Wir bauen eine Maschine – und bauen das IoT gleich mit ein!

Scherz beiseite - wie kann man IoT-Methoden im Maschinenbau sinnvoll nutzen?

Robert Schachner, RST Industrie Automation GmbH bzw. Embedded4You e.V.

IoT – das „Internet of Things“ und „Industrie 4.0“: Diese zwei Gespenster geistern derzeit überall durch Veranstaltungen und Messen. Das Gesagte klingt dann oft verheißungsvoll; das Ziel der Träume, der große Erfolg werden angekündigt. Meistens bleibt es jedoch bei diversen Schlagworten, deren tieferer Sinn sich aber nicht so recht erschließen will.

Dieser Vortrag klärt auf, was sich dahinter verbirgt, und zeigt am Beispiel einer Maschine, wie man diese Technologien wirklich sinnvoll umsetzen kann.

Kommunikationsformen:

1. **Das Prozessdatenmodell**

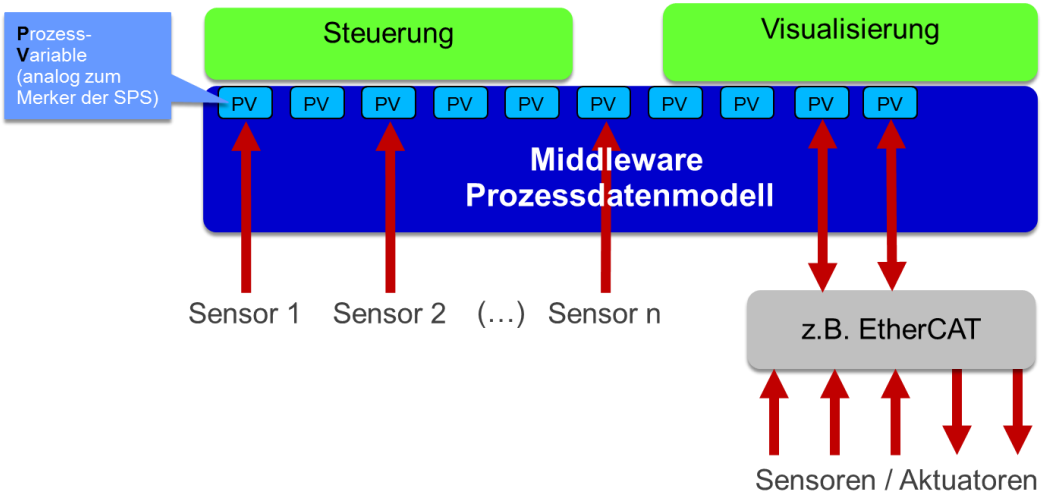

Abb. 1: Klassisches Prozessdatenmodell

Das Bild zeigt das Kommunikationsmodell, das in jeder SPS Steuerung zu finden ist. Ein globaler Speicher – das Prozessdatenmodell - dient als Ort für die Ablage von sämtlichen Prozessgrößen (Prozessvariablen oder Merker). Über diverse I/O-Funktionen werden elektrische Signale zyklisch eingespielt bzw. ausgegeben.

Ein Steuerprogramm greift diese Prozessvariablen auf, verarbeitet die Daten, und schreibt Prozessgrößen wieder in das Prozessdatenmodell zurück.

Da mit diesem Verfahren in jeder Aktualisierungsphase der „alte“ Wert überschrieben wird, liegen immer nur die aktuellsten Prozessgrößen vor. Echtzeitregelung, aber auch -steuerung, lässt sich so universell und einfach realisieren. Dieser simple und elegante Weg wird aber zum Nachteil, wenn mehrere Werte oder Statusmeldungen (z.B. Kommandos oder Fehlermeldungen) schneller auf eine Prozessvariable geschrieben als von der Gegenstelle ausgelesen werden.

Wichtige Informationen gehen verloren, weil sie zu schnell überschrieben werden. In solchen Situationen versagt das Modell.

2. Messagebasierte Kommunikation

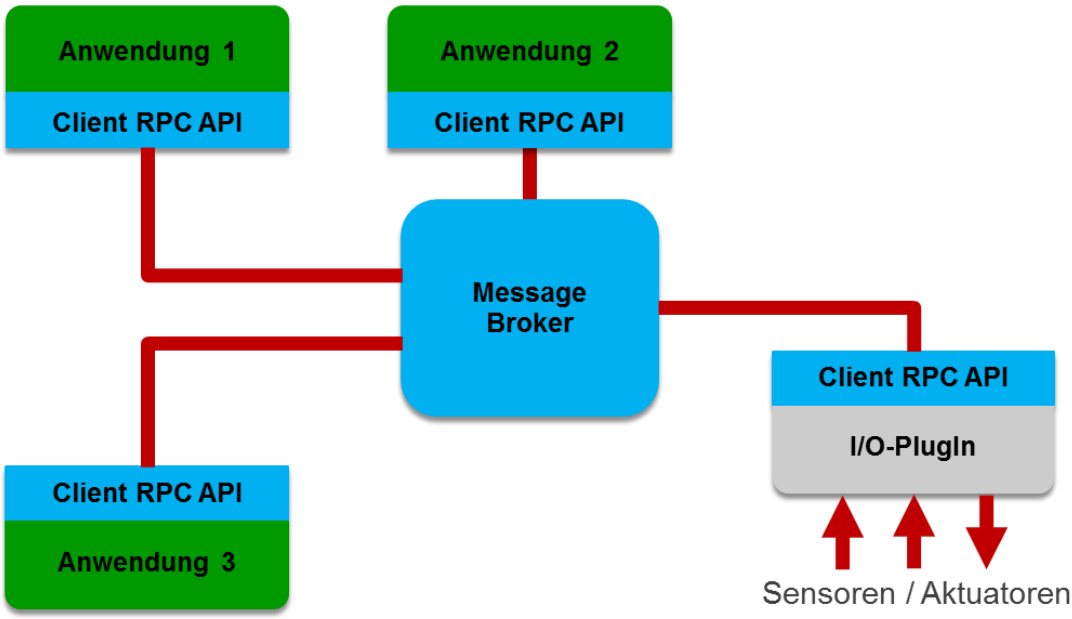

Abb. 2: Messagebasiertes Kommunikationsmodell

Diese Art von Kommunikation steckt eigentlich immer dahinter, wenn vom „IoT“ die Rede ist. Auf den ersten Blick klingen beide Kommunikationsarten erstmal sehr ähnlich. Bei genauerer Prüfung merkt man jedoch, dass sie sich beide Arten in ihrer Auswirkung fundamental unterscheiden. Im messagebasierten Kommunikationsmodell werden Botschaften serialisiert und wie im obigen Beispiel über einen zentralen Vermittler, den sogenannten Broker, an andere Clients verteilt. Im Unterschied zum Prozessdatenmodell liegt hier der Schwerpunkt in der zuverlässigen Zustellung **aller** Informationen – ein Überschreiben von noch nicht abgeholten Werten ist nicht mehr möglich. Das System bietet Vor- und Nachteile; so ist z.B. die Umsetzung von Reglern äußerst schwierig.

Neben der üblichen Adressierung, bei der der Sender jeweils den Empfänger benennt (Request/Response), gibt es den Publish/Subscribe-Mechanismus, der in der embedded-Welt etwas völlig neues darstellt. Der Sender (Publisher) adressiert seine Nachricht nicht mehr an einen spezifischen Empfänger, sondern stellt sie einfach unter einem eindeutigen Topic zur Verfügung. Jeder Empfänger (Subscriber) kann nun bei Bedarf dieses Topic abonnieren und erhält ab dann vom Broker die jeweils anfallenden Informationen, bis er das Abonnement wieder abbestellt. Somit sind Sender und Empfänger nicht mehr direkt miteinander verbunden; dies ermöglicht die beliebige Verteilung von Information.

Um wieder in die IoT-Begriffswelt zurückzukommen, ist hierdurch sozusagen eine „Cloud“ entstanden – ein universeller Ablageort für Daten, die beliebig vielen Teilnehmern zur Verfügung gestellt werden können. Diese muss jedoch entgegen landläufiger Meinung nicht im Internet liegen, sondern kann ihre Arbeit auch als

sogenannte „Private Cloud“ innerhalb einer Maschinensteuerung verrichten. Die Sicherheit der Daten bleibt damit gewahrt.

Anwendung in der Maschinensteuerung
Mit diesem Wissen wollen wir nun Strategien für eine moderne Maschinensteuerung erarbeiten. Begleitend dazu kann man auch das vom VDI und ZVEI vorgeschlagene Referenzarchitekturmodell RAMI4.0 in seine eigenen Betrachtungen hinzuziehen.

Mit dem bereits beschriebenen Prozessdatenmodell greifen wir wieder auf die Methodik zurück, mit der derzeit die Steuerung in fast jeder Maschine realisiert wird. Ob dabei als Programmiermethode eine Hochsprache, Structured Text, Funktionsplan oder ein modellbasierter Ansatz verwendet wird, ist vom Kommunikationsmodell unabhängig und hängt im Schwerpunkt von den Fähigkeiten des Programmierers ab. Beim Einsatz der SPS-Technik muss nur darauf geachtet werden, dass die Hardware in ein offenes Betriebssystem eingebunden wurde. Bei Middlewareplattformen wie Gamma V kann das integrierte Prozessdatenmodell mit nahezu jeder embedded-Hardware auf einem Linux- oder Windowsbetriebssystem eingesetzt werden.

Es bietet sich auch an, bereits bestehende I/O-Konzepte wiederzuverwenden. Bis jetzt bewegen wir uns in unseren Betrachtungen immer noch im Standardumfeld. Nun wollen wir zusätzlich dazu aufzeigen, wie effektiv und elegant Broker-Architekturen die klassische Steuerung bereichern können. Hierzu präsentieren wir zwei Beispiele.

Beispiel 1: Fehlermanagement
In klassischen Steuerungen wird dieses Thema sehr unterschiedlich umgesetzt. Häufig werden nur Prozessvariablen für die Zwischenspeicherung von Fehlerwerten verwendet. Werden jedoch kurz hintereinander zwei Fehlermeldungen generiert, so kann die ältere überschrieben werden, wenn diese nicht schnell genug ausgelesen werden; häufig kommt so nur die jeweils letzte Fehlermeldung bei der Anwendung an. Als Workaround können Merker angelegt werden; schlimmstenfalls führt das dann aber zu Hunderten von Merkern, die jeweils mit einer Fehlernummer belegt sind. So wird ein Dilemma vermieden, indem ein anderes entsteht.

Mit den Methoden der messagebasierten Kommunikation lässt sich das sehr viel eleganter lösen.

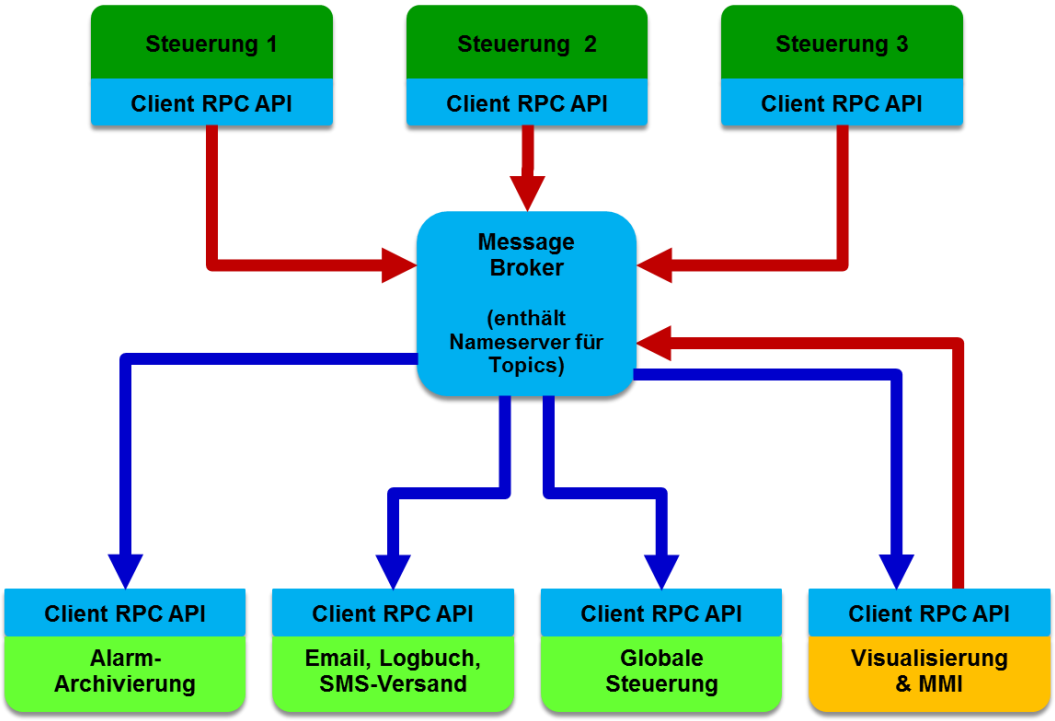

Abb. 3: Umsetzung des Fehlermanagements über eine Broker-Architektur

Die Steuerungssysteme (Steuerung 1 – Steuerung 3) entsprechen der bereits beschriebenen klassischen Architektur, wurden aber jeweils um einen Client für das Messaging erweitert. Dieser Client ermöglicht das Absetzen einer Fehlermeldung über die Client RPC API. Diese Meldung kann über Properties diverse Details wie Fehlernummer, Fehlerklasse, Datum und Uhrzeit usw. enthalten. Innerhalb des Kommunikationsmodells tritt nun jede Steuerung als sogenannter Publisher auf, der seine Fehlermeldungen über ein Topic (beispielsweise „ERROR“) zur Verfügung stellt.

Der Broker kann entweder auf einem dedizierten Rechner oder – bei einem Multibroker-Konzept – verteilt als eigene Instanz auf jedem teilnehmenden Rechner laufen. Er verteilt die auf dem „ERROR“-Topic von den Publishern eingehenden Daten weiter an die diversen Subscriber.

Empfangen und verarbeitet werden diese Daten dann von Diensten, die im Netzwerk als Subscriber angemeldet sind. So können durch die bereits beschriebene Trennung zwischen Sendern und Empfängern beliebige Dienste zur Verarbeitung der Fehlerdaten angedockt werden, welche – völlig unabhängig von den Publishern und den anderen Subscribern – jeweils nur mit dem Topic „ERROR“ kommunizieren. Wie im Bild dargestellt lassen sich nun unterschiedlichste Anwendungen programmieren, die universelle Arbeiten wie Archivierung durchführen oder z.B. im Fehlerfall Emails oder SMS an Servicetechniker verschicken. Die Architektur lässt sogar zu, dass beliebig viele Fehlervisualisierungen an der Maschine oder auf entfernten Terminals laufen, die voneinander unabhängig bedient werden können.

So entsteht ein rechnerübergreifendes Fehlermanagement, dessen Flexibilität und Einfachheit von keiner anderen Technik übertroffen werden kann. Ganze Maschinen mit ihren dedizierten Steuerungen können nun ohne weitere Konfiguration vernetzt

und ihre Dienste zentral genutzt werden. So benötigt man in einer ganzen Maschinenhalle nur noch ein zentrales UMTS-Interface für die Benachrichtigung des Servicetechnikers.

Beispiel 2: Ad-hoc-Vernetzung von Maschinenteilen

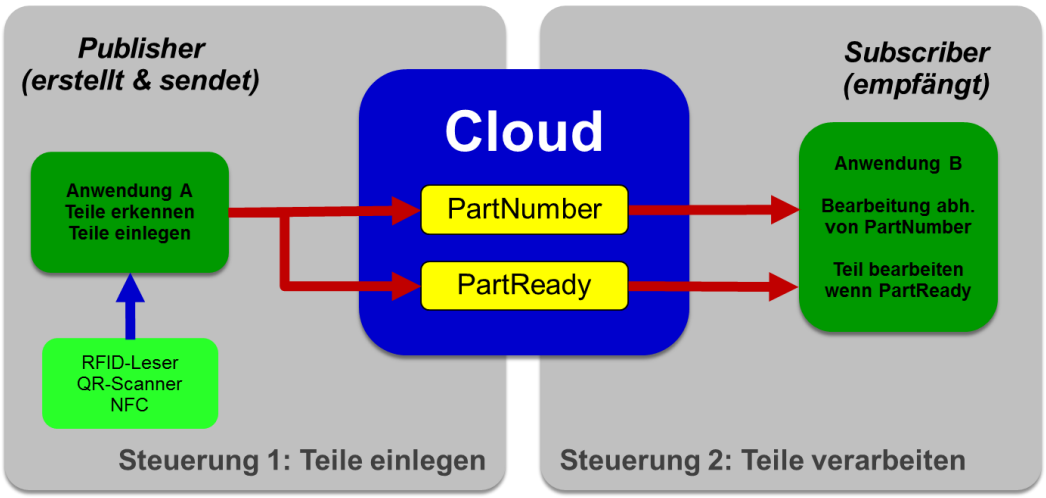

Abb. 4: Ad-hoc-Vernetzung von Maschinenteilen über eine Broker-Architektur

Die Vernetzung von Maschinen geschieht heutzutage meistens über Feldbusse. Je nach Anzahl der Verbindungen kann eine Vernetzung einen beträchtlichen Konfigurationsaufwand mit entsprechendem Fehlerpotential nach sich ziehen.

Der Publish/Subscribe-Mechanismus bietet auch hier eine äußerst elegante Alternative. Wie in Abbildung 4 dargestellt, generiert zum Beispiel Anwendung A über den angedockten RFID-Leser eine Partnummer und das dazugehörige Statussignal „PartReady“. Als Publisher werden diese Informationen im Netz zur Verfügung gestellt. Nun können mit dem Netzwerk verbundene andere Maschinenteile als Subscriber diese und beliebige andere Informationen abonnieren. In diesem Beispiel erhält System 2 als Subscriber die Informationen darüber, welches Teil (PartNumber) wann abzuholen ist (PartReady).

So entsteht wieder eine private Cloud, über die einzelnen Maschinenteile untereinander Informationen austauschen. Maschinenteile können sich nun – nur über die relevanten Topics – automatisch zu einer vernetzten Maschine zusammenfügen. Da in einem komplexen Gesamtsystem unter Umständen sehr viele Topics verwaltet werden müssen, ist es wichtig, die Topicnamen im Rahmen einer Topologie oder sogar einer Ontologie einheitlich zu strukturieren.

Die messagebasierte Kommunikation bietet noch viele weitere Möglichkeiten, um eine Steuerung effektiver zu gestalten und zu steuern. Deshalb wird der Vortrag über das hier kurz Zusammengefasste hinaus unter anderem auch noch Strategien der Vernetzung mit Produktionsnetzwerken aufzeigen und als Praxisbeispiel detailliert auf eine bereits erfolgreich realisierte Maschinensteuerung eingehen.