CONCEPTION DE L'INTERNET DES OBJETS (IoT)

CONSIDÉRATIONS POUR L'INTÉGRATION

APPAREILS CONNECTÉS

ANDRÉ CAPLES
RESPONSABLE MARKETING PRODUITS SÉNIOR, NUCLEUS

INTRODUCTION

Bien que nous entendions de nombreuses promesses sur ce qu'apportera l'Internet des objets (IoT), le potentiel derrière l'IoT est entravé par la complexité des logiciels des appareils machine à machine (M2M). De nombreux premiers systèmes M2M consistaient en des appareils distants dans des réseaux segmentés relayant les informations vers un ordinateur pour les décisions de supervision. Pour ces systèmes, les décisions étaient centralisées, les informations circulaient principalement à sens unique et la segmentation du réseau assurait une sécurité adéquate. Le M2M ne s'appuyait pas sur l'Internet public ; il suffisait donc d'un réseau « privé » doté des précautions appropriées.

L'émergence de l'IoT aujourd'hui a modifié le paradigme M2M traditionnel. L'étendue des fonctionnalités M2M oblige désormais les développeurs de logiciels à intégrer du code provenant de nombreuses sources, notamment locales, commerciales et open source, afin de créer des appareils connectés, sécurisés et capables de prendre des décisions. Contrairement aux systèmes M2M traditionnels, le modèle IoT inclut un flux de données bidirectionnel et s'appuie sur des réseaux publics pour transmettre une grande partie des données. La promesse de l'IoT permettra aux réseaux intelligents d'interagir avec les réseaux domestiques dans le but de minimiser la consommation d'énergie, aux véhicules de communiquer avec d'autres véhicules pour éviter les accidents, et aux moniteurs de patients à distance de fournir des données en temps réel pour des soins médicaux rentables. citer quelques exemples.

Les logiciels nécessaires à la construction de ces appareils IoT connectés – des appareils sécurisés et capables de s'insérer de manière autonome dans un réseau afin d'échanger des informations et des services – sont aujourd'hui très demandés.
La promesse de l'IoT ne se réalisera pas tant que des plates-formes logicielles intégrées ne seront pas disponibles, permettant aux développeurs de logiciels de développer ces appareils de manière efficace et de la manière la plus rentable possible. Et en même temps, répondez à la liste croissante des exigences croissantes en matière de M2M et de mise en réseau.

CARACTÉRISTIQUES M2M - BEAUCOUP PLUS COMPLEXES DANS UN MONDE IoT

À première vue, la complexité des logiciels peut paraître intimidante. Pour que l'IoT soit à la hauteur de son potentiel, les appareils M2M à faible coût devront s'intégrer de manière transparente aux réseaux « sans serveur » et communiquer avec d'autres appareils en réseau sans intervention manuelle. Quelques exigences supplémentaires incluent :

- Les actions de configuration de routine, telles que la résolution d'adresse IP, devront être coordonnées à la fois par le périphérique existant sur le réseau et par les nouveaux périphériques introduits sans l'avantage d'un serveur réseau gérant l'activité.

- Une fois mis en réseau, les appareils devront découvrir dynamiquement d'autres appareils en réseau et leurs ressources tout en introduisant leurs propres services. Certains protocoles IoT permettront aux appareils d'agir à la fois en tant que client et serveur en fonction du cas d'utilisation ou de l'application.

Les exigences de sécurité pour les appareils M2M varient en fonction du secteur et des segments de marché. Cependant, pour maintenir l'intégrité des données, quelques principes généraux s'appliquent :

- Il doit y avoir une authentification avant les transactions de données et un cryptage des données avant la transmission pour gérer les menaces passives.
- Les appareils devront résister à des menaces actives, telles que des tempêtes IP ou des inondations.
- Étant donné que de nombreux appareils fonctionnent sur batterie et sont déployés à distance, les systèmes économes en énergie doivent être capable de tirer pleinement parti des fonctionnalités de faible consommation du processeur et des autres composants matériels du système.

CONNECTIVITÉ POUR LES RÉSEAUX IoT

Les réseaux IoT doivent être évolutifs afin de prendre en charge la nature dynamique de l'IoT (à mesure que des appareils sont ajoutés et supprimés du réseau). Pour de nombreuses applications, la découverte des ressources et les annonces de services devront être effectuées de manière autonome. Heureusement, les protocoles réseau sans configuration tels que le système de noms de domaine multicast (mDNS) et l'annuaire de services basé sur DNS (DNS-SD) prennent en charge ces services et peuvent être utilisés pour intégrer de nouveaux appareils à un réseau IoT. mDNS fournit un canal permettant aux appareils de diffuser des données de services sans serveur centralisé. DNS-SD étend mDNS en fournissant la découverte de services. Les appareils peuvent diffuser leurs services tout en découvrant les services et ressources d'autres appareils.

Pour faciliter une communication M2M efficace, les architectures REST (Representational State Transfer) devront également être exploitées. Les avantages de REST incluent des gains en termes d'évolutivité, de performances et de sécurité du réseau. Étant donné que l'architecture REST comporte une infrastructure en couches conçue pour maintenir la séparation entre le client et le serveur, les systèmes basés sur REST sont facilement plus évolutifs et prennent en charge l'ajout et la suppression transparents d'appareils IoT.

Il convient également de noter qu'à mesure que des appareils et divers intermédiaires sont ajoutés au réseau, les chemins de données peuvent être modifiés, ce qui peut avoir un impact négatif sur les performances du système. Pour corriger cela, en mettant en cache les données sur des appareils plus proches du client, tels que des serveurs proxy ou des passerelles, les performances du réseau seront améliorées, ou moins maintenues, à mesure que les chemins de données changent. De plus, grâce à l'utilisation d'identificateurs de ressources uniformes (URI), les ressources réseau peuvent être adressées par les clients IoT. En utilisant HTTP pour la transmission, les ressources peuvent être consultées et/ou modifiées via des protocoles couramment utilisés tels que JSON et XML. Pour une protection supplémentaire, le cryptage peut être utilisé pour la transmission sécurisée des données HTTP(s).

SÉCURITÉ IdO

Étant donné que de nombreux appareils IoT peuvent être connectés sur des réseaux publics dispersés et prendre en charge un flux de données bidirectionnel, ces appareils seront très vulnérables aux attaques. Il suffit de regarder les options de connectivité réseau d'aujourd'hui qui n'existaient pas il y a 20 ans ; Wi-Fi, ZigBee, Bluetooth et cellulaire 4G pour n'en nommer que quelques-uns (on peut apprécier l'étendue des appareils qui seront connectés.) Chaque option de connectivité présente des avantages très évidents – ainsi que des inconvénients en matière de communications sécurisées. Sans une conception de méthodes de sécurité permettant de répondre à l'ensemble des menaces, les appareils IoT sont vulnérables aux attaques, même provenant des méthodes les plus simples.
Il ne fait aucun doute que les menaces actives et passives doivent être détectées, neutralisées et corrigées avant que tout dommage causé à l'appareil individuel ou au système IoT ne se produise.

Il est essentiel que les points de contrôle de sécurité de base et renforcés soient respectés. Quelques-uns de ces points de contrôle peuvent inclure :

- L'intégration de protocoles de sécurité pour le chiffrement et l'authentification doit toujours être requise.
- Avant qu'une donnée soit transférée, la source des données doit être vérifiée. L'utilisation de clés publiques et de certificats x.509 est un outil extrêmement utile pour vérifier la source des données avant toute tentative d'échange.
- Le protocole OCSP (Online Certificate Status Protocol) rationalisera les ressources côté client requises pour x.509. vérification du certificat, ce qui est important pour les appareils IoT avec une mémoire limitée. Étant donné que les réponses OCSP aux « demandes d'authentification », elles contiennent moins d'informations qu'une liste de révocation de certificats (CRL) classique. La complexité côté client peut être réduite en éliminant le besoin d'analyser de longues listes.
- Pour faire face au risque d'écoute clandestine ou d'autres menaces passives lors d'une session de communication, Transport Layer Security (TLS v1.2) (ou OpenSSL) constitue la base de la sécurité en chiffrant les données avant le transport. TLS s'appuie sur l'utilisation de méthodes de cryptage contemporaines telles que Advanced Encryption Standard (AES -256) et 3DES pour fournir un niveau élevé de cryptage pour les appareils IoT.

L'utilisation du cryptage empêche la perte de données pour les auditeurs passifs, mais n'empêche pas l'altération des données lors de leur traversée du réseau. Des fonctions de hachage pour générer des codes d'authentification de message (MAC) sont nécessaires pour garantir l'intégrité du message et garantir que le contenu n'est pas modifié pendant la transmission.
De nombreux appareils en réseau sont facilement mis hors service par des inondations IP, des tempêtes ou un barrage de paquets fragmentés. Pour faire face à ces menaces actives, les appareils IoT doivent être conçus pour détecter avec succès les attaques afin d'éviter les débordements de mémoire ou d'autres défauts susceptibles de désactiver un système. Une façon de maintenir l'intégrité du système contre ces menaces actives consiste à concevoir l'appareil pour qu'il soit certifiable par WurldTech, une autorité en matière de sécurité IP.

GESTION DE L'ÉNERGIE POUR LES APPAREILS IoT

De nombreux systèmes IoT seront composés d'appareils distants sur des réseaux dispersés qui agiront comme des capteurs, des agrégateurs et/ou des passerelles. Les exigences relatives à ces appareils distants peuvent inclure une efficacité énergétique aussi élevée que possible, soit pour prolonger la durée de vie de la batterie, soit pour atteindre les objectifs en matière d'énergie verte.

La bonne nouvelle est que les fournisseurs de silicium ont conçu des processeurs dotés de nombreuses fonctionnalités à faible consommation, notamment : Clock and Power Gating, CPU inactif, modes veille, mise à l'échelle dynamique de tension et de fréquence (DVFS), veille et mise en veille prolongée. Cependant, la tâche de développer un système économe en énergie incombe au développeur d'applications qui doit écrire du code qui utilise réellement les fonctionnalités de faible consommation du silicium. En règle générale, les développeurs d'applications reçoivent le système une fois que les pilotes de périphériques, les packages de support de carte ou les systèmes d'exploitation intégrés avec middleware ont déjà été développés. À moins que le logiciel sous-jacent n'ait été conçu pour tirer parti des fonctionnalités de faible consommation du silicium, l'écriture de code économe en énergie au niveau de l'application devient très difficile, voire peu pratique.

Par exemple, une transition de point de fonctionnement (abaissement de la fréquence d'horloge pour économiser de l'énergie) nécessite que le logiciel :

- Vérifiez l'état de chaque périphérique sur le système pour garantir qu'une transition peut avoir lieu.
- Vérifiez la fréquence de fonctionnement la plus basse à laquelle l'appareil peut fonctionner en fonction de la cas d'utilisation pour garantir que chaque appareil peut fonctionner à la nouvelle fréquence.
- Vérifiez la durée pendant laquelle l'appareil peut être mis hors ligne (latence de stationnement) pendant que l'horloge système est réduite pour garantir qu'il n'y a pas de dégradation du système ou de perte de données.
- Déterminer l'ordre dans lequel l'appareil sera mis hors ligne et remis en ligne. basé sur la latence du parc.
- Recalculer les paramètres de fonctionnement, par exemple le débit en bauds, si la référence était l'horloge système.

Afin de maximiser l'efficacité énergétique, le système d'exploitation embarqué doit inclure un cadre prenant en charge les fonctionnalités de faible consommation du silicium et fournissant des API intuitives. De cette façon, le développeur de logiciels peut créer des systèmes qui répondent aux besoins en énergie tant souhaités.

LE BESOIN D'UN RTOS SOUS-JACENT COMPLET CADRE

Le Nucleus® RTOS fourni par Mentor Graphics est un bon exemple (Figure 1) de solution IoT intégrée qui répond aux différents défis décrits ci-dessus pour le développement d'appareils IoT connectés.

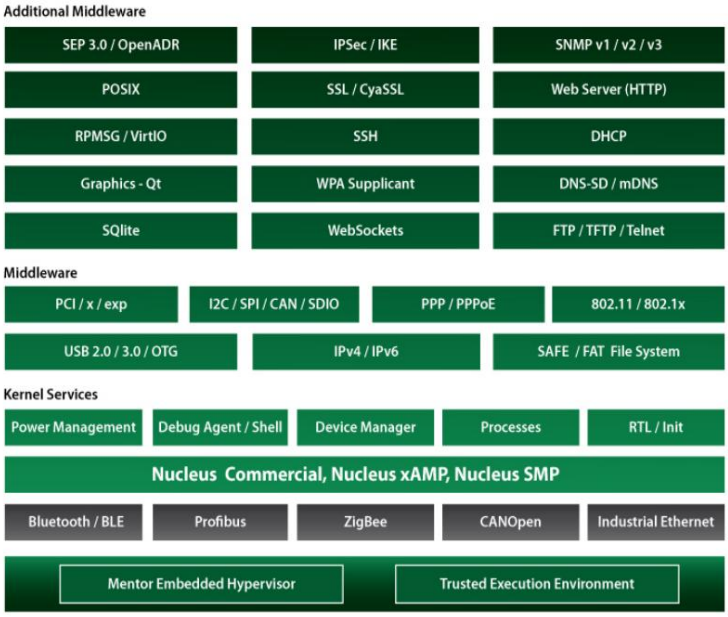

Figure 1 : L'écosystème Nuclues RTOS pour le développement d'appareils IoT connectés.

Nucleus est un RTOS largement déployé et évolutif basé sur un micronoyau de 3 Ko, conçu pour répondre aux exigences des appareils M2M pour les systèmes IoT. Il offre des performances en temps réel et des services de gestion de l'énergie intégrés, une prise en charge de la connectivité et une vaste gamme de protocoles de mise en réseau et de sécurité.

De plus, Nucleus s'intègre parfaitement dans un appareil basé sur MCU à mémoire limitée, tout en fournissant les fonctionnalités requises pour les systèmes IoT. Voici quelques points forts de ce RTOS particulier :

GESTION DE L'ÉNERGIE

Le vaste cadre de gestion de l'alimentation disponible dans Nucleus (Figure 2) correspond directement aux fonctionnalités de faible consommation que l'on trouve dans le matériel le plus populaire d'aujourd'hui. Ces fonctionnalités incluent : les modes DVFS, inactif et veille. Les appareils IoT peuvent être placés dans différents modes de faible consommation grâce à des appels intuitifs de l'API Nucleus. Pour les transitions complexes de l'appareil, telles que le passage en mode veille prolongée ou veille, Nucleus fournit le cadre permettant d'éteindre les périphériques en toute sécurité, de déplacer le code dans la mémoire non volatile et de modifier le point de fonctionnement de l'appareil.

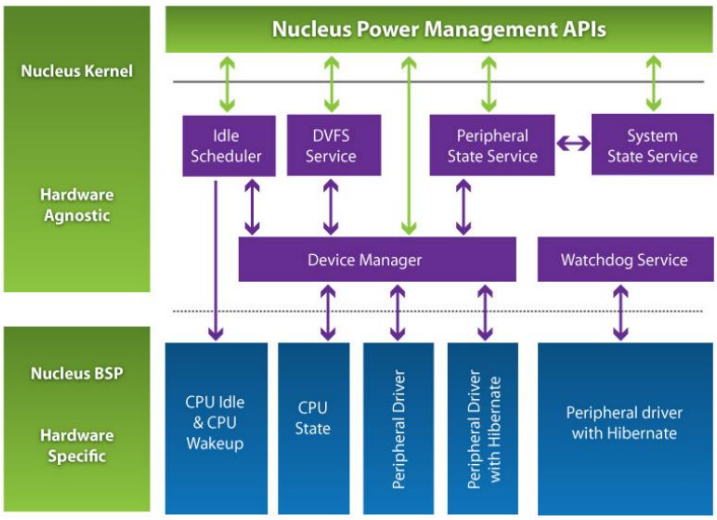

Figure 2 : Les API Nucleus Power Management simplifient l'utilisation des fonctionnalités d'économie d'énergie.

CONNECTIVITÉ

Nucleus prend en charge une vaste gamme d'options de connectivité, notamment : Wi-Fi, Bluetooth, Bluetooth Low Energy (BLE), USB 2.0/3.0 pour les réseaux IPv4/IPv6. Son organisation modulaire et hautement structurée offre la possibilité d'installer des protocoles logiciels supplémentaires en fonction de l'évolution des besoins.

SÉCURITÉ

Nucleus offre des options de sécurité de bout en bout pour protéger les données pendant leur stockage ou pendant leur transmission. Les options de stockage de l'appareil incluent des bases de données sécurisées protégées par mot de passe qui peuvent stocker des données cryptées. La sécurité de la transmission inclut TLS/SSL avec des options de cryptage notamment AES-256, 3DES, DES, RC4 et bien d'autres.
La prise en charge de l'authentification OCSP (Online Certificate Support Protocol) et les fonctions de hachage sont disponibles pour garantir l'intégrité du message et garantir que le contenu n'a pas été modifié pendant la transmission.

RÉSEAU Nucleus est

disponible avec une pile IPv4/IPv6 complète avec plus de 50 protocoles et une prise en charge de la mise en réseau sans configuration qui inclut mDNS et DNS-SD.

CONCLUSION

La complexité logicielle constitue un obstacle au développement de systèmes qui répondent pleinement aux promesses de l'IoT. Développer des solutions locales et les intégrer à du code commercial et open source présente de nombreux défis et augmente le risque de développement.

À mesure que le marché commence à s'accélérer, le besoin d'appareils IoT rentables ne fera qu'augmenter. L'utilisation d'un RTOS évolutif et économe en énergie avec des protocoles réseau et M2M étendus est nécessaire pour développer des systèmes rentables qui répondent aux exigences de l'IoT. Il est révolu le temps d'un système d'exploitation limité ou rudimentaire qui exploitait autrefois la majorité des appareils M2M. Grâce à un cadre RTOS sous-jacent complet, les développeurs de logiciels et les architectes de conception réduiront considérablement leurs délais de commercialisation tout en atteignant tous leurs objectifs en matière d'applications IoT.

Vous pouvez en savoir plus sur le système d'exploitation Nucleus Real-Time en visitant http://www.mentor.com/embedded-software/nucleus

À propos de l'auteur

Andrew Caples est responsable du marketing produit pour la division des logiciels embarqués (ESD) de Mentor Graphics. Il possède plus de 20 ans d'expérience dans des start-ups et des entreprises de haute technologie classées Fortune 500 et a occupé divers postes allant du marketing technique à la gestion des ventes. Il est titulaire d'un baccalauréat en génie électrique et informatique de l'Université polytechnique de Californie. Ses responsabilités actuelles incluent la gestion de produit pour le système d'exploitation en temps réel Nucleus.

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