INTERNET OF THINGS (IoT) DESIGN CONSIDERATIONS FOR EMBEDDED CONNECTED DEVICES

ANDREW CAPLES
SENIOR PRODUCT MARKETING MANAGER, NUCLEUS

Mentor Graphics®

E M B E D D E D   S O F T W A R E

W H I T E P A P E R

INTRODUCTION

While we hear many promises of what the Internet of Things (IoT) will bring, the potential behind the IoT is hindered by the complexity of machine-to-machine (M2M) device software. Many early M2M systems consisted of remote devices in segmented networks relaying information back to a computer for supervisory decisions. For these systems, decisions were centralized, information flowed primarily one-way, and network segmentation provided adequate security. M2M did not rely on the public Internet so a “private” network with the proper precautions was all that was needed.

The emergence of the IoT today has changed the traditional M2M paradigm. The sheer breadth of M2M features now requires software developers to integrate code from numerous sources including home-grown, commercial, and open source in order to build devices that are connected, secure, and capable of making decisions. Unlike traditional M2M systems, the IoT model includes bidirectional data flow and relies on public networks to transmit much of the data. The promise of the IoT will allow smart grids to interact with home area networks in an effort to minimize energy consumption, vehicles will communicate with other vehicles to circumvent accidents, and remote patient monitors will deliver real time data for cost-effective medical care, to name a few examples.

The software required to build these connected IoT devices – devices that are secure and capable of autonomous network insertion in order to exchange information and services – is in great demand today. The promise of the IoT won’t be fulfilled until integrated software platforms are available that allow software developers to develop these devices efficiently and in the most cost-effective manner possible. And at the same time, meet the growing list of expanding M2M and networking requirements.

M2M FEATURES - FAR MORE COMPLEX IN AN IoT WORLD

On the surface the complexity of software can appear daunting. In order for the IoT to live up to its potential, low cost M2M devices will be required to seamlessly integrate onto “server-less” networks and communicate with other networked devices without manual intervention. A few additional requirements include:

- Routine configuration actions, such as IP address resolution, will need to be coordinated by both the existing device on the network and new devices that are introduced without the benefit of a network server managing the activity.
- Once networked, devices will be required to dynamically discover other networked devices and their resources while also introducing their own services. Some IoT protocols will allow devices to act as both a client and server depending on the use case or application.

Security requirements for M2M devices vary based on industry and market segments. However, to maintain data integrity, a few general principles apply:

- There has to be authentication before data transactions and data encryption before transmission to manage passive threats.
- Devices will have to withstand active threats, such as IP storms or floods.
- Because many devices are battery-operated and deployed remotely, power efficient systems must be able to take full advantage of the low power features of the processor and other hardware components in the system.

CONNECTIVITY FOR IoT NETWORKS

IoT networks must be scalable in order to support the dynamic nature of the IoT (as devices are added and removed from the network). For many applications, resource discovery and service announcements will need to be completed autonomously. Fortunately, zero configuration networking protocols such as multicast Domain Name System (mDNS) and DNS-based Service Directory (DNS-SD) support these services and can be used to integrate new devices to an IoT network. mDNS provides a channel for devices to broadcast services data without a centralized server. DNS-SD extends mDNS by providing service discovery. Devices can broadcast their services while discovering the services and resources of other devices.

To facilitate efficient M2M communication, Representational State Transfer (REST) architectures will also need to be leveraged. The benefits of REST include gains in network scalability, performance, and security. Because the REST architecture features a layered infrastructure designed to maintain separation between the client and server, REST-based systems are easily more scalable and support the seamless addition and removal of IoT devices.

It's also worth noting that as devices and various intermediaries are added to the network, the data paths can be altered which may negatively impact system performance. To correct this, caching data on devices closer to the client, such as proxy servers or gateways, network performance will be enhanced, or least maintained as the data paths change. Additionally, through the use of Uniform Resource Identifiers (URIs), network resources can be addressed by IoT clients. Using HTTP for the transmission, resources can be accessed and/or modified through commonly used protocols such as JSON and XML. For added protection, encryption can be used for safe data transmission HTTP(s).

IoT SECURITY

Because many IoT devices can be connected on dispersed public networks and support bidirectional data flow, these devices will be highly susceptible to attack. Just look at the network connectivity options today that did not exist 20 years ago; Wi-Fi, ZigBee, Bluetooth, and 4G cellular to name a few (one can appreciate the breadth of devices that will be connected.) Each connectivity option has very clear advantages – as well as disadvantages when it comes to secure communications. Without designing in security methods to address the full of range of threats, IoT devices are vulnerable to attacks from even the most unsophisticated methods. Without question, active and passive threats have to be detected, neutralized, and corrected before any harm to the individual device or the IoT system occurs.

It is essential that checkpoints basic and enhanced security be followed. A few of these checkpoints might include:

- Integration of security protocols for encryption and authentication must always be required.
- Before any data is transferred, the source of the data needs to be verified. The use of public keys and x.509 certificates are an extremely useful tool to verify the source of the data before any exchange is attempted.
- Online Certificate Status Protocol (OCSP) will streamline the client side resources required for x.509 certificate verification, which is significant for IoT devices with limited memory. Because OCSP responses to "authentication requests" it contains less information than a typical Certificate Revocation List (CRL). The complexity on the client side can be reduced by eliminating the need to parse through long lists.
- To address the potential for eavesdropping or other passive threats during a communication session, Transport Layer Security (TLS v1.2) (or OpenSSL) provide the foundation for security by encrypting data before transport. TLS relies on the use of contemporary encryption methods such as Advanced Encryption Standard (AES -256) and 3DES to provide a high level of encryption for IoT devices.

The use of encryption prevents the loss of data to passive listeners, but is does not prevent the alteration of data while traversing the network. Hash functions to generate Message Authentication Codes (MAC) are needed to ensure the integrity of the message and guarantee the content is not altered during transmission. Many networked devices are easily taken down by IP floods, storms, or a barrage of fragmented packets. To address these active threats, IoT devices must be designed to successfully detect attacks to prevent memory overflows or other faults that can disable a system. One way to maintain system integrity against these active threats is to design the device to be certifiable by WurldTech, an authority in IP security.

POWER MANAGEMENT FOR IoT DEVICES

Many IoT systems will be comprised of remote devices on dispersed networks that will act either as sensors, aggregators, and/or gateways. Requirements for these remote devices can include being as power efficient as possible either to extend battery life or to meet green energy goals.

The good news is silicon providers have designed processors with many low-power features including; Clock and Power Gating, CPU idle, Sleep modes, Dynamic Voltage and Frequency Scaling (DVFS), Standby and Hibernate. However, the task to develop a power efficient system falls on the application developer to write code that actually uses the low-power features of the silicon. Typically, application developers receive the system after peripheral drivers, board support packages, or embedded OSes with middleware have already been developed. Unless the underlying software has been designed to take advantage of the low-power features of the silicon, writing power-efficient code at the application level becomes very challenging or even an impractical exercise.

For example, an Operating Point transition (lowering the clock frequency to save power) requires the software to:

- **Verify the status** of each device on the system to ensure a transition can occur.
- **Verify the lowest operating frequency** the device can run at based on the current use-case to ensure each device can function at the new frequency.
- **Verify the amount of time** the device can be taken off-line (park latency) while the system clock is reduced to ensure there is no degradation to system or loss of data.
- **Determine the order** the device will be taken off-line and brought back on-line based on park latency.
- **Recalculate operating parameters**, for instance, the baud rate, if the reference was the system clock.

In order to maximize power efficiency, the embedded operating system must include a framework that supports the low-power features of the silicon and provides intuitive APIs. In this way the software developer can create systems that meet the much desired power requirements.

THE NEED FOR A FULL-FEATURED UNDERLYING RTOS FRAMEWORK

The Nucleus® RTOS provided by Mentor Graphics is a good example (Figure 1) of in integrated IoT solution which addresses the various challenges outlined above for developing connected IoT devices.

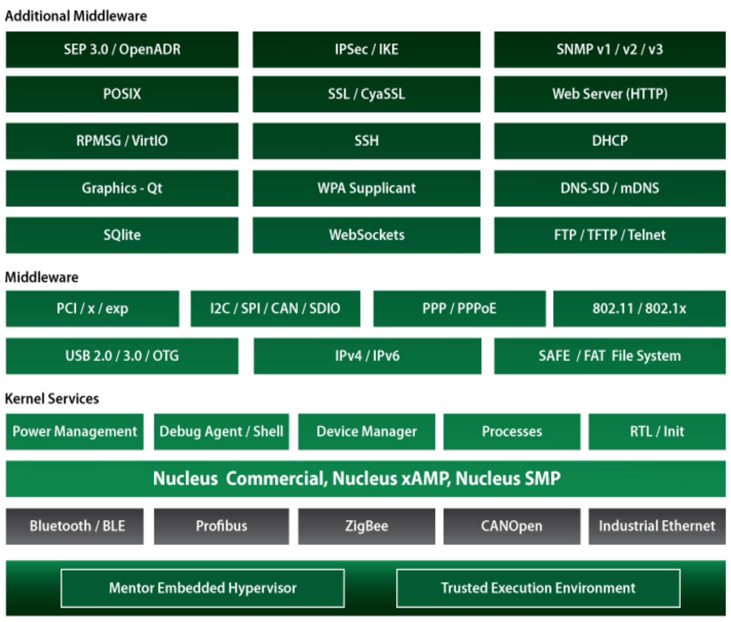

Figure 1: *The Nuclues RTOS ecosystem for the development of connected IoT devices.*

Nucleus is a widely deployed and scalable 3KB microkernel based RTOS that is designed to meet the M2M device requirements for IoT systems. It has hard, real-time performance and integrated power management services, connectivity support, and a vast array of networking protocols and security.

Further, Nucleus fits nicely into a memory constrained MCU-based device, and yet provides the functionality required for IoT systems. A few highlights of this particular RTOS include:

POWER MANAGEMENT

The extensive Power Management Framework available in Nucleus (Figure 2) directly maps to the low-power features found in today's more popular hardware. These features include: DVFS, idle, and sleep modes. IoT devices can be placed in various low-power modes through intuitive Nucleus API calls. For complex device transitions, such as moving into hibernate or standby mode, Nucleus provides the framework to safely turn off peripherals, move code into non-volatile memory, and change the operating point of the device.

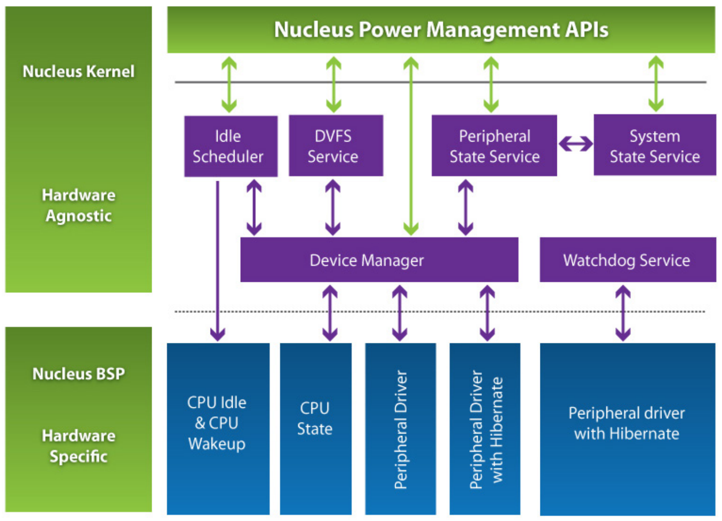

Figure 2: *Nucleus Power Management APIs simplify use of power-saving capabilities.*

CONNECTIVITY

Nucleus provides support for a vast array of connectivity options including: Wi-Fi, Bluetooth, Bluetooth low energy (BLE), USB 2.0/3.0 for IPv4/IPv6 based networks. Its modular and highly structured organization provides for the ability to install additional software protocols as requirements change.

SECURITY

Nucleus delivers end-to-end security options to protect data while in storage or during transmission. Device storage options include password protected secure databases that can store encrypted data. Transmission security includes TLS/SSL with encryption options including, AES-256, 3DES, DES, RC4 and many others. Online Certificate Support Protocol (OCSP) authentication support and Hash functions are available to ensure the integrity of the message and guarantee the content was not altered during transmission.

NETWORKING

Nucleus is available with a full featured IPv4/IPv6 stack with over 50 protocols and support for zero-configuration networking that includes mDNS and DNS-SD.

CONCLUSION

Software complexity is serving as a headwind for the development of systems that meet the full promise of IoT. Developing homegrown solutions and integrating with commercial and open source code presents numerous challenges and increases developmental risk.

As the market begins to accelerate, the need for cost-effective IoT devices will only increase. The use of a scalable and power-efficient RTOS with extensive networking and M2M protocols is required to develop cost-effective systems that meet the IoT requirements. Gone are the days of a limited or rudimentary OS that once operated a majority of M2M devices. With a full-featured underlying RTOS framework, software developers and design architects will significantly reduce their time to market and still realize all of their IoT application goals.

You can learn more about Nucleus Real-Time Operating System by visiting,http://www.mentor.com/embedded-software/nucleus

About the Author

Andrew Caples is a Product Marketing Manager for the Embedded Software Division (ESD) of Mentor Graphics. He has over 20 years of experience in start-ups and fortune 500 high tech companies and has served in a variety of roles ranging from technical marketing to sales management. He has a B.S. in Electrical and Computer Engineering from California Polytechnic University. His current responsibilities include product management for the Nucleus Real Time Operating System.

Corporate Headquarters
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, OR 97070-7777
Phone: 503.685.7000
Fax: 503.685.1204

Sales and Product Information
Phone: 800.547.3000
sales_info@mentor.com

Silicon Valley
Mentor Graphics Corporation
46871 Bayside Parkway
Fremont, CA 94538 USA
Phone: 510.354.7400
Fax: 510.354.7467

North American Support Center
Phone: 800.547.4303

Europe
Mentor Graphics
Deutschland GmbH
Arnulfstrasse 201
80634 Munich
Germany
Phone: +49.89.57096.0
Fax: +49.89.57096.400

Pacific Rim
Mentor Graphics (Taiwan)
11F, No. 120, Section 2,
Gongdao 5th Road
HsinChu City 300,
Taiwan, ROC
Phone: 886.3.513.1000
Fax: 886.3.573.4734

Japan
Mentor Graphics Japan Co., Ltd.
Gotenyama Garden
7-35, Kita-Shinagawa 4-chome
Shinagawa-Ku, Tokyo 140-0001
Japan
Phone: +81.3.5488.3033
Fax: +81.3.5488.3004

MGC 05-14 TECH12120-w