color and other adverse effects on the image quality. The camera's image processing algorithms can eliminate these IR-induced effects if they have a measure of the ratio of IR light to visible light in the ambient lighting.

Another challenge for cameras is the ability to distinguish between color reflected from an object and the color of the ambient light when capturing an image. The combined RGB and IR sensing capability of the color sensor allows a camera subsystem to automatically make this determination and subsequently set the optimal white point for the image capture. The accurate IR light measurements allow the best capture

FIGURE 1 Color sensor technology raised the bar for camera innovation and performance improvement.

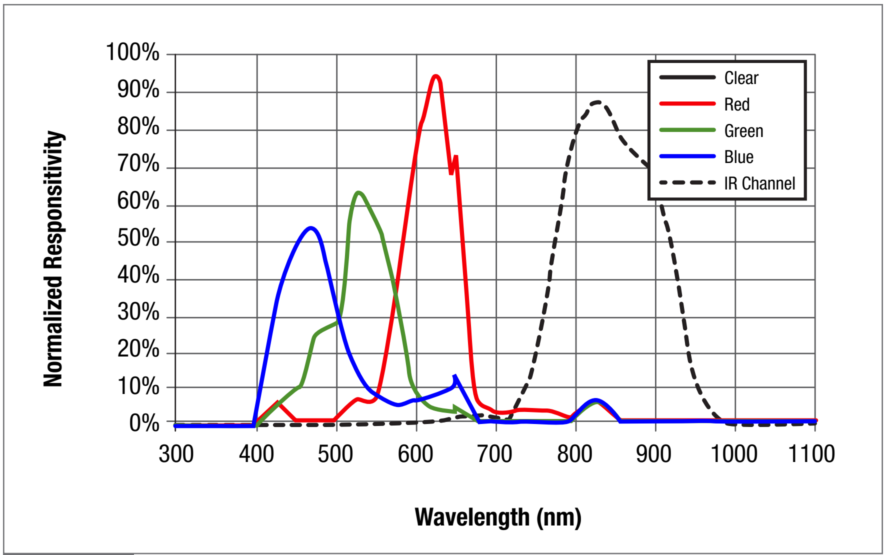


FIGURE 2 Spectral responsivity of the ams TCS3400

of the actual color of the object rather than a misinterpreted color due to the IR light in the surrounding environment. Selecting the optimal white point is critical for the most natural and balanced appearance of a picture.

Additional color sensor applications

Color sensing technology along with the ability to accurately measure the ambient light CCT and intensity enables other smartphone applications that enhance the user experience. Among these applications are automatic camera LED flash color control and automatic display white point setting.

Smartphone camera flash implementations may use multiple LEDs, white and non-white in color, to intelligently fire the LEDs with the right color and amount of light based on the CCT of the lighting environment. Automatic determination of the optimum flash intensity and color, based on the scene's lighting conditions, provides the ability to make both indoor and outdoor images more balanced.

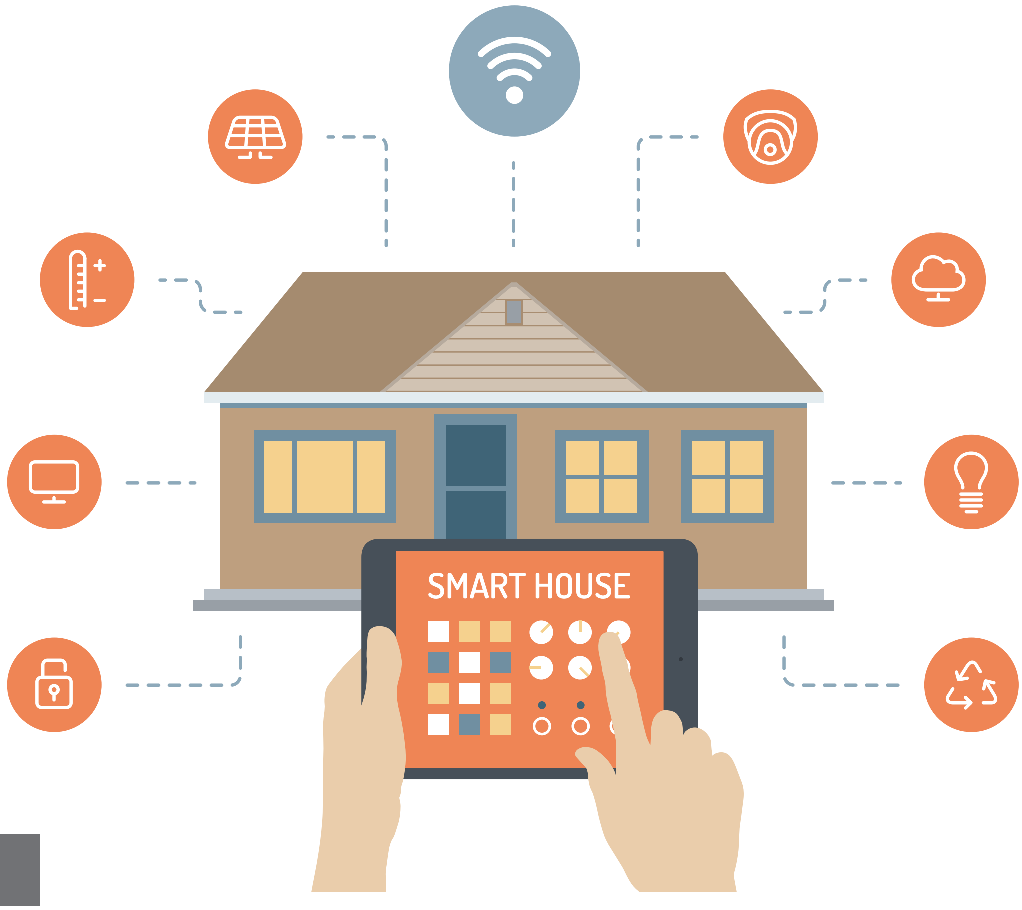
> **"VISION SCIENCE RESEARCH HAS CONCLUDED THAT OUR VIEWING EXPERIENCE IS BEST WHEN THE DISPLAY WHITE POINT IS ADJUSTED BASED ON THE CCT OF THE LIGHTING ENVIRONMENT."**

Another feature enabled by color sensing is automatic adjustment of the display's white point to match the lighting environment's CCT. The display white point, sometimes referred to as reference white, is typically a fixed setting where the displayed "white color" appears "white" to our eyes. Vision science research has concluded that our viewing experience is best when the display white point is adjusted based on the CCT of the lighting environment. Therefore, a fixed setting is not ideal for smartphones that are used in a broad range of lighting conditions. The ability to dynamically adjust the display white point based on the lighting environment allows images to look more appealing and colors appear accurate for a better user experience. **ECD**

***Darrell Benke** is a Strategic Program Director for the Advanced Optical Solutions Division of ams AG with a focus on smartphone solutions. He has over 20 years of experience in business development, strategic planning, marketing and semiconductor/system design roles for emerging technologies for semiconductor companies including Texas Instruments, National Semiconductor, Micron Technology, and Rockwell International. He has M.S. in Electrical Engineering and VLSI Design from the University of Texas at Dallas and a B.S. in Electrical Engineering from North Dakota State University.*

BLUETOOTH LOW ENERGY MESH NETWORKS CAN ENABLE SMART HOMES

By Dr. Wenjun Sheng



The Internet of Things (IoT) is all about connecting devices in a way that allows data to be collected and acted upon, with some level of intelligence. The smart home is all about connecting everyday devices in the home into a network that can then be easily monitored and managed. Hence, the IoT is a fundamental part of the smart home network, connecting devices, monitoring appliances and home data, and taking some action to make the user more comfortable, provide convenience, or save costs.

To connect everyday devices into a network there are several different communications technologies, including Bluetooth, ZigBee, 6LoWPAN/Thread, and Apple's HomeKit. Of these, Bluetooth Smart/Bluetooth Low Energy (BLE) mesh networking is increasingly being considered as an alternative or complement to other established home network-ing standards.

ZigBee has been around for over 10 years in low-power wireless networking applications, while Bluetooth has traditionally only been used for short-range, low-power device-to-device communications. Bluetooth historically hasn't provided wider network connectivity, but that's changing as not long ago the Bluetooth Special Interest Group (SIG) announced plans for a standardized mesh networking architecture capability with BLE technology.

BLE mesh networking still needs to be standardized, but it's already being deployed in some applications, like GE's connected lighting products. By utilizing this new mesh networking capability, individual Bluetooth devices can connect to other devices in networks, enabling coverage of an entire building or home, which makes the technology an ideal fit for home and building automation applications.

What makes the smart home network even more possible with the emergence of BLE mesh networks is that Bluetooth is already embedded into many consumer electronics products, particularly smartphones, tablets, and laptops. So not only can Bluetooth sensors be deployed widely in the home, but the management and control of devices is also made easier since an app or software can easily be developed for smartphones or laptops. This opens up a whole new opportunity for Bluetooth in

"… NOT ONLY CAN BLUETOOTH SENSORS BE DEPLOYED WIDELY IN THE HOME, BUT THE MANAGEMENT AND CONTROL OF DEVICES IS ALSO MADE EASIER SINCE AN APP OR SOFTWARE CAN EASILY BE DEVELOPED …"

smart locks; lights; heating, ventilation, and air conditioning (HVAC) systems; and many other appliances, which can form ad hoc networks to deliver a seamless smart home experience.

From a system control point of view, ubiquitous BLE support in mobile devices means an extra remote control isn't necessary and users can experience things like smart lighting immediately using a mobile app and connected light bulbs. System implementation is also simplified since no gateway or router is needed if long-distance, offsite control is not required.

The removal of a gateway also avoids degradation of the user experience caused by indirect user interactions, and also possible complexity challenges for ordinary users that involve installing and setting up the gateway. This saves considerable cost and also eases the deployment process.

In terms of implementing this capability at a chip level, a highly integrated system-on-chip (SoC) like the TLSR8263 from Telink Semiconductor means fewer off-chip components and a low bill of materials (BOM) cost. Also, BLE mesh firmware requires less SRAM and program memory than some other technologies. All of these factors result in significant overall BOM savings for smart device networks adopting BLE mesh.

The high level of integration and small firmware requirements help drive down the cost of BLE mesh-based solutions significantly for end users as well. For example, the C by GE light bulb sets based on a BLE mesh (four LED bulbs with a free iPhone app) cost around $50, which is significantly lower compared to other available smart lighting solutions (Figure 1a and 1b).

The low price threshold of products such as the C by GE light bulbs is significant in creating user adoption. In addition, in the connected light bulb application, Telink's BLE mesh technology has been able to demonstrate support for a large number of nodes even without a gateway or hub. Companies deploying such solutions in industrial or building automation applications have been able to simply use a smartphone app or a remote control to reliably control over 1,000 smart LED bulbs at the same time. As BLE technology is mature and low cost, BLE mesh networks are suitable for many IoT applications, regardless of industry.

However, some challenges still remain for BLE mesh. Currently, the technology is only available on the market from a few vendors, and among them even fewer provide stable solutions at the commercial level. Since it's new, some manufacturers are still skeptical and being cautious in picking up the technology, with its lack of interoperability with other standards and other devices already in production being a major concern.

Many components still must be added into the BLE mesh mix, such as more advanced cloud integration/interaction and the option for remotely accessible gateways. What's clear, though, is that BLE mesh network technology will continue to evolve and emerge, and become a more integral part of smart home networks as it becomes standardized, deployments increase, and costs are driven down even further. ECD

***Dr. Wenjun Sheng** is the CEO of Telink Semiconductor. Dr. Sheng has over 15 years of semiconductor industry experience, holds 30 patents in RF and mixed-signal IC design, and is a member of China's top recruitment program of global experts, the "Thousand Talents Program."*

FIGURE 1A AND 1B

C by GE light bulbs are an example of affordable smart home devices that can be connected using Bluetooth Low Energy (BLE) mesh networks and controlled via a companion app.

FIGURE 1A

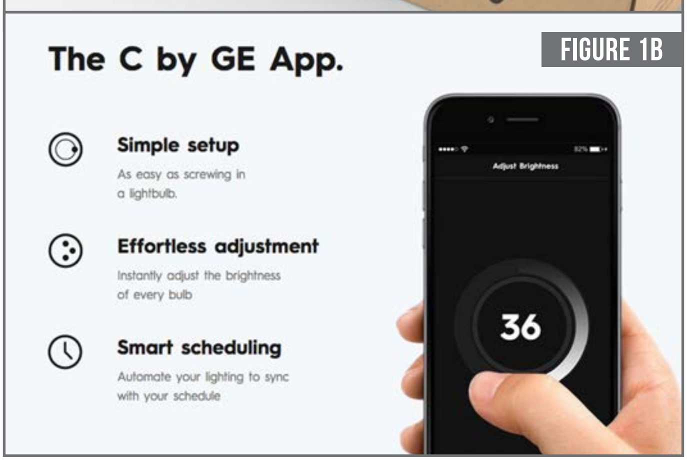


FIGURE 1B

MAGNETIC RESONANCE: THE NEXT GENERATION OF WIRELESS CHARGING

By Sanjay Gupta

With technological advances and ubiquitous connectivity enabled by Wi-Fi, Bluetooth, and 4G, the world is shifting to wireless. Seeing the convenience of replacing the wires in many of our devices and appliances, consumers are now, more than ever, demanding that same convenient and flexible experiences for charging our devices, too.

We've had the knowledge to deliver wireless power for some time. The first generation of wireless power products used the principles of magnetic induction. Electric toothbrushes were among the first products to employ this technology, and more recently some mobile phones have used magnetic induction for charging. These solutions, however, have had limited traction because while they represented a good first step, they haven't delivered the full flexibility and convenience that users want to charge their devices. For example, first-generation approaches required precise placement of the device on the charging source and couldn't keep up with wired charging speeds.

Magnetic resonance, on the other hand, meets those challenges. Also known as highly resonant non-radiative wireless energy transfer, magnetic resonance delivers in ways that magnetic induction cannot address. Magnetic resonance allows wireless charging over distance; multiple device charging; charging through materials (i.e. wood, granite, water, skin); and high efficiency of power transfer. Because of these benefits, magnetic resonance creates next-generation experiences for end users, enabling a seamless "drop and go" charging experience. In other words, consumers now have the ability to power "snack" all day long, wherever they go – grabbing a bit of power at home, at the office, in the car, etc. – without ever having to worry about plugging into an outlet or forgetting their power cord.

What this can do for design engineers

With the newest offering of magnetic resonance in semiconductor form, it is easier than ever to develop and embed this technology into end products. Magnetic resonance technology is attractive to design engineers because it gives them the freedom to create much lighter, thinner, more reliable, and less expensive devices. The technology enables delightful user experiences, while providing product developers and designers the ability to eliminate wires and failure-prone connectors for their end users.

Member companies of the AirFuel Alliance – the standards group behind magnetic resonance in consumer markets – are

"MAGNETIC RESONANCE ALLOWS WIRELESS CHARGING OVER DISTANCE; MULTIPLE DEVICE CHARGING; CHARGING THROUGH MATERIALS (WOOD, GRANITE, WATER, SKIN); AND HIGH EFFICIENCY OF POWER TRANSFER."

defining the technical specifications to ensure that interoperable solutions incorporating magnetic resonance technology can be brought to market. Today, the specification is mature and products are beginning to make their way through the approval process for certification.

Magnetic resonance at work

Consider these use cases for wireless charging that we can expect in the near future, enabled by magnetic resonance:

- The entire office experience will be more user-friendly and aesthetically pleasing as phones and laptops can charge and operate throughout the day without a wire in sight.
- Electric and autonomous vehicles can charge wirelessly by simply driving over a charging pad in the garage or parking lot, eliminating gas stations and making potentially autonomous vehicles even more self-sufficient.
- Numerous industrial applications will benefit from the removal of wires that pose hazards (for example, electrical wires in a drilling and mining environment are a fire hazard), carry electrical noise (critical to avoid in precision manufacturing), or are otherwise impractical.
- By going wireless, billions of power cords will be prevented from being manufactured and, later, from being dumped into landfills.

The stage is set for products incorporating magnetic resonance technology to hit the market. Key players across the ecosystem have been working together to bring this latest innovation to market and finally free us from the tangle of charging wires that surround us. ECD

***Sanjay Gupta** is Vice President of Product Management at WiTricity.*

WiTricity
www.witricity.com

TWITTER
@witricity

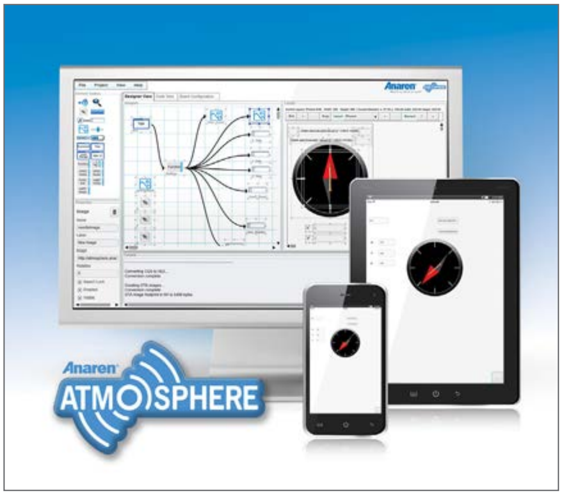
LINKEDIN
www.linkedin.com/company-beta/1442309

FACEBOOK
www.facebook.com/WiTricity

YOUTUBE
www.youtube.com/user/WiTricityCorp

Bluetooth Low Energy (BLE) embedded application development

Anaren's Atmosphere combines graphical drag-and-drop development with software building blocks for creating Bluetooth Low Energy (BLE) applications. The environment is free and tutorials, wiki, and a community forum are available to shrink learning curves for beginners and accelerate access for experienced developers. The environment features an integrated development environment (IDE) that allows a developer to simultaneously create a program for the embedded device and develop the mobile user interface. As one would expect, the IDE targets the development of IoT applications like wearables, GPS navigation, and smart homes and appliances, and includes a range of remote sensing device support.



Anaren
www.anaren.com | www.embedded-computing.com/p373280

Online Scrum Master training course

Exin, an independent certification institute, and Simplilearn, a leading provider of online training and certification courses, are teaming to provide a professional online Agile Scrum Master training program and certification course targeted at the North America market. The program covers all aspects and competencies needed to become a successful Agile Scrum Master for co-located teams, while also specifically addressing the needs and challenges of geographically-dispersed Scrum teams. The course provides e-learning content in a workshop-style method, replete with game-based learning and scrum agile scenarios.

EXIN
www.exin.com | www.embedded-computing.com/p373911

Simplilearn
www.simplilearn.com | www.embedded-computing.com/p373911

IIoT integration with Programmable Automation Controllers (PACs)

Opto 22 has announced availability of Node-RED nodes for Programmable Automation Controllers (PACs), which make prototyping easier and enable rapid application development when connecting industrial assets to cloud applications. RESTful application programming interface (API) supports Opto 22 SNAP PAC R-series and S-series controllers. Node-RED nodes allow quick-connect of legacy physical assets, providing an easy-to-use software API for connecting these assets to cloud services. Node-RED is an open-source, cross-platform technology available on Github.com and npmjs.org for OS X, Microsoft Windows, Linux, and Raspberry Pi platforms. The Node-RED library contains over 500 pre-built nodes that allow industrial Internet of Things (IIoT) developers to leverage existing code and deploy it directly into their environment.

Opto 22
www.opto22.com | www.embedded-computing.com/p373910

TOP TECH TO WATCH AT CES

Blind spot warning automatically detects trailers with short-range radar

Continental's Short Range Radar with Trailer Merge Assist and Trailer Range Detection is a smart sensor that's mounted on the back of a vehicle and serves as a blind spot warning system. Typically, the drawback to systems like this is that drivers must disable them when towing a trailer, lest they fall victim to false alarms. Not so with this system. The Short Range Radar can automatically sense and measure a trailer attached to the vehicle, extending object tracking beyond the length of the trailer with no user input required. Get a glimpse of this first-of-its-kind product at CES 2017 this January 5-8 in Las Vegas.

Continental
www.conti-online.com | embedded-computing.com/p373915

R2 3D printer has the end-to-end experience you're looking for

ROBO continues to bring high quality at a reasonable price to the consumer 3D printing space with their new R2 3D printer, which offers a 8" x 8" x 10" build volume; on-board camera, fast print speed; a removable, heated, and automatic self-leveling print bed; and the ability to use more than 30 different materials, as well as an additional extrusion head for printing two materials at once. It's also Wi-Fi enabled, which will allow users to print, manage, and monitor projects through their new mobile app. ROBO is taking pre-orders for the printer, which will be released in February 2017, but you can see it in action at CES 2017 this January 5-8 in Las Vegas.

ROBO
www.robo3d.com | www.embedded-computing.com/p373916

Jungo takes the wheel

Jungo Connectivity is introducing a new in-cabin monitoring system called CoDriver, which promises to improve safety in autonomous and semi-autonomous driving by monitoring the vehicle's occupants. Using deep learning, machine learning, and computer vision algorithms, CoDriver can gather information about the driver's state, including whether or not a driver is paying attention to the road, which could prompt a vehicle to take the wheel when a driver is distracted or yield control when a driver is ready to take over in advanced autonomous drive systems. The technology will be demonstrated at CES 2017 this January 5-8 in Las Vegas.

Jungo Connectivity
www.jungo.com | www.embedded-computing.com/p373917


SUPERMICRO
Video Surveillance
with High Capacity Storage & Max I/O
City Surveillance
Banking and Finance
Critical Infrastructure
Hospitals
Retail & Commercial
Transportation
X10DRX Motherboard
11x PCI-E Slots (10 x8 & 1 x4)
SC846XE2C
4U 24x 3.5" Drive Bays w/BBP
SYS-6048R-TXR
4U 5x 3.5" Drive Bays
SYS-6038-TXR
3U 8x 3.5" Drive Bays
SYS-2028R-TXR
2U 16x 2.5" Drive Bays
SSG-6048R-E1CR60N
4U 60x3.5" Drive Bays
SSG-6038R-E1CR16N
3U 16x3.5" Drive Bays
SSG-6028R-E1CR24N
2U 24x3.5" Drive Bays
SYS-5029S-TN2
Mini Tower 4x3.5" Drive Bays
SYS-E200-9B
Compact Box System
Low Power Intel® Quark™, Intel® Atom™ Intel®, Core™ processor families, and High Performance Intel® Xeon® processors
Performance Data Analytics
Security Fast Data Encryption
Media Transcoding
Storage Expansion
Video Surveillance with Max I/O
Remote Management OOB
Digital Video Solutions
Analog Video Solutions
intel XEON inside
Intel Inside®. Powerful Productivity Outside.
TITANIUM
We Power the Cloud
Learn more at www.supermicro.com/embedded
Intel, the Intel logo, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation in the U.S. and/or other countries.
03_Q3_SM_USP_082316_MasterFile