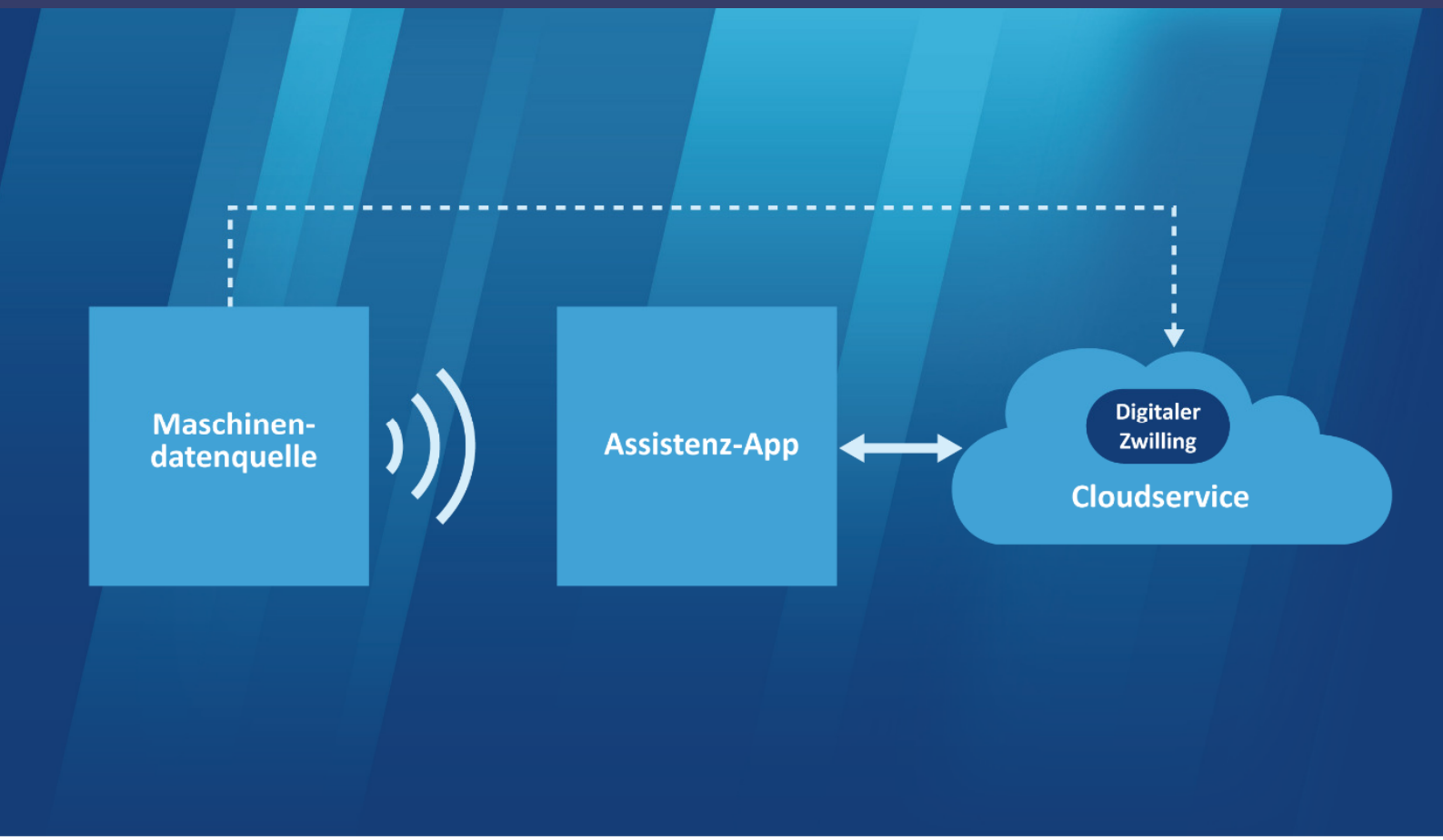

Whitepaper

Virtuelle IoT-Serviceassistenten

Hochqualifizierte Online-Hilfe im Störungsfall

Von Klaus-Dieter Walter

SSV Software Systems GmbH

Revision 1.0, 08/2022, kdw

SSV Software Systems GmbH

Dünenweg 5
D-30419 Hannover

Tel.: +49 (0)511/40 000-0
Fax: +49 (0)511/40 000-40

E-Mail: sales@ssv-embedded.de

Web: www.ssv-embedded.de
Social: www.linkedin.com/company/ssv-software-systems

Zusammenfassung

Die durchgängige Vernetzung im Internet der Dinge ermöglicht automatisierte virtuelle Serviceassistenten als zusätzliche Produkteigenschaft und darauf aufbauend wertvolle neue Kommunikationsmöglichkeiten zwischen Anbieter und Anwender. Sie helfen Produktherstellern und Anlagenbauern nicht nur bei der Umsetzung von Pay-per-Use-Geschäftsmodellen, sondern verbessern in jedem Fall die Kundenzufriedenheit.

Einleitung

Jeder von uns kennt die folgende Situation: Man steht vor einer Maschine, die nicht einwandfrei funktioniert. Eine rote Leuchte blinkt und in einem relativ kleinen LCD ist eine Meldung der Kategorie „Fehler 4711 …“ zu lesen. Was genau ist zu tun? Sehr häufig ist der Weg zur Antwort recht mühevoll. Auf jeden Fall hat ein solches Erlebnis einen nicht unerheblichen Einfluss auf die Kundenzufriedenheit. Deshalb sollten Maschinenbauer in allen Branchen zielgerichtete Maßnahmen anwenden, um negative Kundenerfahrungen auf Grund von Maschinenstörungen und ähnlichen Zuständen weitestgehend zu vermeiden.

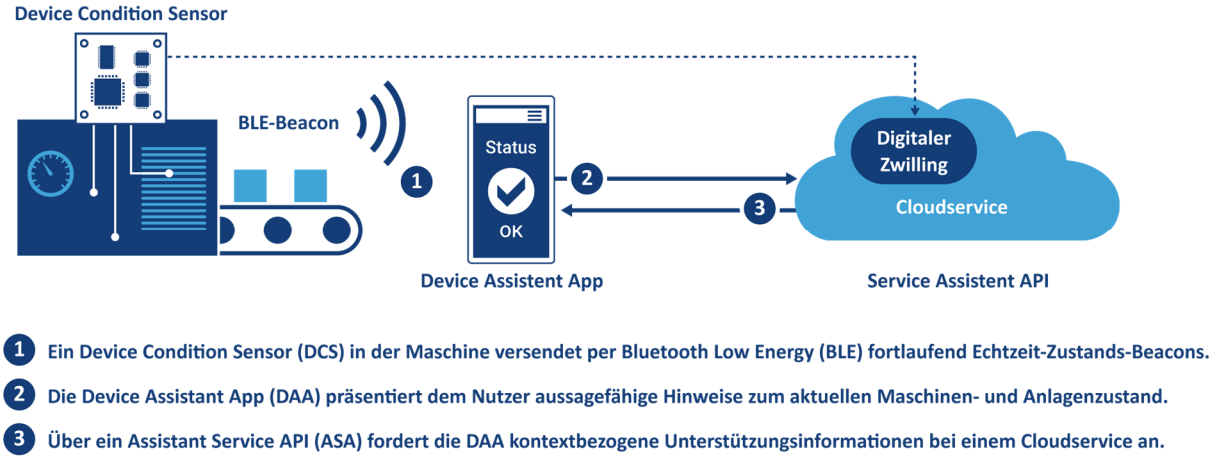

Abbildung 1: Ein virtueller IoT-Serviceassistent entsteht durch das Zusammenwirken der drei Funktionseinheiten Device Condition Sensor, Device Assistent App und einem Cloudservice.

Eine Bluetooth Low Energy (BLE)-Anwendung mit entsprechenden Unterstützungsfunktionen hilft dabei, auch in diesem Bereich die Wettbewerbsfähigkeit auszubauen. Abbildung 1 illustriert die Zusammenhänge: Eine Maschine versendet periodisch BLE-Beacons mit dem aktuellen Maschinenzustand. Ein Smartphone in Funkreichweite mit der dazu passenden App kann die Meldungen empfangen und an Hand eines an der Maschine angebrachten QR-Codes oder des BLE-RSSI-Werts feststellen, ob sich der Benutzer im Nahbereich des Beacon-Senders befindet.

Ist das der Fall, wird die App in den Vordergrund gebracht, der Zustands-Beacon ausgewertet und eine deutlich informativere Meldung mit Hintergrundinformationen und Verhaltenshinweisen auf dem Smartphone visualisiert. Gleichzeit fragt die App, ob ein Fernsupport (Remote Assistance) benötigt wird. Falls der Benutzer diese Unterstützung wünscht, übermittelt die App Maschinenzustandsdaten an einen Cloudservice. Von dort kommt dann nach einer Datenanalyse die gewünschte Hilfe. Bei Bedarf wird sogar ein digitaler Zwilling einbezogen, um die Ursachenforschung zu vertiefen.

Die Umsetzung einer solchen Lösung würde auf Seiten der Maschine lediglich Mehrkosten von wenigen Euro für ein IoT-Bluetooth-SoC (SoC = System-on-Chip) oder ein hochintegriertes BLE-SiP (SiP = System-in-Package) plus Antenne sowie einige weitere Bauteile verursachen. Die Anbindung an die jeweilige Maschinensteuerung wäre über einfache serielle oder parallele Schnittstellen möglich.

Details der Funktionseinheiten

Ein **Device Condition Sensor (DCS)** bildet die Datenquelle eines virtuellen Serviceassistenten. Er wird direkt in die Maschine bzw. Anlage integriert, beinhaltet zumindest ein Sensorelement und/oder eine Verbindung zur lokalen Maschinen- bzw. Anlagensteuerung und versorgt den digitalen Zwilling mit Daten. Darüber hinaus besitzt ein DCS einen Bluetooth-Transceiver, um einen BLE-Beacon zu versenden. Hardwaretechnisch wird ein DCS als ultrakompaktes SoC- oder SiP-Modul mit integrierter Antenne und anwendungsspezifischer Sensorik konzipiert. Es lässt sich herstellerseitig in jede Maschinensteuerelektronik integrieren und verursacht nur geringe Mehrkosten.

Eine **Device Assistant App (DAA)** dient als Schnittstelle zwischen dem DCS in der Maschine bzw. Anlage und einer Person vor Ort. Die DAA präsentiert nach einer erfolgreichen Datenauthentifizierung zunächst einmal die im BLE-Zustands-Beacon enthaltenen Informationen. Des Weiteren wird durch die Nutzung eines Cloudservice ein erweiterter Zustandskontext hergestellt; beispielsweise durch Diagramme, die bestimmte Maschineneigenschaften über einen längeren Zeitraum visualisieren. Über die DAA-Rückkopplung zum Cloudservice lassen sich aussagefähige Informationen zum aktuellen Zustand von Maschinen und Anlagen mit weitreichenden Wartungs- und Instandsetzungshinweisen an den Benutzer weitergeben. Per DAA sind auch virtuelle und physische Wartungs- und Instandsetzungs- sowie Online-Chat-Termine mit externen Experten reservierbar. Bei Bedarf kann der DAA-Nutzer auch Echtzeit-Fernzugriffe durch hochspezialisierte Servicefachkräfte auf eine Maschinensteuerung autorisieren.

Beim **Assistance Service API (ASA)** handelt es sich um einen Cloudservice für kontextbezogene Datenanalysen zum Maschinen- bzw. Anlagenzustand sowie weitere Unterstützungsfunktionen, die von externen Endgeräten verschiedener Teilnehmer über ein entsprechendes Application Programming Interface (API) angefordert werden. Der Input für diesen Cloudservice basiert auf einem BLE-Zustands-Beacon, der den Echtzeitzustand einer externen Funktionseinheit beschreibt. Für die jeweilige Datenanalyse stehen dem ASA verschiedene Methoden mit und ohne individuelle Device-Historie zur Verfügung. Im einfachsten Fall wird ein Zustands-Token lediglich über einen Entscheidungsbaum klassifiziert, um die Wartungs- und Instandsetzungshinweise für die Antwort an die DAA zu erzeugen. Für anspruchsvolle Anwendungen werden die Datenbestände eines digitalen Zwillings hinzugezogen, um über künstliche neuronale Netzwerke die Maschinen- und Anlagen-zustände so genau wie möglich zu bestimmen und die individuellen Besonderheiten einer Maschine bzw. die Produktlebenszyklusphase in den Handlungsempfehlungen der Antwort an die DAA zu berücksichtigen.

Verbund aus drei Softwarewelten

Die drei Funktionseinheiten eines intelligenten Serviceassistenten sind hinsichtlich der zu entwickelnden Software in völlig unterschiedlichen Welten verankert. Ein Device Condition Sensor besteht in einfachen Anwendungen lediglich aus einem Single-Chip-Microcontroller sowie einem MEMS-Inertial-Sensorelement. Eine solche Hardwareplattform lässt sich in die Kategorie „Deeply Embedded“ einordnen. Die für den Betrieb erforderliche Software wird primär in C/C++ erstellt und im integrierten Flash-Speicherbereich des Mikrocontrollers abgelegt.

Die Device Assistant App wird in den meisten Anwendungen durch eine Smartphone-App für Android und iOS gebildet. Sie wird zum Beispiel mit den dafür vorgesehenen Plattform-SDKs und jeweils einem separaten Quellcode für jedes Betriebssystem entwickelt. Alternativ ist auch ein HTML5-Cross-Plattform-Projekt mit Werkzeugen wie Apache Cordova, AngularJS oder Ionic möglich. Als Output der Smartphone-App-Entwicklung wird auf jeden Fall entweder ein APK (Android Package) oder IPK (iOS App Store Package) erzeugt und über die jeweiligen App Stores verbreitet.

Alternativ zur DAA für Smartphones ist auch eine HTML5-Webseite möglich, die durch einen JavaScript-Code in die Lage versetzt wird, BLE-Zustands-Beacons zu empfangen, über das Assistance Service API mit kontextbezogenen Zusatzinformationen zu ergänzen und das alles dem Webseitennutzer zu präsentieren. Solche DAA-Desktop-Anwendungen werden beispielsweise durch Werkzeuge wie dem Electron-Framework unterstützt.

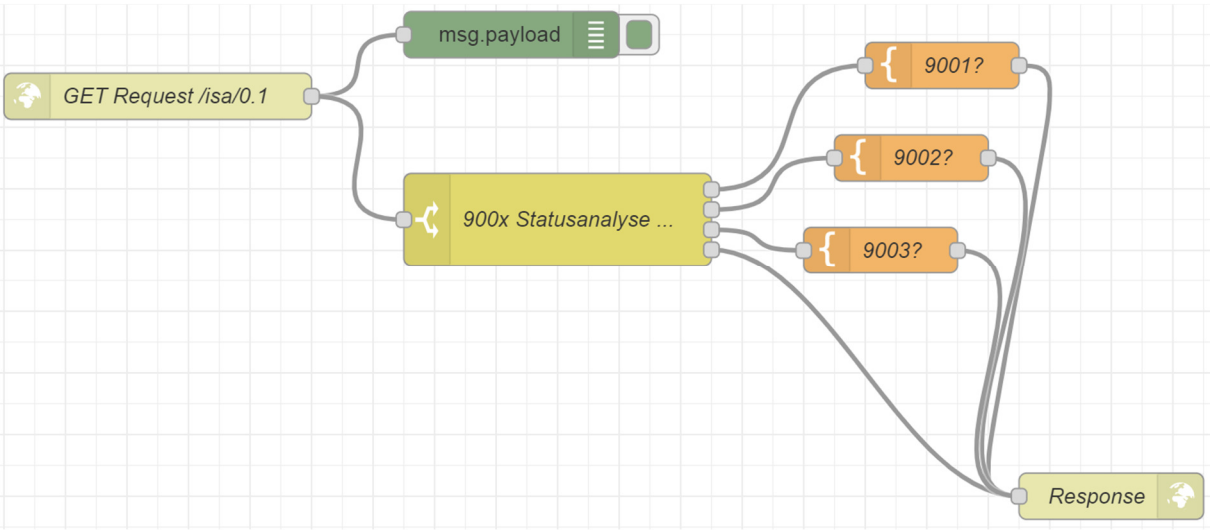

Abbildung 2: Für Testzwecke oder relativ einfache Anwendungen lässt sich ein Assistance Service API auch mit Hilfe eines Node-RED Flows realisieren. Die Abbildung zeigt ein Beispiel für ein Sensorsystem zur Überwachung einer Maschinenkomponente mit drei Zuständen, die per Machine Learning automatisch erkannt werden.

Das Assistance Service API wird typischerweise auf einer Serverplattform ausgeführt. Die zuständigen Entwickler werden daher hochentwickelte Programmiersprachen wie beispielsweise Python oder JavaScript mit einer Node.js-Laufzeitumgebung für die Implementierung der REST-basierten Schnittstellen favorisieren. Ein ASA-Code lässt sich gezielt für eine bestimmte Cloudumgebung (z. B. Amazon-AWS oder Microsoft-Azure) oder universell für Docker-Container in Docker-Laufzeitumgebungen entwickeln. Mit einem solchen Docker-Container lassen sich Assistance Service API-Applikationen nahezu plattformunabhängig weitergeben, um sie auf Rechnern mit 32- bzw. 64-Bit-AMD/Intel-, Arm- oder RISC-V-Prozessoren sowie Linux-, Microsoft-Windows-, macOS- und anderen UNIX-ähnlichen Betriebssystemen zu betreiben.

Die Bereitstellung und Installation eines Docker-basierten ASA erfolgt über Registry-Server und Repositories. Die zu installierende Applikation wird dabei als Docker-Container-Image wahlweise auf einem Internet-Server (z. B. Docker Hub) oder einer lokalen Serveranwendung zur Verfügung gestellt.

Einzeln wartbar, aber aufwendige Tests erforderlich

Um die einzelnen Softwarebausteine einer Serviceassistentenanwendung über den gesamten Lebenszyklus an die jeweiligen Anforderungen anzupassen, werden sogenannte DevOps eingesetzt. Hinter diesem aus „Dev“ (Development, Entwicklung) und „Ops“ (Operations, Vorgänge) zusammengesetzten Begriff verbirgt sich eine sehr effektive Vorgehensweise zur Softwareentwicklung und Wartung. DevOps vereinen Menschen, Methoden, Prozesse und Technologien, um kontinuierlich hochwertige Produkte und Lösungen zu schaffen und in der Praxis zu betreiben.

Die acht typischen DevOps-Phasen werden vielfach als Endlosschleife dargestellt (oberer Teil der Abbildung 3), die vom zuständigen Produktmanagement, dem Entwicklerteam und den Anwendungsverantwortlichen als Prozesskette immer wieder durchlaufen werden. In diesem Konstrukt ist jeweils eine CI- (Continuous Integration, Softwarecodierungs- und Änderungsprozess) und CD- (Continuous Delivery, Software-Auslieferungsprozess) Aktivitäten-Pipeline enthalten. Alle anfallenden Aufgaben werden dabei von einem Entwickler- und einem Anwendungsteam bearbeitet.

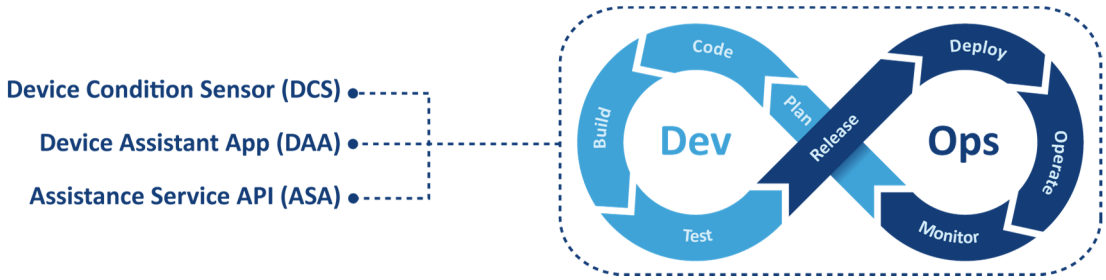

Abbildung 3: Die Funktionseinheiten eines virtuellen Serviceassistenten sind softwaretechnisch in völlig unterschiedlichen Welten verankert. Der Entwicklungsprozess basiert daher auf DevOps-Konzepten.

DevOps wurden in der IT-Welt entwickelt und werden dort auch seit vielen Jahren in unzähligen Smartphone-App- und Cloudanwendungen eingesetzt. Die DevOps-Methoden lassen sich unter bestimmten Voraussetzungen aber auch für vernetzte Embedded Systeme, wie beispielsweise das SoC- oder SiP-basierte Sensorik-Subsystem eines intelligenten Serviceassistenten, anwenden. Eine besondere Herausforderung ist dabei allerdings der Unterschied zwischen der Entwicklungsumgebung (in der die Embedded-Software erstellt und getestet wird) und der sogenannten Produktionsumgebung – also das Umfeld, in dem diese Software in der Praxis zum Einsatz kommt.

Kundenzufriedenheit ständig verbessern

Kundenzufriedenheit ist ein abstraktes Konstrukt, um die Differenz zwischen Kundenerwartung und Bedürfnisbefriedigung zu beschreiben. In der Welt der Maschinen und Anlagen gibt es nach einer erfolgreichen Inbetriebnahme zwei besonders kritische Phasen mit erheblichem Einfluss auf die Zufriedenheit: Zum einen kann z. B. die Leistung einer Maschine unzureichend sein; die möglichen Ursachen dafür sind im Verkaufsprozess, bei der Planung, der Vor-Ort-Montage oder sogar in der Qualifikation der beteiligten Fachkräften zu suchen.

Zum anderen können sich Betriebsstörungen und Fehlerzustände sehr negativ auf die Zufriedenheit eines Kunden auswirken. Manchmal liegt es einfach daran, dass im Fehlerfall kein qualifizierter Expertenrat zur Verfügung steht, der wirklich weiterhilft. Dieses Problem lässt sich mit einem virtuellen IoT-Serviceassistenten in jedem Fall lösen, zumal sich in die App umfangreiche Zusatzfunktionen – vom Online-Voice-Chat-Client bis zur Mixed Reality-Ansicht – integrieren lassen.

Der Autor

Klaus-Dieter Walter ist Geschäftsführer bei der SSV Software Systems GmbH in Hannover und ist durch zahlreiche Vorträge auf internationalen Veranstaltungen, Seminaren, Workshops und Artikel in Fachzeitschriften bekannt. Er hat vier Bücher zu den Themen Embedded Linux, Embedded Internet und ARM-basierte Mikrocontroller veröffentlicht.

Anfang 2007 hat Klaus-Dieter Walter aktiv an der Gründung der M2M Alliance e.V. in Aachen mitgewirkt und war langjähriges Mitglied des Vorstands. Er war außerdem viele Jahre Vorstandsmitglied des VHPready Industrieforums zur Standardisierung der Kommunikation in virtuellen Kraftwerken. Neben seiner Tätigkeit als Geschäftsführer ist Klaus-Dieter Walter Mitglied der Expertengruppe Internet der Dinge - Digitaler Gipfel Deutschland.