# Entwurf einer Softwarearchitektur für dezentrale mikrocontrollerbasierte Steuerungssysteme

## Komplexitätsbeherrschung auf Steuerungsebene

M.Sc. **J. Schmidt**, Prof. Dr.-Ing. **M. Weyrich**,
Institut für Automatisierungs- und Softwaretechnik, Universität Stuttgart

**Kurzfassung**

Steuerungssysteme für die Anlagenautomatisierung werden zunehmend dezentral realisiert. Zukunftsfähige Konzepte für solche Systeme müssen vor dem Hintergrund von Industrie 4.0 und dem Internet der Dinge betrachtet werden. Kürzlich veröffentlichte Referenzarchitekturen weisen den Weg für zukunftsorientierte Steuerungssysteme. Gleichzeitig stellt die Referenzarchitektur AUTOSAR eine etablierte Architektur für dezentrale Systeme in der Automobilindustrie dar, die vor ähnlichen Herausforderungen stand, wie die Automatisierungstechnik heute und somit als Vorbild dienen kann.

In diesem Beitrag wird die AUTOSAR-Architektur betrachtet und teilweise mit anderen Architekturen verglichen. Auf dieser Basis wird ein Vorschlag zur Umsetzung auf Steuerungsebene entwickelt, um die Komplexität der Steuerungssoftware für dezentrale mikrocontrollerbasierte Systeme beherrschbar zu machen.

**Abstract**

In future, control systems for plant automation will be realized more and more decentralized. Sustainable concepts for such systems must be considered regarding Industry 4.0 and the internet of things. The recently published reference architectures of the two approaches give important impetus for sustainable control systems. At the same time the reference architecture AUTOSAR represents a successful architecture for decentralized systems in the automotive industry, which had years ago the same challenges as the automation industry today.

In this paper the AUTOSAR architecture is considered and partially compared with the other architectures. Upon those considerations a proposal for a realization on control level is derived. Based on the concept the complexity of decentralized microcontroller based control systems can be mastered.

## 1. Zielsetzung

Zukünftige Anlagensteuerungen werden zunehmend dezentral realisiert, wozu bestehende Hierarchien aufgebrochen werden. Dabei gewinnen Mikrocontroller zunehmend an Bedeutung, wenn es darum geht eine zentrale Steuerung durch viele dezentrale Einheiten zu ersetzen.

Im Gegensatz zu PC-Systemen oder SPS, die eine hohe Rechenleistung aufbringen müssen, um eine zentrale Steuerung zu realisieren, sind bei einer großen Anzahl von dezentralen Steuerungseinheiten andere Faktoren wichtiger. Da nur ein kleinerer Teil der zu steuernden Prozesse von einer Einheit abgedeckt wird, spielt die Rechenleistung keine zentrale Rolle. Stattdessen müssen die Steuerungen kostengünstig, echtzeitfähig und hochgradig kommunikationsfähig sein.

Heutige Mikrocontroller-Systeme können aufgrund der hohen Stückzahlen kostengünstig produziert werden und sind in der Regel echtzeitfähig. Die Herausforderung liegt in der Kommunikation, da sich dezentrale Systeme koordinieren müssen. Im Hinblick auf Kommunikationstechnologien für eine solche Machine-to-Machine-Kommunikation liefert [1] einen guten Überblick über Wireless-Technologien und Kommunikationsprotokolle.

Die Entwicklung von kostengünstigen mikrocontrollerbasierten Steuerungssystemen birgt jedoch neue Herausforderungen. Die Komplexität dezentraler Systeme ist deutlich größer, als die Komplexität zentraler Systeme mit äquivalenten Steuerungsaufgaben. Die Komplexität wird hierbei durch zwei wesentliche Faktoren bestimmt: Zum einen steigt die Anzahl von Steuerungskomponenten zum anderen nimmt die Kommunikation zur dynamischen Koordination der Komponenten zu.

Zunächst wird eine größere Anzahl an Steuerungen benötigt, denn jeder Teilprozess, der dezentral gesteuert werden soll, benötigt eine Steuerungseinheit. Dadurch steigt der Entwicklungsaufwand für Hard- und Software.

Neben der Entwicklung der einzelnen Steuerungen, stellt sich die Frage, wie diese koordiniert werden. Eine zentrale Koordinierung würde dem Gedanken der Dezentralität widersprechen. Die „Intelligenz“, die für die Koordination der dezentralen Einheiten notwendig ist, muss also ebenfalls verteilt werden.

Für die Komplexitätsbeherrschung solcher Systeme wurden bereits verschiedene Referenzarchitekturen entwickelt. Im Folgenden werden drei dieser Architekturen gegenüber gestellt und analysiert. Daraus werden Einsichten für eine Komplexitätsbeherrschung auf Steuerungsebene abgeleitet und ein Architekturentwurf vorgestellt, der den Entwicklungsprozess dezentraler Systeme auf Steuerungsebene vereinfachen soll.

Mithilfe dieser Architekturbetrachtung wird ein Vorschlag zur Umsetzung von Software für dezentrale Mikrocontrollersysteme unterbreitet.

## 2. Analyse relevanter Architekturen

Für eine zukunftsorientierte mikrocontrollerbasierte Softwareentwicklung sind drei Architekturen besonders wesentlich:

- Kürzlich wurde ein Modell einer Referenzarchitektur für Industrie 4.0 [2] von ZVEI veröffentlicht. Diese Architektur für das Zukunftsprojekt Industrie 4.0 wird maßgeblich beeinflussen, wie Wertschöpfung in der Industrie zukünftig funktionieren wird.
- Die zweite Architektur, die hier Erwähnung findet, ist eine Referenzarchitektur des Internet der Dinge (IoT). Sie beschreibt die Vernetzung von Alltagsgegenständen über das Internet, um Anwender besser zu unterstützen. [5]
- Die dritte Architektur ist die in der Automobilindustrie erfolgreiche Referenzarchitektur AUTOSAR. Diese ist relevant, da Steuergeräte im Automobilbereich schon immer dezentral entwickelt wurden und bereits vor einigen Jahren ähnliche Probleme mit der Komplexität gelöst wurden, wie sie jetzt bei der Automatisierungstechnik auftreten. [6]

### 2.1 Modell der Referenzarchitektur für Industrie 4.0

Das Modell einer Referenzarchitektur für Industrie 4.0 (RAMI4.0) beschreibt in Verbindung mit dem Konzept der Industrie 4.0-Komponente, die sich über Bekanntheitsgrad und Kommunikationsfähigkeit definiert, wie Wertschöpfung im Kontext von Industrie 4.0 möglich sein wird. Die Referenzarchitektur berücksichtigt mehrere Dimensionen der Automatisierungstechnik:

- Im „Value Stream“ wird nach der Norm IEC 62890 das Life-Cycle-Management definiert.
- In den „Hierarchy Levels“ werden die Hierarchie-Ebenen vom Produkt über die Firma bis hin zur vernetzten Welt definiert. Grundlage dafür sind die IEC-Normen 62264 und 61512.
- Die dritte Dimension beschreibt ein weiteres Schichtenmodell, das orthogonal zu den anderen beiden Dimensionen steht.

In [3] wird ausführlich beschrieben, wie eine Industrie 4.0-Verbundanlage aussehen kann. Dabei ist die Vernetzung von heterogenen Industrieanlagen und der Logistik von Bedeutung. Für die Steuerungsebene entwickeln die Control- und Field-Devices besondere Bedeutung in der Achse Hierarchy Levels. Diese müssen, sollten sie „I4.0-konform“ entwickelt werden, den Anforderungen einer I4.0-Komponente entsprechen.

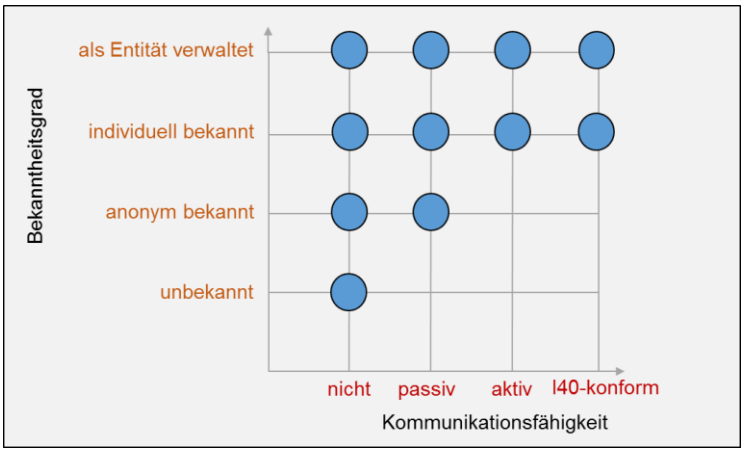


*Abbildung 1: I4.0-Komponente nach VDI [4]*

Dabei stellt die Industrie 4.0-Komponente zwei wesentliche Anforderungen an zukunftsfähige Steuerungssysteme: Eine I4.0-konforme Kommunikationsfähigkeit und einen als Entität verwalteten Bekanntheitsgrad im Informationssystem. Abbildung 1 zeigt die nach VDI festgelegten Klassifikationsstufen, die eine besondere Relevanz erhalten.

### 2.2 Internet of Things Referenzarchitektur

Das Internet der Dinge entwickelt sich parallel zum Zukunftsprojekt Industrie 4.0. Hierbei werden Alltagsgegenstände an das Internet angebunden, um den Nutzer in seinem Alltag zu unterstützen.

Eine veröffentlichte Referenzarchitektur für das Internet der Dinge [5] wird ebenfalls als relevant betrachtet, da sie ähnliche Ziele wie die Industrie 4.0 Architektur verfolgt.

Das sogenannte IoT-Domain Model besteht aus mehreren Sub-Modellen. Für die Realisierung einer Steuerung ist das IoT-Functional-Model, in das zwei weitere Sub-Modelle integriert sind, relevant: Das IoT Communication Model und das IoT-Trust, Security & Privacy Model.

Abbildung 2 stellt das Functional-Model dar und zeigt, wie Steuerungssysteme funktional realisiert werden und welche Voraussetzungen berücksichtigt werden müssen, wenn die Steuerung an das Internet angebunden wird.

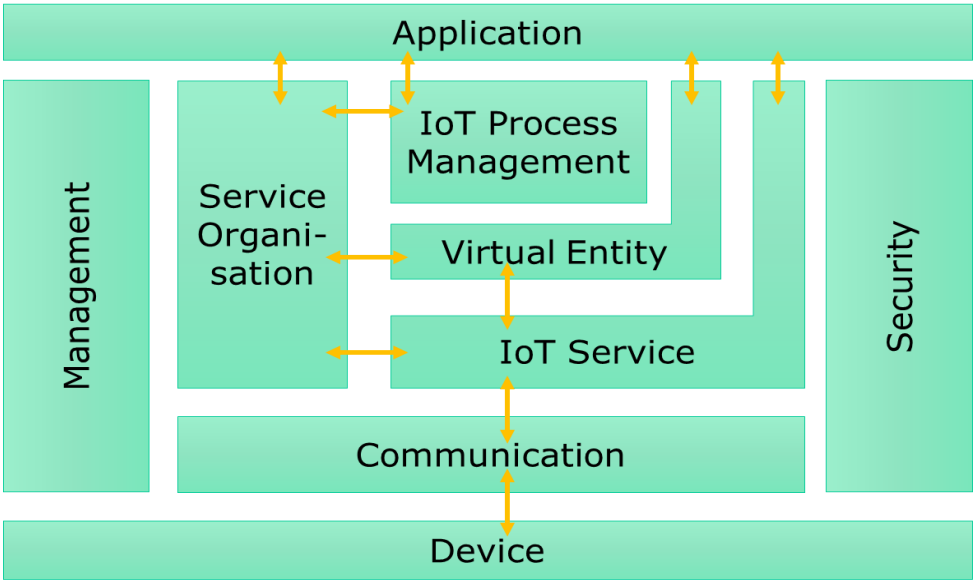


*Abbildung 2: IoT Functional Model nach [5]*

Neben den Blöcken, die Management- und Sicherheitsfunktionen bereitstellen, besteht das funktionale Modell im Wesentlichen aus einer Schichtenarchitektur, die über verschiedene Ebenen hinweg die Hardware abstrahiert. Die Blöcke, die in diesem Modell als Functional Groups (FG) bezeichnet werden, definieren grundlegende Funktionen, die für eine Realisierung notwendig sind. Ein Beispiel für eine solche Gruppe ist die FG Communication, da ohne Kommunikationsschnittstellen eine Internetanbindung nicht möglich ist.

### 2.3 AUTOSAR

Die Automotive Open System Architecture – AUTOSAR – [6] ist die etablierte Referenzarchitektur für Steuergerätesoftware in der Automobilindustrie. Steuergeräte (ECUs) und die zugehörige Software im Automobil wurden schon immer dezentral entwickelt. Dieses Vorgehen ist historisch gewachsen, bedingt durch neue Funktionen im Fahrzeug, die zunächst dezentral durch ein eigenes Steuergerät realisiert wurden. Ein Automobil kann also als dezentral gesteuertes automatisiertes System betrachtet werden.

Funktional betrachtet ist die Steuergeräte-Software in drei Ebenen gegliedert. Die unterste Ebene stellt die Basissoftware (BSW) dar. Grundlegende Funktionalitäten werden hier zur Verfügung gestellt, wobei in höheren Schichten der Abstraktionsgrad zunimmt. Der modulare Aufbau der Basissoftware ermöglicht die Wiederverwendung einzelner Module. Die zweite Komponente ist die Laufzeitumgebung. Diese dient zum einen als Abstraktion der Schnittstellen für die Applikation und zum anderen als Middleware, über die Softwarekomponenten auch Steuergeräte- und Busübergreifend kommunizieren können.

Als dritte Ebene bleibt die Applikationsschicht, in der die Anwendung in Form von Softwarekomponenten realisiert wird.

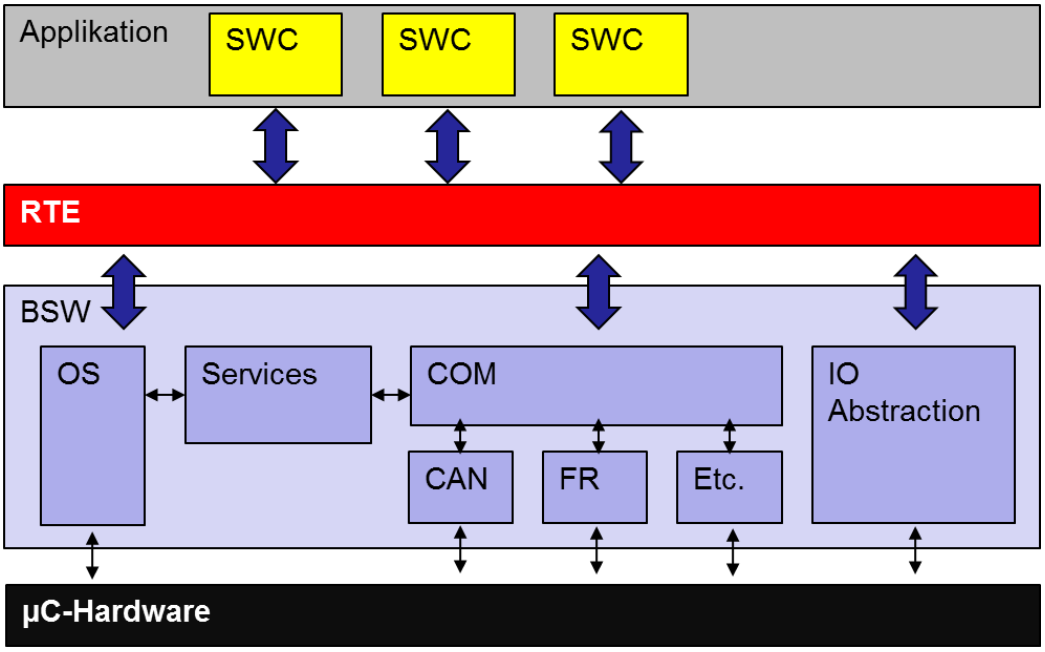


*Abbildung 3: AUTOSAR Grundlegende Architektur, Quelle [6]*

Aufgrund der Laufzeitumgebung, die als Middleware fungiert, können Softwarekomponenten steuergeräteunabhängig entwickelt werden. Durch ein anschließendes Mapping wird jede Softwarekomponente dem passenden Steuergerät zugeordnet. Die Schnittstellenabstraktion ermöglicht eine modellbasierte Entwicklung, sodass die gesamte Entwicklung auf eine zuverlässigere Modellierung der Applikation und Konfiguration wiederverwendbarer Basissoftware umgestellt werden kann, die weniger fehleranfällig ist.

## 3 Ableitung von Konzepten zur Komplexitätsbeherrschung auf Steuerungsebene

Eine zukunftsfähige Steuerungssoftware für dezentrale Mikrocontroller-Systeme sollte im Sinne einer Industrie-4.0-Komponente in Industrie 4.0-Systeme integrierbar sein. Daher ergeben sich aus der I4.0-Architektur die beiden wesentlichen Anforderungen der Bekanntheit und der Kommunikationsfähigkeit.

Um die Komplexität beherrschbar zu machen, kann von den anderen beiden Architekturen das Konzept der Wiederverwendung genutzt werden. Sowohl IoT als auch AUTOSAR nutzen eine Schichtenarchitektur als Basis, die grundlegende Funktionalitäten bietet.

### 3.1 Wiederverwendung von modularisierter Basissoftware

Den ersten grundlegenden Aspekt stellt die Widerverwendung von Basissoftware-Modulen dar. Die Basissoftware der oben eingeführten Architekturen kann nicht ohne Änderungen

übernommen werden, da sie domänenspezifisch für Automotive-Steuergeräte entwickelt wurden, oder die Steuerungsebene nicht im Detail berücksichtigen. Auch das IoT-Functional-Model entspricht nicht den Anforderungen, da sich die dezentralen Einheiten zwar koordinieren müssen aber nicht immer an das Internet angebunden werden. Trotzdem sind einige Software-Module bei einer dezentralen Anlagenautomatisierung für jede Steuerungskomponente zwingend erforderlich. Dazu gehören mindestens ein echtzeitfähiges Betriebssystem, eine Abstraktion der Hardware-Ansteuerung (z.B. GPIO) und ein Kommunikationsstack. Eine Schnittstellendefinition und modulare Softwareentwicklung ist hier der erste Schritt, die Komplexität des Gesamtsystems zu reduzieren.

### 3.2 Middleware mit standardisierten Schnittstellen

Die Laufzeitumgebung (RTE) bei AUTOSAR bietet zwei essentielle Funktionen, die auch hier wieder aufgegriffen werden sollen. Zum einen müssen Applikationsteile kommunizieren können, zum anderen soll auch die Option bestehen, modellbasiert zu entwickeln. Die Voraussetzung für beides kann auch in einer neuartigen Architektur genutzt werden. Sollten sich für eine zukünftige Middleware Protokolle wie Message Queue Telemetry Transport (MQTT), OPC Unified Architecture (OPC UA) oder andere durchsetzen, bleibt die RTE trotzdem erhalten, koordiniert dann aber nicht mehr die Inter-ECU-Kommunikation, sondern nur noch die Intra-ECU-Kommunikation und stellt die Schnittstellenabstraktion für die Modelle zur Verfügung.

### 3.3 Modellbasierte Applikationsentwicklung

Eine modellbasierte Entwicklung von Anwendungen in der Applikationsschicht ist in der Automobilindustrie Stand der Technik. OEMs liefern Modelle an Zulieferer, die dann aus den Modellen Code generieren. Diese Technologie bietet enorme Vorteile gegenüber der klassischen Softwareentwicklung, da der OEM bereits frühzeitig modellbasiert testen kann und so Anforderungen validieren kann, bevor die Steuergeräte geliefert werden. Im Gegensatz zur konventionellen Implementierung wird eine bessere Qualität bei geringerem Aufwand erreicht. Diese Technologie soll auch für mikrocontrollerbasierte Steuerungen beibehalten werden. In [7] wurde die Bedeutung der modellbasierten Entwicklung in einer Gegenüberstellung verschiedener Realisierungs-Ansätze für Industrie 4.0 bereits hervorgehoben. Wesentliche Voraussetzung sind standardisierte Schnittstellen für die Applikation, die modelliert werden soll. In der Automobilindustrie wird die Voraussetzung durch AUTOSAR erfüllt. In der hier vorgestellten Architektur wird sie durch den im vorherigen Abschnitt beschriebenen Transfer

der Laufzeitumgebung, die eine einheitliche Schnittstellenabstraktion bereitstellt, auf die neue Architektur erfüllt.

### 3.4 Berücksichtigung der Konzepte bei den Referenzarchitekturen

Die vorgestellten Architekturen haben alle das Ziel, Komplexität beherrschbar zu machen, indem ein Rahmen für die Entwicklung verschiedener Systeme vorgegeben wird.

Aufgrund der unterschiedlichen Zielsetzungen ergeben sich jedoch einige grundlegende Unterschiede zwischen den dargestellten Architekturen. Der wesentliche Unterschied von AUTOSAR zu den anderen beiden Architekturen ist, dass AUTOSAR im Wesentlichen die Steuerungsebene betrachtet, während die I4.0 und IoT Architekturen vor allem auf die Koordination zwischen Komponenten und die damit erreichbare Wertschöpfung eingehen. Die IoT-Architektur bietet allerdings eine funktionale Sicht, die der Basissoftware von AUTOSAR nicht unähnlich ist. Die I4.0-Architektur mit dem Ebenenmodell benötigt die I4.0-Komponente, die durch Kommunikationsfähigkeit und Bekanntheitsgrad gekennzeichnet ist.

Die folgende Tabelle zeigt eine Übersicht über die Architekturen unter Berücksichtigung der oben dargestellten Konzepte zur Komplexitätsbeherrschung auf Steuerungsebene:

*Tabelle 1: Gegenüberstellung der Architekturen*

| **Konzept** | **AUTOSAR** | **Internet of Things** | **Industrie 4.0** |
|---|---|---|---|
| Strukturelle Verortung | Steuerungsebene | Steuerungsebene, Vernetzung über Internet | Mehrdimensionales, vielschichtiges Ebenenmodell |
| Middleware | Laufzeitumgebung koordiniert Kommunikation | Verschiedene M2M-Protokolle möglich | Verschiedene M2M-Protokolle in Verbindung mit industriellen Feldbussystemen |
| Wieder-verwendung | Modularer Aufbau der Basissoftware | Modulare Schichtenarchitektur in Funktionaler Sicht | Keine Vorgaben auf Steuerungsebene |
| Modellbasierte Applikations-Entwicklung | Standardisierte Schnittstellen[1] der RTE | Nicht vorgesehen | Keine standardisierten Schnittstellen auf Steuerungsebene |

[1] Voraussetzung für eine modellbasierte Applikationsentwicklung sind einheitliche Schnittstellen für die aus den Modellen generierte Software.

Die vorgeschlagenen Konzepte sind bei AUTOSAR umgesetzt und Stand der Technik in der Automobilindustrie. Da im Kontext von Industrie 4.0 auch heterogene Steuerungsnetzwerke vernetzt werden, gibt es keine konkreten Vorgaben für die Entwicklung einer Steuerungseinheit, es ist aber durchaus möglich und im Falle der modellbasierten Entwicklung gefordert [7], die beschriebenen Konzepte Industrie 4.0-konform umzusetzen.

## 4. Entwurf einer Architektur unter Berücksichtigung der Konzepte

Die Architektur, die für mikrocontrollerbasierte dezentrale Steuerungen realisiert werden soll, ist also deutlich an AUTOSAR angelehnt. Die wiederverwendbare, modularisierte Basissoftware liefert grundlegende Dienste. Eine Laufzeitumgebung liefert eine Abstraktion für die modellbasierte Entwicklung in der Applikationsschicht.

Abbildung 4 stellt auf der linken Seite den Grundaufbau für die Steuerungssoftware dar:

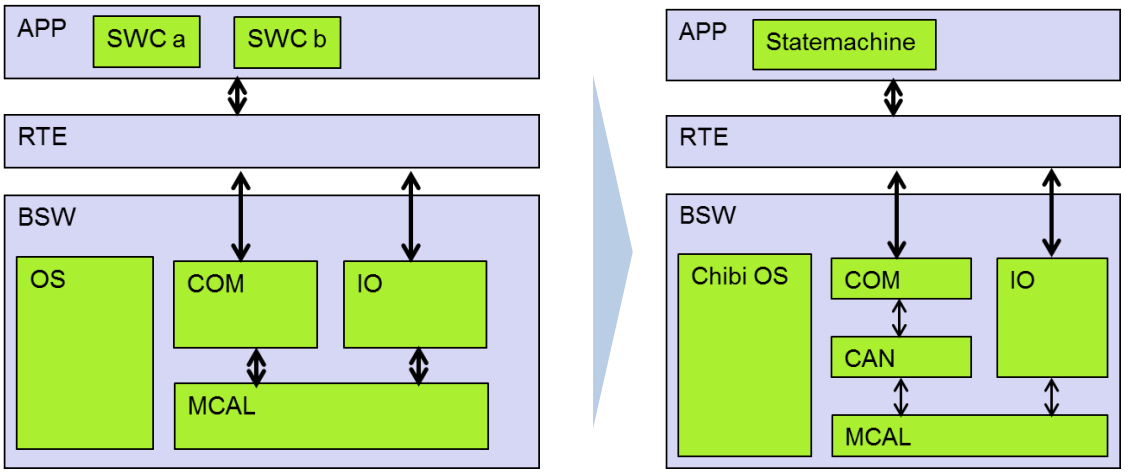


*Abbildung 4: Ansatz für Mikrocontroller-Steuerung*

Die Module der **Basissoftware** bieten grundlegende Funktionen:

- MCAL: Der Mikrocontroller Abstraction Layer ist grundsätzlich notwendig, um auf die Peripherie des Mikrocontrollers zugreifen zu können.
- OS: ein Echtzeitbetriebssystem verwaltet grundlegende Dienste, wie Tasks, Threads oder Interrupts.
- IO: Da es sich um eine Steuerung handelt, muss der Zugriff auf den Prozess möglich sein. Dies erfolgt über die Treiber im MCAL-Layer, die über das IO-Modul abstrahiert werden.
- COM: für die Koordination zwischen Steuerungseinheiten ist ein Kommunikationsstack erforderlich. Dieser wird hier als COM-Modul dargestellt. Er interagiert mit den Treibern der entsprechenden Bussysteme im MCAL-Cluster und abstrahiert die Busbotschaften nach oben hin für die Laufzeitumgebung bzw. die RTE.

Die Laufzeitumgebung **RTE** abstrahiert die Dienste IO und Kommunikation der Basissoftware und stellt der Applikationsschicht standardisierte Schnittstellen zur Verfügung. Die Applikationsschicht **APP** enthält Softwarekomponenten, die aus Modellen generiert werden und so die eigentliche Steuerung realisieren. Auf der rechten Seite von Abbildung 4 ist die Ausprägung der Architektur dargestellt, die in einem Prototyp realisiert wurde. Im Prototyp wird das schlanke Echtzeitbetriebssystem Chibi OS eingesetzt, sowie ein CAN-Stack realisiert. Als Anwendung wird eine in Matlab realisierte Statemachine genutzt.

Da eine so realisierte Steuerung den bereits hergeleiteten Anforderungen Kommunikationsfähigkeit und Bekanntheit im Sinne einer I4.0-Komponente entsprechen soll, muss der Kommunikationsstack COM näher betrachtet werden. Innerhalb des Kommunikationsstacks ist es notwendig, verschiedenartige Bussysteme zu integrieren. Wie bei AUTOSAR kann dieses Problem durch ein zusätzliches Modul PDU-Router gelöst werden, das die Botschaften verschiedenen Bussystemen zuordnet. Unterhalb des PDU-Routers können dann verschiedenartige Bussysteme integriert werden:

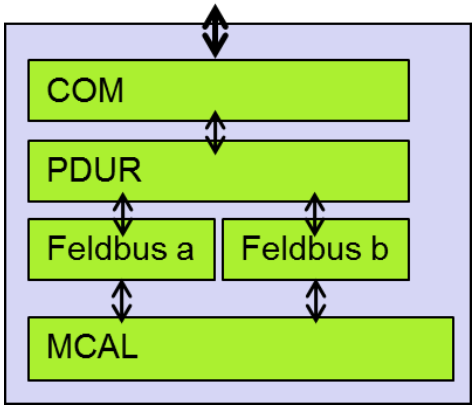


*Abbildung 5: Der Com-Stack*

Innerhalb der Feldbus-Schicht des Kommunikationsstacks ist können weitere Protokolle realisiert werden. So ist es beispielsweise möglich einen IP-Stack um die Protokolle TCP oder UDP zu erweitern. Auch eine Realisierung von feldbusunabhängigen Protokollen wie MQTT oder OPC UA [8] sind denkbar.

So werden dezentrale Steuerungseinheiten je nach Anforderungen über unterschiedliche Kommunikations-Netze verbunden. Die eigentliche Steuerung kann zunächst Hardware- und Feldbusunabhängig realisiert werden.

Die Kommunikationsfähigkeit kann I4.0-konform realisiert werden.

Es bleibt noch zu klären, ob eine solche Steuerungseinheit individuell oder als Entität bekannt sein kann. Für den Bekanntheitsgrad stellt dieser Architekturansatz keinen Lösungsvorschlag. Aktuell werden verschiedene Ansätze erforscht. Dazu gehören beispielsweise eine

diensteorientierte Architektur (SOA) [9] oder Softwareagenten [10]. Mit der hier vorgestellten Architektur lassen sich aufgrund des modularen BSW-Aufbaus grundsätzlich beide Konzepte realisieren.
Ein hoher Bekanntheitsgrad im Sinne der I4.0-Klassifikation als Maß der Vernetzung ist also nicht automatisch gegeben, aber umsetzbar.

## 5. Prototyp

Ein erster Prototyp wurde am IAS entwickelt, um zu zeigen, wie die Ansätze realisiert werden können. Als Betriebssystem wurde das schlanke Echtzeitbetriebssystem Chibi OS gewählt. Für die Kommunikation wurde ein CAN-Stack realisiert. Die MCAL-Schicht wurde vom Controllerhersteller mitgeliefert und in den Stack integriert. Aus einem über Matlab Stateflow erzeugten Zustandsautomaten wurde der Code für die Applikation generiert. Die Abstraktion für den so generierten Code über die RTE und der busunabhängige Teil des Kommunikationsstacks oberhalb des CAN waren für das Betriebssystem Chibi OS nicht verfügbar und mussten implementiert werden.
Als Controller wurden SPC-Evaluationsboards der Firma ST-Microelectronics gewählt. Die CPU dieser Boards wurde nach der Power-PC-Architektur entwickelt. Sie sind so leistungsfähig, dass der komplette Stack mit OS und anspruchsvolle Softwarekomponenten problemlos darauf lauffähig sind.
Der realisierte Anwendungsfall ist ein einfacher Zustandsautomat, der mit Matlab Stateflow modelliert wurde. Der Code für die Applikation auf dem Controller wurde mit dem Matlab Embedded Coder aus dem Modell generiert.
Für die weitere Evaluierung soll dieser Stack für die Steuerung einer Demonstrations-Anlage des IAS eingesetzt werden. Eine so entwickelte dezentrale Steuerung kann zum einen mit zentralen SPS-Steuerungen in Bezug auf Leistungsmerkmale und Entwicklungsaufwand verglichen werden. Zum anderen wurden bereits dezentrale Steuerungen ohne modellbasierte Entwicklung und Wiederverwendung realisiert. In einem Vergleich mit diesen Steuerungen kann der Nutzen dieses Stacks weiter evaluiert werden.

## 6. Zusammenfassung und Ausblick

In diesem Beitrag wurden zunächst drei wesentliche Referenzarchitekturen mit Relevanz für dezentrale mikrocontrollerbasierte Steuerungssysteme analysiert. Industrie 4.0 wird für zukünftige Automatisierungssysteme eine große Rolle spielen. Damit verwandt stellt der Trend des Internet der Dinge weitere wesentliche Aspekte einer solchen Architektur dar. AUTOSAR ist im Gegensatz zu den beiden relativ neuen Architekturen bereits erfolgreich etabliert und

zeigt wie die Entwicklung komplexer dezentraler Systeme beherrschbar gemacht werden kann.

Aus diesen Architekturen wurden Anforderungen für eine moderne dezentrale Steuerungsarchitektur abgeleitet. Eine solche Architektur, die die Anforderungen erfüllt, wurde vorgestellt und beschrieben.

Anhand eines Prototyps wurde erprobt, wie sich die Architektur für die dezentrale Entwicklung von Steuerungen eignet und wie der zukünftige Entwicklungsprozess aussieht.

In der weiteren Forschungsarbeit sollen weitere Bussysteme integriert werden. Des Weiteren sollen Modellanlagen dezentral mit Steuerungen nach dieser Architektur gesteuert werden. Anhand dieser Anlagen soll der konkrete Nutzen, bzw. die Einsparung im Engineering-Aufwand messbar gemacht werden.

**Quellen:**


[1] Weyrich, M.; Schmidt, J.; Ebert, C.: Machine-to-Machine Communication, IEEE Software, July/August 2014, S. 19-23

[2] ZVEI: Wichtige Etappenziele bei Industrie 4.0 erreicht, http://www.zvei.org/Presse/Presseinformationen/Seiten/Wichtige-Etappenziele-bei-Industrie-40-erreicht.aspx, Pressemitteilung, 12.03.2014

[3] Weyrich, Diedrich, Fay, Sollschlaeger, Kowalewsky, Göhner, Vogel-Heuser: Industrie 4.0 am Beispiel einer Verbundanlage, atpEdition 7-8/2014

[4] VDI/VDE: Industrie 4.0 Statusreport, April 2014 http://www.vdi.de/fileadmin/vdi_de/redakteur_dateien/sk_dateien/VDI_Industrie_4.0_Komponenten_2014.pdf

[5] IoT Architecture, Deliverable D1.3, Grant agreement no. 257521 www.iot-a.eu/public/public-documents/d1.5/at_download/file, abgerufen am 01.04.2015

[6] AUTOSAR Development Cooperation: Technical Overview: http://www.autosar.org/about/technical-overview/, abgerufen am 01.04.2015

[7] Niggemann, O.: Industrie 4.0 ohne modellbasierte Softwareentwicklung, atpEdition 5/2014

[8] Deiretsbacher, K.: OPC UA für Industrie 4.0, atpEdition 6/2014

[9] Epple, U.: Ansätze einer Referenzarchitektur für eine Industrie 4.0 konforme Systemplattform in der industriellen Automation, Automation 2014

[10] Göhner, P., Weyrich, M.: Agent-Based Concepts for Manufacturing Automation, Multiagent System Technologies, 12th German Conference, MATES 2014, Stuttgart, Germany, Proceedings, 2014