a- FreeRTOS

Mit Mikrocontrollern in die Cloud:

Brücke in die IoT-Welt

Aufmacher: (Orla – Shutterstock)

Das Internet der Dinge erobert unseren Alltag und die Industrie. Doch bei Umsetzung und Nutzung des IoT stehen Anwender häufig vor zahlreichen Fragen und Problemen. Hier kann AWS mit Lösungen helfen, die die Kette vom Mikrocontroller bis zur fertigen Heim-, Industrie- oder Geschäftsanwendung abdecken können.

Von Constantin Gonzalez

Der Anwendungsbereich des Internet der Dinge (IoT) reicht weit – vom Industrial IoT (IIoT) im Konzept des Industrie 4.0 bis hin zu Smart-Home-Anwendungen wie der „intelligenten Glühlampe". In dem Prozess, IoT für sich zu nutzen, stellen Anwender häufig folgende Fragen: Wie können einfache Geräte ohne „großes" Betriebssystem mit dem IoT verbunden werden? Was ist, wenn die Verbindung in die Cloud unzuverlässig ist? Was, wenn Netzwerk-Latenzen eine Steuerung von Echtzeit-Komponenten unmöglich machen? Wie können Daten, die in der Cloud zur Verfügung stehen, genutzt werden? Hierfür haben Hersteller wie Amazon Web Services (AWS) passende Lösungen entwickelt.

Typische Cloud-Protokolle verwenden HTTPS-basierte RESTful Web Services, die leistungsstark und sicher sind, aber auch verhältnismäßig viel Speicher- und CPU-Aufwand erfordern. Kein Problem für PC, Notebook und Mobiltelefon oder z.B. das eine oder andere Projekt mit dem Raspberry Pi. Mikrocontroller, die kleiner und stromsparender sind, können jedoch mit solchen Protokollen schnell überfordert sein.

Auf der anderen Seite ist es für Anwendungen wie Sensordaten-Erfassung, Smart Watches oder Fitness-Tracker nicht kosten- oder energieeffizient, einen „richtigen" Rechner zu verbauen, der Web-Protokolle unterstützen könnte. Hier hilft oft eine Zweiteilung der Konnektivität: Ein schlankes Protokoll wie Zigbee oder Bluetooth verbindet das Endgerät mit einem Brückengerät – z. B. ein Smart Home Hub, einem Router oder ein Mobiltelefon. Das wiederum stellt die Verbindung in die Cloud her.

Allerdings können derartige Szenarien schnell kompliziert werden: Wie können die Netzwerk-Verbindungen abgesichert werden? Wie funktionieren in so einem Szenario Firmware-Updates? Wie werden große Mengen von Mikrocontroller-Geräten durch Connectivity-Hubs hindurch verwaltet? Auch dann, wenn Mikrocontroller mächtig genug sind, um eigene Verbindungen zu Cloud-Diensten mithilfe von Protokollen wie HTTPS oder MQTT über TLS (MQTTS) aufzubauen, bleiben diese Fragen häufig offen, und das Potenzial für schlecht erprobte und damit unsichere Lösungen ist groß.

Hier hat AWS mit Software wie Amazon FreeRTOS und AWS Greengrass sowie dem Cloud-Service AWS IoT Core eine Lösung geschaffen, die Entwicklern hilft, Mikrocontroller und andere IoT-Geräte sicher und skalierbar mit der

Begriffserklärungen

- Amazon FreeRTOS: IoT-Betriebssystem für Mikrocontroller. Erweitert den FreeRTOS-Kernel um Libraries für Security, Verbindungsaufbau und Update-Fähigkeit.
- AWS Greengrass: Sichere, direkte Verbindung zu IoT-Edge-Geräten. Lokale Logik.
- Greengrass Group: Eine definierte Gruppe mit einem Greengrass Core und anderen Geräten, die für die Kommunikation untereinander konfiguriert sind. Eine Greengrass Group kann z. B. ein Stockwerk in einem Gebäude, ein Fahrzeug oder einen Haushalt repräsentieren.
- Greengrass Core: Der Greengrass Core ist die Laufzeitumgebung, die die lokale Ausführung von AWS-Lambda-Funktionen, Device Shadows und Sicherheitsfunktionen ermöglicht. Der Greengrass Core interagiert direkt mit der Cloud und arbeitet bei unterbrochener Verbindung auch lokal.

Cloud zu verbinden und diese Geräte zuverlässig zu aktualisieren und zu verwalten.

Echtzeit-OS für IoT-Anwendungen

Wie der Name schon andeutet, ist Amazon FreeRTOS ein Open-Source-Echtzeit-Betriebssystem für Mikrocontroller, das auf dem populären FreeRTOS-Betriebssystem aufbaut und es um eine Reihe von Libraries ergänzt, die die Nutzung von FreeRTOS für IoT-Anwendungen optimiert:

- Das Amazon FreeRTOS – kurz „a:FreeRTOS" – ist kostenlos und unter der MIT-Open-Source-Lizenz freigegeben. Entwickler können damit völlig nahtlos an das existierende FreeRTOS-Ökosystem mit seinen Werkzeugen und passender Software anknüpfen, da a:FreeRTOS auf dem unmodifizierten FreeRTOS-Kernel aufsetzt.
- Das Betriebssystem kommt mit einer Reihe von nützlichen Libraries, die typische IoT-Fähigkeiten wie die Konfiguration des Mikrocontrollers für lokale Netzwerke oder over-the-air (OTA) Updates erleichtern.
- Über die Amazon-FreeRTOS-Konsole können Entwickler ihre eigene a:FreeRTOS-Distribution konfigurieren, indem sie eine der unterstützten Hardware-Plattformen und die benötigten Zusatz-Libraries je nach Anforderung auswählen. Daraufhin können sie von dort ihre maßgeschneiderte a:FreeRTOS-Distribution herunterladen und installieren.
- Das Betriebssystem legt besonderen Fokus auf die Sicherheit von Endgeräten und IoT-Anwendungen: Es kommt mit Libraries für Datenverschlüsselung und Schlüsselmanagement sowie TLS 1.2 für die verschlüsselte Netzwerk-Kommunikation. Eine Code-Signier-Funktion ermöglicht den Nachweis, dass der Programmcode etwa bei Updates nicht verändert wurde.
- Geräte, die Amazon FreeRTOS verwenden, können je nach Ausstattung direkt mit Cloud-Diensten über sichere Protokolle kommunizieren oder über ein lokales Gerät als Brücke ins Internet. Für diesen Fall bietet Amazon mit „AWS Greengrass" eine passende Edge-Computing-Lösung.
- Neben der Software und den zusätzlichen Libraries qualifiziert AWS a:FreeRTOS für eine Reihe von Chipsätzen und Architekturen wie MIPS und ARM, in Verbindung mit Partnern wie Espressif, Microchip, NXP Semiconductors, STMicroelectronics, Texas Instruments und weiteren Hardwareanbietern. Das Qualifikationsprogramm steht jedem Hardwarehersteller offen.

Die Nutzung des Betriebssystems ist für Hardware-Entwickler einfach: Der Anwender sucht sich einen passenden Mikrocontroller aus der Liste der unterstützten Geräte heraus. Danach stellt er sich in der Amazon FreeRTOS-Konsole die benötigten Libraries zusammen und lädt anschließend eine angepasste Distribution von a:FreeRTOS samt Hardware-spezifischer Unterstützung und aller gewählten Libraries herunter. Sobald das Betriebssystem auf dem Gerät installiert ist, kann es für die Verbindung mit lokalen Edge-Geräten oder Cloud Services wie AWS IoT Core konfiguriert werden. Die eingebaute Unterstützung von Firmware-Updates erleichtert den Aufbau einer Over-the-Air-Update-Funktion.

Edge-Computing mit AWS Greengrass

Grundsätzlich erlaubt FreeRTOS herstellerunabhängiges Arbeiten, in dem es offene Standardprotokolle wie MQTT und TLS 1.2 verwendet. Der a:FreeRTOS-Mikrocontroller kann also problemlos mit diversen IoT-Diensten verknüpft werden. Außerdem können beliebige Edge-Lösungen angebunden werden. Wer eine Lösung für den Aufbau und das Management von Edge-Computing-Geräten nicht selber entwickeln möchte, kann die Greengrass-Software nutzen, die dem IoT-Entwickler viele typische Edge-Computing-Aufgaben erleichtern kann.

Die Software unterstützt die koordinierte Implementation von lokalem Code, die Verwaltung von Nachrichten nach dem MQTT-Standard, das Zwischenspeichern – das Caching – und die Synchronisation von Daten sowie auch die Nutzung von Machine-Learning-Modellen auf Edge-Geräten. Dabei agiert Greengrass als lokaler Hub – Greengrass Core – für Endgeräte wie Mikrocontroller, die über lokale Protokolle wie MQTTS, OPC UA oder andere Schnittstellen an Greengrass gekoppelt werden können. Angebundene Endgeräte werden „Greengrass aware" genannt. Greengrass kann auch eigenständig als IoT-Komponente genutzt werden, z. B. als Software-Plattform in Industrie-PCs oder -Controllern. Damit ermöglicht AWS die nahtlose Integration von beliebigen Geräten aller Größen mit Cloud-Diensten in unterschiedlichen IoT-Szenarien.

Über den offenen MQTT-Standard oder Industrie-spezifischen Protokollen wie OPC UA können Geräte untereinander, mit Greengrass-Core-Geräten oder mit der Cloud (MQTT, HTTPS) kommunizieren. Gemäß dem MQTT-Standard werden dabei Nachrichten in Topics

sortiert, die abonniert werden können, um den Nachrichtenfluss zu steuern. Zur Verschlüsselung wird der TLS-1.2-Standard verwendet, X.509-Zertifikate sorgen für Authentifizierung auf beiden Seiten. Ein Policy-Management erlaubt dabei eine fein granulare Vergabe von Rechten im Sinne eines Least-Privilege-Modells.

Oft sind Geräte zeitweise vom Netzwerk getrennt, beispielsweise wenn Fahrzeuge in einen Tunnel fahren oder Sensoren in entlegenen Gebieten betrieben werden. In diesen Fällen kann Greengrass als lokaler Hub die Kontrolle übernehmen und mithilfe von Message Queues Zustände puffern sowie mit Device-Shadows-Zustände mit der Cloud bzw. mit an Greengrass Core angeschlossenen Geräten synchronisieren. Dieses Device-Shadow-Modell erleichtert Entwicklern den Umgang mit

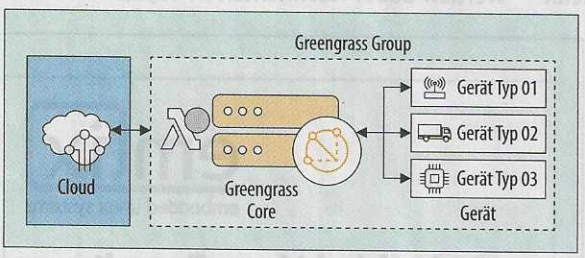

Bild 1. AWS Greengrass als Brücke zwischen Cloud und Geräten.

(Quelle: AWS)

Verbindungs-Abbrüchen: Der Device Shadow verhält sich wie ein Gerät, das immer online ist und das sich automatisch mit dem echten Gerät abstimmt und die Zustände synchronisiert. Device Shadows können sowohl innerhalb von Greengrass-Core-Geräten genutzt werden als auch in der AWS-Cloud über den AWS-IoT-Core-Dienst.

Manche IoT-Anwendungen erfordern kurze Reaktionszeiten, für die typische Netzwerk-Latenzen zu hoch sind. Beispiele hierbei sind Roboter-Steuerungen oder Regelkreise. Für diese Fälle stellt Greengrass eine Software-Umgebung für sogenannte Lambda-Funktionen bereit, in denen Code lokal ausgeführt werden kann, der entweder dauerhaft läuft oder über bestimmte Ereignisse – z. B. MQTT-Messages – gestartet werden kann, um darauf zu reagieren. Das Lambda-Modell standardisiert die Verwaltung von Benutzer-Code und schottet einzelne Funktionen voneinander ab. Mit solchen Lambda-Funktionen können auch beliebige Protokolle konvertiert oder physische Schnittstellen wie serielle Busse angebunden werden.

Auch bei Greengrass steht die Sicherheit im Vordergrund: Über verschlüsselte Verbindungen und Zertifikat-Authentifizierungen für alle Seiten wird sichergestellt, dass zu keinem Zeitpunkt Daten unkontrolliert zwischen Greengrass-Core-Geräten und der Cloud ausgetauscht werden. Greengrass-aware-Geräte sowie im Greengrass Core laufende Lambda-Funktionen können über Subscriptions verschaltet werden, die die Kommunikation untereinander und das Triggern der Lambda-Funktionen konfigurierbar regeln (**Bild 1**).

Gerade in einer Kombination mit Greengrass können Geräte, die FreeRTOS als Echtzeitbetriebssystem verwenden, eine sichere, robuste und leistungsfähige IoT-Architektur aufbauen, die auch Logik vor Ort für Echtzeit-Entscheidungen unterstützt und dabei unabhängig von der Cloud arbeiten kann. Das Preismodell richtet sich nach der Anzahl der Greengrass-Core-Geräte und der weltweiten AWS-Region, in der sie genutzt werden.

Noch mehr Möglichkeiten ergeben sich durch die Integration von FreeRTOS und Greengrass in Zusammenarbeit mit Cloud-Diensten. Hierbei haben Kunden die Wahl, eigene Dienste aufzubauen und sie über Protokolle wie MQTTS oder HTTPS anzubinden, oder sie verwenden den Dienst AWS IoT Core:

Geräte sicher in Cloud-Dienste einbinden

In der Cloud angekommen, ergeben sich für Entwickler eine Fülle von Möglichkeiten, aus den IoT-Daten Mehrwerte für beliebige Applikationen zu schaffen. Dabei bildet der AWS IoT Core-Service das Rückgrat. Er liefert einen MQTT Message Broker, der für die Verbindung von IoT-Geräten mit dem Cloud-Dienst sorgt. Prinzipiell kann jedes Gerät mit AWS IoT Core vernetzt werden, das MQTT mit TLS-1.2- und X.509-Zertifikaten oder Websockets über HTTPS mit dem AWS-eigenen SigV4-Signierungs-Algorithmus implementieren kann. Das gilt für besser ausgestattete Mikrocontroller mit dem FreeRTOS-Betriebssystem und entsprechend vorkonfigurierten Libraries oder für Geräte mit Greengrass Core, die entweder allein als Hub dienen können oder für beliebige andere Geräte, die diese Protokolle sprechen können – etwa Raspberry Pi, Industrie-Controller auf PC-Basis, Apps für Handy oder Tablet oder auch Web-Applikationen mit JavaScript. So sind auch gemischte Setups möglich, bei denen z. B. der Mikrocontroller Sensor-Daten über MQTT liefert, die von einer Web-Applikation in JavaScript in „Nahe-Echtzeit" empfangen und angezeigt werden können.

Die für die Authentifizierung benötigten X.509-Zertifikate werden vom IoT Core automatisiert bereit gestellt; sie bilden zusammen mit der TLS-1.2-Verschlüsselung das Fundament für eine sichere Kommunikation zwischen Geräten und Cloud. Zusätzlich können Browser und z. B. Apps auf Mobiltelefonen über Identity-Federation angebunden werden. Policies erlauben dabei die fein granulare Regelung von Kommunikations-Strömen auf MQTT-Topic- und -Subtopic-Ebene, sodass auch komplexe Security-Setups realisiert werden können. So kann z. B. in einer Taxi-Applikation über die geschickte Nutzung von Topics und Policies erreicht werden, dass GPS-Positionen von Fahrzeugen nur mit Nutzern ausgetauscht werden, die das Fahrzeug auch gebucht haben und umgekehrt die Nutzer-Position nur Fahrzeugen zugänglich gemacht wird, die vom Nutzer gebucht wurden. Nach erfolgter Fahrt können diese Berechtigungen problemlos entzogen werden, indem Fahrt-spezifische Topics nicht mehr genutzt oder Policies angepasst werden.

Die Verwaltung von Geräten und ihrer Zertifikate wird über eine eingebaute Device Registry erledigt. Für größere Geräteflotten und weiter gehende Verwaltungs-Aufgaben wie z. B. das Management von Firmware-Updates oder Fernwartungs-Protokolle bietet AWS optional mit AWS IoT Device Management einen eigenen Service an.

Die Interaktion von Nachrichten von Geräten untereinander oder mit Backend-Services in der Cloud oder im eigenen Rechenzentrum wird über einen regelbasierten Mechanismus geregelt – der AWS IoT Rules Engine: Entwickler können dort Regeln hinterlegen, die mithilfe einer SQL-artigen Konfiguration Nachrichten im MQTT-Message-Strom des Brokers filtern und diese an konfigurierbare Aktionen weiterleiten. Solche Aktionen können eine einfache Weiterleitung von Messages an andere Topics sein oder AWS-spezifische Aktionen, wie die Aktualisierung von Daten

in der Amazon-DynamoDB-Datenbank, die Weiterleitung von Messages an eine Amazon SQS Message Queue oder das Starten einer AWS-Lambda-Funktion, in der beliebiger Code in einer der unterstützten Sprachen – z. Zt. Python, Java, Go, C#, Node.js und PowerShell – die Nachricht weiter verarbeiten kann. Letzterer Mechanismus öffnet der Anwendung Tür und Tor in beliebige Integrationen, etwa mit Services aus dem eigenen Rechenzentrum, anderen Cloud-Providern oder SaaS-Diensten.

Ähnlich wie Greengrass unterstützt IoT Core ebenfalls einen Shadow-Mechanismus zur Kommunikation und zur Synchronisierung mit Geräten, die zeitweise offline sind: Entwickler können über spezielle Messages Zustandsänderungen in einem Device Shadow festhalten, die über ein Protokoll mit dem Gerät synchronisiert werden, wenn es online ist.

Eine Reihe von AWS IoT Device SDKs für Mobiltelefone sowie andere Geräte und Programmiersprachen – Arduino Yún, Embedded C, C++, Java, JavaScript, Python – erleichtert die Programmierung für Entwickler unterschiedlicher Geräte-Typen. Neben einfach nutzbarer MQTT- bzw. Websocket-Libraries enthalten die SDKs auch Funktionen zur einfachen Nutzung des Shadow-Mechanismus. Natürlich können auch beliebige andere MQTT-Bibliotheken mit TLS-1.2-Unterstützung benutzt werden (**Bild 2**).

Beliebige IoT-Szenarien umsetzen

Mit FreeRTOS, Greengrass und IoT Core deckt AWS die komplette Kette vom kleinsten Mikrocontroller über Edge-Geräte bis zur Cloud-Lösung ab und hilft damit Entwicklern, beliebige IoT-Szenarien umzusetzen. Alle Bausteine können unabhängig voneinander verwendet und miteinander oder mit anderen Services kombiniert werden, da sie auf offenen Protokollen basieren.

AWS erweitert sein IoT-Portfolio fortlaufend: Auf der Hannover Messe veröffentlichte der Cloud-Anbieter mit AWS IoT Analytics eine vollständig verwaltete Plattform für die Datenanalyse von IoT-Daten. Mit AWS IoT Device Defender steht ein Werkzeug für die automatisierte Überwachung von Sicherheits-Richtlinien bei IoT-Geräten bereit. AWS IoT 1-Click reduziert die Komplexität von IoT auf das einfache Drücken eines Knopfes und eine daran verknüpfte AWS-Lambda-Funktion für bestimmte Geräte in vereinfachten IoT-Szenarien. Und wer kein solches Gerät hat, kann mit dem AWS IoT Button einen „IoT-Taster" bei Amazon bestellen und sofort mit AWS IoT beginnen. *cd*

Constantin Gonzalez
ist Principal Solutions Architect bei AWS.
glez@amazon.de

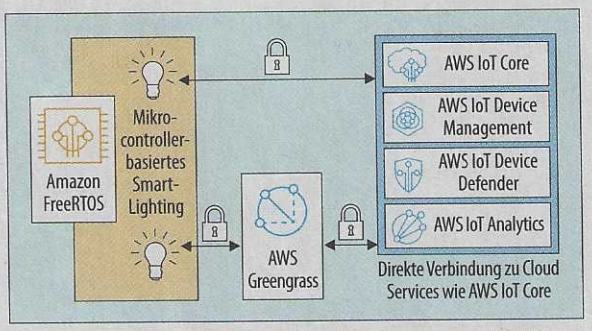

Bild 2: Zusammenspiel von a:FreeRTOS, AWS Greengrass und AWS IoT Services an einem Smart-Lighting-Beispiel. (Quelle: AWS)