Jahreskolloquium "Kommunikation in der Automation" (KommA), Lemgo, November 2016 | ISBN: 978-3-9814062-6-9

# Übertragbarkeit des Plug&Play-Prinzips aus der Informationstechnik auf die Automatisierungstechnik

Lars Dürkop, Jürgen Jasperneite

Institut für industrielle Informationstechnik
Hochschule Ostwestfalen-Lippe
Langenbruch 6
32657 Lemgo
lars.duerkop@hs-owl.de
juergen.jasperneite@hs-owl.de

**Abstract:** Die Inbetriebnahme von Automatisierungssystemen auf der Feldebene ist untrennbar mit einem hohen manuellen Konfigurationsaufwand verbunden: Feldgeräte müssen in die Steuerungsapplikation integriert werden und die zur Vernetzung zwischen Feld- und Steuerungsebene eingesetzten Kommunikationssysteme müssen parametriert werden. In der Informationstechnik gibt es ähnliche Problemstellungen: So müssen Peripheriegeräte über einen Kommunikationsbus an einen PC angeschlossen werden oder in einem Heimnetz verteilte Geräte müssen miteinander interagieren. Hierzu gibt es etablierte Ansätze wie USB oder UPnP, welche dem Endanwender ein „Plug&Play" bieten, sodass er sich im Idealfall mit keinerlei Konfigurationsarbeiten befassen muss. In diesem Beitrag wird untersucht, wie „Plug&Play" (PnP) im Bereich der Informationstechnik realisiert wird und ob und ggf. wie sich diese Mechanismen auf die Automatisierungstechnik übertragen lassen.

## 1 Einleitung

Ein frühes Beispiel für PnP ist die Installation von Erweiterungskarten bei PCs der x86-Architektur. Bei dem ursprünglich verwendeten Industry Standard Architecture (ISA)-Bus war es erforderlich, dass der Benutzer die benötigten Systemressourcen der Erweiterungskarte mittels Steckbrücken oder kleinen Schaltern zuteilen musste. Dieser fehlerträchtige Vorgang wurde beim ISA-Nachfolger, dem Peripheral Component Interconnect (PCI)-Bus, durch eine automatische Ressourcenzuweisung ersetzt.

Auch bei USB erfolgt der Anschluss von Peripheriegeräten für den Endanwender (in den meisten Fällen) vollkommen transparent. Im Prinzip kann dieses Szenario mit der Inbetriebnahme eines Automatisierungssystems verglichen werden: Ein Anwender möchte verteilte Endgeräte (PC-Peripheriekomponenten in der Informationstechnik (IT) / Feldgeräte in der Automatisierungstechnik (AT)) über ein Kommunikationssystem (z. B. USB in der IT / Echzeit-Ethernet in der AT) an eine zentrale Steuerungskomponente (PC in der IT / SPS in der AT) anschließen.

Beiden Anwendungsfällen ist weiterhin gemein, dass Geräte mit unterschiedlichen Dienstgüte-Anforderungen an demselben Kommunikationssystem betrieben werden müssen. So gibt es sowohl bei USB wie auch bei Echtzeit-Ethernet (real-time Ethernet, RTE) verschiedene Kommunikationsklassen für zeitkritische und nichtzeitkritische Daten. Bei der Inbetriebnahme solcher Systeme sind jedoch grundlegende Unterschiede zu beobachten:

Die Bus-Konfiguration erfolgt bei USB autark ohne das Erfordernis manueller Eingriffe durch den Anwender. Dieser Enumeration genannte Vorgang ist einer der wesentlichen Bestandteile des PnP-Prinzips bei USB. Bei industriellen Bussystemen müssen viele Vorgänge manuell vorgenommen werden: So müssen die angeschlossenen Geräte, ihre Adressen sowie die zeitlichen Vorgaben an die Kommunikation manuell vorgegebenen werden.

Auch auf Applikationsebene unterscheidet sich USB erheblich von industriellen Systemen. Bei USB erfolgt die Integration der Peripheriegeräte in die Applikation in der Regel automatisch: Schließt der Benutzer eine Maus an einen PC an, steht ihre Funktionalität sofort ohne weitere Konfigurationsschritte zur Verfügung. Damit eine SPS die Funktionalität eines Feldgerätes nutzen kann, muss dagegen zuerst eine Prozessdatenzuordnung zwischen der SPS-Applikation und den IO-Objekten der Feldgeräte durchgeführt werden.

Im Folgenden wird untersucht, durch welche Mechanismen PnP in der Informationstechnik realisiert wird. Der

Schwerpunkt liegt dabei auf den Standards USB und UPnP. Anschließend wird diskutiert, inwiefern sich die identifizierten Mechanismen auch auf die Inbetriebnahme von RTEs in der Automatisierungstechnik übertragen lassen. Bei diesem Beitrag handelt es sich um einen Auszug aus [Dür16].

## 2 PnP in der IT: Stand der Technik

### 2.1 Universal Serial Bus

Der Universal Serial Bus (USB) [USB00] ist eine Schnittstelle zum einheitlichen Anschluss von Peripheriegeräten an Computer. Vor der Einführung von USB wurden im PC-Bereich verschiedene Standards zum Geräteanschluss wie die serielle und parallele Schnittstelle sowie PS/2 genutzt. Wie auch beim ISA-Bus mussten den dort angeschlossenen Geräten bestimmte Systemressourcen zugeteilt werden, weiterhin konnte pro Anschluss in der Regel nur ein Gerät angeschlossen werden. Bei nicht ausreichenden Schnittstellen mussten Erweiterungskarten installiert werden, wobei unterschiedliche Gerätearten oftmals unterschiedliche Stecker erforderten. Der Anschluss eines neuen Gerätes war mit dem Neustart des kompletten Systems verbunden.

Die Einführung von USB hat die genannten Nachteile beseitigt und hat weiterhin durch kontinuierliche Weiterentwicklung immer höhere Datenübertragungsraten ermöglicht, sodass USB die meisten anderen Schnittstellen für Peripheriegeräte heute ersetzt hat. USB folgt einer Master/Slave-Architektur, in der bis zu 127 Geräte, die Slaves, in einer mehrstufigen Sterntopologie an einen zentralen Host, dem Master, angeschlossen werden können. Die Mehrstufigkeit wird durch Hubs erreicht, welche jeweils den Anschluss mehrerer Geräte erlauben, wobei maximal sechs Hubs kaskadiert werden können. Ein Beispiel für eine mögliche USB-Topologie wird in Abbildung 1 gezeigt.

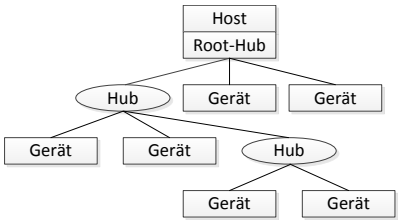


Abbildung 1: Beispielhafte Topologie eines USB-Systems

Wie in Abbildung 2 gezeigt wird, enthält jedes USB-Gerät einen oder mehrere Endpunkte, zu denen der Host jeweils eigenständige Kommunikationsverbindungen aufbauen kann. Ein Endpunkt kann dabei uni- oder bidirektionale Kommunikation unterstützen sowie in einem von vier Übertragungsmodi betrieben werden: Der isochrone Transfer wird für Verbindungen mit garantierter Datenrate verwendet, der Interrupt-Transfer für die zyklische Abfrage von Geräten, der Bulk-Transfer für große und nicht zeitkritische Datenmengen sowie der Control-Transfer für den Austausch von Konfigurationsdaten. Die genaue Konfiguration der Endpunkte wird vom jeweiligen USB-Gerät festgelegt.

Die PnP-Fähigkeit von USB beruht hardwareseitig auf der automatischen Konfiguration der businternen Kommunikation sowie softwareseitig auf einheitlichen funktionalen Schnittstellen für bestimmte Geräteklassen. Die Grundzüge beider PnP-Mechanismen werden im Folgenden dargestellt. Die Beschreibungen beziehen sich auf den USB-Standard 2.0.

#### 2.1.1 Konfiguration der Kommunikation

Die Konfiguration der Kommunikation, welche in USB Enumeration genannt wird, ist für den Endanwender vollkommen transparent und besteht u. a. aus den folgenden Schritten:

- Geräteerkennung: Physikalisch verwendet USB ausschließlich Punkt-zu-Punkt Verbindungen zwischen Hub und Gerät. Wird ein neues Gerät an einen Hub angeschlossen, so erkennt der Hub den Einsteckvorgang über die veränderten elektrischen Eigenschaften am entsprechenden Port.
- Aushandeln der Datenrate: USB unterstützt mehrere Geschwindigkeitsklassen. Nachdem der Hub den

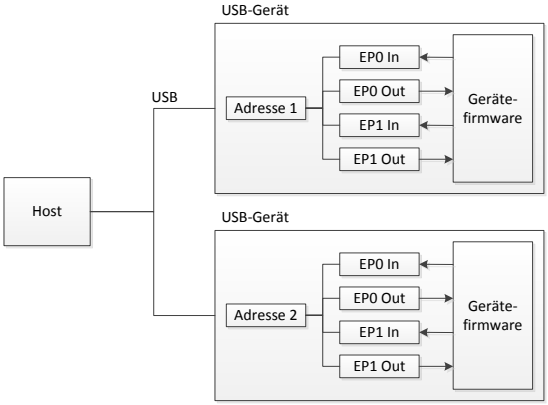


Abbildung 2: USB-Kommunikationsstruktur

Anschluss eines neuen Gerätes erkannt hat, handeln Hub und Gerät über eine sogenannte Chirp-Sequenz die höchstmögliche Geschwindigkeit aus, die von beiden Partnern unterstützt wird.

Nachdem die physikalische Verbindung zwischen Hub und Gerät aufgebaut wurde, informiert der Hub den Host über den Anschluss eines neuen Gerätes. Der Host übernimmt anschließend den weiteren Konfigurationsvorgang:

- Adresszuweisung: Im uninitialisierten Zustand ist das neue Gerät über die Standard-Adresse 0 erreichbar. Der Host nutzt diese Adresse, um mit dem Gerät eine Kommunikationsverbindung aufzubauen und diesem anschließend eine neue eindeutige Adresse zuzuweisen.
- Abruf der Gerätekonfiguration: Der Host ruft grundlegende Informationen über das Gerät ab. Diese Daten enthalten u. a. eine Hersteller-ID, eine Produkt-ID, eine Seriennummer sowie die Anzahl der Endpunkte einschließlich deren Konfigurationen. Die Informationen sind in festgelegten Datenstrukturen, den Deskriptoren, enthalten. Der Zugriff auf die Deskriptoren erfolgt immer über den Endpunkt 0, den somit jedes USB-Gerät unterstützen muss.
- Kommunikationsplanung: Die zur Verfügung stehende Bandbreite wird in Zeitschlitze unterteilt, welche vom Host den einzelnen Endpunkten gemäß ihren in den Deskriptoren enthaltenen Anforderungen zugeteilt werden. Isochrone und Interrupt-Transfers dürfen dabei in der Summe maximal 80 % der zur Verfügung stehenden Bandbreite reservieren. Überschreitet die Anforderung eines neu angeschlossenen Gerätes diese Grenze, schlägt der Verbindungsaufbau fehl. Für Control-Transfers werden statisch 20 % der Bandbreite reserviert. Bei Bulk-Transfers erfolgt keine Bandbreitenreservierung, diese werden immer dann ausgeführt, wenn im jeweiligen Kommunikationszyklus alle anderen Transfers abgeschlossen sind.
- Abschluss der Konfiguration: Nachdem die Kommunikation zwischen Host und Gerät aufgebaut ist, meldet der Host den Anschluss eines neuen Gerätes an das Betriebssystem. Ab diesem Zeitpunkt kann eine Applikation unter Nutzung eines generischen USB-Treibers Daten mit den Endpunkten des Gerätes austauschen.

### 2.1.2 Einheitliche funktionale Schnittstelle

In USB wird die komplette Buskonfiguration, wie soeben beschrieben, autark von internen Mechanismen gesteuert, sodass Applikationen nach dem Hinzufügen eines Gerätes automatisch auf dessen Endpunkte zugreifen können. Um die Kommunikation mit der Hardware zu erleichtern, werden zusätzlich Gerätetreiber verwendet, die den Zugriff auf das Gerät abstrahieren, sodass die Anwendung keine binären Rohdaten mit den Geräte-Endpunkten austauschen muss. Stattdessen bietet der Treiber für die Nutzung der Gerätefunktionalitäten entsprechende Funktionsaufrufe an und setzt diese intern in die Steuersignale und -daten um.

Allerdings müssen Applikationen bei der Verwendung von Gerätetreibern normalerweise an den gerätespezifischen Treiber angepasst werden, da verschiedene Treiber die gleiche Gerätefunktionalität unterschiedlich

implementieren können. Im Ergebnis könnte eine Applikation nur mit den Geräten derjenigen Hersteller zusammenarbeiten, für deren Treiber sie explizit entworfen wurde.

Aus diesen Gründen nutzt USB das Prinzip der Geräteklassen. Eine Geräteklasse fasst dabei die Ansteuerung verschiedener Geräte ähnlicher Funktionalität zusammen, indem in ihr einheitliche Attribute und Funktionen, über die alle Geräte einer Klasse gesteuert werden können, definiert werden. Durch die Verwendung eines einzigen Klassentreibers wird eine Applikation damit in die Lage versetzt, unterschiedliche Geräte verschiedener Hersteller zu nutzen. Die Zugehörigkeit zu einer bestimmten Geräteklasse teilt ein USB-Gerät dem Host während der Enumeration durch entsprechende Deskriptoren mit.

Durch die Verwendung von Klassentreibern wird es dem Endbenutzer ermöglicht, Geräte an einen PC anzuschließen und diese ohne weitere manuelle Schritte zu benutzen. Die Voraussetzung dafür sind zum einen die Unterstützung der jeweiligen Geräteklasse durch das Betriebssystem und zum anderen die Existenz einer entsprechenden Klasse für das Gerät. Letzterer Punkt ist weniger eine technologische als eine organisatorische Problemstellung. So ist die Definition und Ausgestaltung von Geräteklassen maßgeblich davon abhängig, ob der Standardisierungsprozess von den Geräteherstellern unterstützt wird und ob sie ihre Produkte den Vorgaben der Klassendefinition entsprechend konzipieren. Im Falle von USB haben sich Hard- und Softwarehersteller in Form des USB Implementers Forum zusammengeschlossen, welches eine Vielzahl von Geräteklassen (z. B. die Human Interface Device Class, welche u. a. Definitionen für Tastaturen und Mäuse enthält) geschaffen hat. Die breite Herstellerunterstützung hat maßgeblich zum Erfolg von USB beigetragen.

Allerdings müssen USB-Geräte nicht zwangsläufig kompatibel zu einer bestimmten Klasse sein, sodass es drei Möglichkeiten für den Entwurf von USB-Geräten gibt:

- Das USB-Gerät entspricht vollständig einer Klassendefinition. Dementsprechend steht seine Funktionalität uneingeschränkt einer Applikation zur Verfügung (vorausgesetzt, dass diese die Klasse ebenfalls unterstützt).
- Das USB-Gerät implementiert eine Klassendefinition, bietet darüber hinaus jedoch noch zusätzliche Funktionalitäten an. In diesem Fall können Applikationen, welche auf der entsprechenden Geräteklasse beruhen, nur eine Grundfunktionalität des Gerätes nutzen. Für alle weiteren Funktionen muss die Applikation gerätespezifisch angepasst werden.
- Das USB-Gerät lässt sich keiner Geräteklasse zuordnen. Hier können die Gerätefunktionalitäten nur von Applikationen genutzt werden, die speziell auf das Gerät ausgelegt sind.

## 2.2 Universal Plug and Play

UPnP [UPn08] bezeichnet einen auf Protokollen aus dem Internet-Umfeld basierenden Standard für die konfigurationslose Zusammenarbeit von verteilten IT-Komponenten, welche über ein IP-basiertes Netzwerk miteinander verbunden sind. In UPnP stellen Endgeräte ihre Funktionalitäten in Form von Diensten zur Verfügung (siehe Abbildung 3), welche von Kontrollpunkten aus aufgerufen werden können. Ein Dienst beinhaltet dabei ausführbare Aktionen und abrufbare Statusvariablen. In einem Gerät können ebenfalls mehrere Dienste und Endpunkte sowie eine Kombination aus beiden realisiert werden.

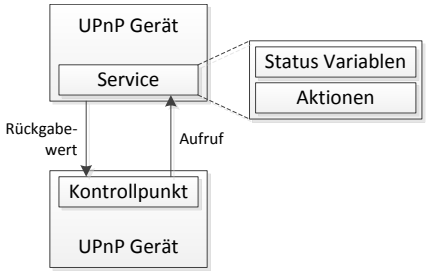


Abbildung 3: Grundlegende Architektur eines UPnP-Systems

UPnP legt fest, wie durch Einsatz bereits existierender Protokolle Dienste definiert, aufgefunden und kontrolliert werden können:

- Adressierung: Jedes UPnP-Gerät enthält einen Client für das Dynamic Host Configuration Protocol (DHCP) [RFC97], welches für die dynamische Zuweisung von IP-Adressen eingesetzt wird. Befindet sich kein entsprechender DHCP-Server im Netzwerk, weist sich jedes Gerät im entsprechenden Netz selbstständig eine IP-Adresse gemäß dem Internet-Standard RFC 3927 [RFC05] zu.

- Erkundung: Geräte geben ihre Dienste im Netzwerk per Multicast bekannt und nutzen dazu das Simple Service Discovery Protocol (SSDP) [UPn08]. Kontrollpunkte können weiterhin ebenfalls über SSDP nach bestimmten Diensten suchen.

- Beschreibung: Nachdem ein Endpunkt einen passenden Dienst gefunden hat, ruft er dessen Beschreibung mittels HTTP [RFC14] ab. UPnP definiert Vorgaben für das Format der Dienstbeschreibung, welche u. a. eine Liste der aufrufbaren Aktionen, Parameter für jede Aktion sowie eine Liste von Statusvariablen enthält.

- Kontrolle: Ein Endpunkt kann mittels HTTP die ihm nach Abruf der Beschreibungen bekannten Aktionen eines Dienstes aufrufen. Die Ergebnisse des Aufrufs spiegeln sich in einer Änderung der Statusvariablen des Dienstes wieder.

- Benachrichtigung: Wenn sich der Zustand eines Dienstes ändert, verschickt dieser Ereignisbenachrichtigungen. Um diese zu empfangen, muss ein Kontrollpunkt sich vorher beim Dienst registrieren und die gewünschten Benachrichtigungen abonnieren.

- Darstellung: UPnP-Geräte können optional in ihrer Gerätebeschreibung eine Adresse anbieten, über die ein Endanwender nähere Informationen über das Gerät abrufen kann. Dabei kann es sich beispielsweise um die URL des Web-Interfaces des Gerätes handeln.

UPnP gibt über die oben genannten Methoden den Rahmen vor, wie verschiedene Geräte über ein Netzwerk miteinander interagieren können. Zusätzlich definiert UPnP –vergleichbar mit den Geräteklassen in USB– konkrete Gerätetypen und deren Dienst-Schnittstellen.

## 3 Bewertung

Im Folgenden soll geprüft werden, welche der vorgestellten Plug-and-Play Konzepte von USB und UPnP sich für die automatische Integration von Feldgeräten und die damit verbundene automatische Konfiguration von Echtzeit-Ethernet eignen.

### 3.1 USB

- Geräteerkennung: Ein ähnliches Verfahren zur elektrischen Erkennung von Netzwerkknoten und zur Aushandlung der Geschwindigkeitsklasse ist unter dem Begriff Autonegotiation bereits im Ethernet-Standard definiert. Eine weitere Eigenschaft von USB ist es, dass der Host über den Anschluss bzw. das Entfernen eines Gerätes automatisch informiert wird. Einen vergleichbaren Mechanismus gibt es bei Ethernet nicht, insbesondere existiert kein mit der Rolle des Hosts vergleichbarer zentraler Knoten, welcher informiert werden könnte.

  *Fazit: nicht übertragbar*

- Adresszuweisung: Auf Ethernet-Ebene besteht keine Notwendigkeit für eine automatische Zuweisung von Adressen, da jeder Ethernet-Knoten bereits eine vom Hersteller voreingestellte weltweit einmalige Hardwareadresse enthält. Für höhere Protokollschichten kann jedoch eine Adressvergabe ähnlich dem in USB eingesetzten Verfahren geeignet sein. So müssen beispielsweise in Ethernet Powerlink alle Geräte manuell mit einer einmaligen Node ID versehen werden. In Analogie zu USB könnte stattdessen jeder Powerlink-Knoten mit einer einheitlichen vordefinierten Node ID das Netzwerk betreten, woraufhin der zentrale Managing Node den Knoten ihre eindeutigen IDs automatisch zuweist. Als Voraussetzung für ein solches Verfahren müsste die starre Prozessdatenzuordnung zwischen Steuerungsapplikation und Feldgerät über die Node ID aufgehoben werden.

  *Fazit: bedingt übertragbar*

- Abruf der Gerätekonfiguration: In Profinet muss für modulare Geräte der Ausbaugrad während des Engineerings angegeben werden. Würde die Gerätekonfiguration dahingegen vom Engineering ausgelesen werden, könnte dieser manueller Schritt der Inbetriebnahme entfallen.

  *Fazit: übertragbar*

- Kommunikationsplanung: USB basiert wie die meisten Echtzeit-Ethernets auf einer zentral durchgeführten Zeitschlitz-basierten Ressourcenzuteilung. Jedoch nutzt USB eine andere Informationsquelle als Planungsgrundlage als dies bei RTEs der Fall ist. So werden in USB die Ressourcenanforderungen von den Endgeräten spezifiziert – bei RTEs werden die zeitlichen Parameter für den Prozessdatenaustausch manuell vom Benutzer vorgenommen.

  Eine Übertragung des USB-Prinzips auf die Automatisierungstechnik ist nur bedingt möglich: Prinzipiell könnten auch die Ressourcenanforderungen der Feldgeräte aus ihren Gerätebeschreibungsdateien ausgelesen werden, da dort alle zyklisch zu übertragenden Daten in Form von Ausgangs- und Eingangsparametern beschrieben werden. Zusätzlich enthalten die Dateien für jedes Gerät die minimal unterstützte Zykluszeit. Auf Basis dieser Informationen könnte ein „worst case"-Kommunikationsplan berechnet werden, welcher für jegliche Parameter aller Feldgeräte eine Übertragung in der minimalen Zykluszeit vorsieht. Auf diese Weise kann sichergestellt werden, dass alle Prozessdaten rechtzeitig zwischen Steuerung und Feldgerät ausgetauscht werden.

  Eine wesentliche Einschränkung dieses Ansatzes stellt jedoch die erzeugte Netzlast dar. Da der Datenaustausch unter Umständen in wesentlich kürzeren Zeitabständen erfolgt als dies für die korrekte Funktionalität der Applikation notwendig wäre, kann die maximale Leistungsfähigkeit des Echtzeit-Netzes deutlich früher erreicht werden als bei der konventionellen manuellen Kommunikationsplanung.

  *Fazit: bedingt übertragbar*

- Einheitliche funktionale Schnittstelle: Dies ist die aus Sicht eines Applikationsentwicklers wichtigste PnP-Fähigkeit von USB. Sie ermöglicht es, Anwendungen ohne Wissen über die konkret eingesetzten Peripheriekomponenten zu entwickeln, solange ein geeigneter Klassentreiber zur Verfügung steht. Weiterhin muss der Endbenutzer keine manuellen Einstellungen vornehmen, um die Kommunikation zwischen Applikation und Peripherie zu ermöglichen. Ein ähnliches Verfahren könnte in der Automatisierungstechnik die manuelle Prozessdatenzuordnung zwischen Steuerung und Feldgeräten ersetzen. Anstatt über Variablen oder ähnliche Kommunikationsobjekte, die mit den physikalischen Adressen der Ein- und Ausgänge verbunden werden müssen, würde eine Steuerungsapplikation nur noch mit den entsprechenden Klassentreibern kommunizieren. Die Weiterleitung der Prozessdaten an die Feldgeräte würde von den Treibern übernommen werden.

  Dieses Prinzip lässt sich jedoch nur bedingt auf die Automatisierungstechnik übertragen: Zur Schaffung entsprechender Klassendefinitionen für Feldgeräte müsste seitens der Hersteller die dafür notwendige breite Unterstützung gewährt werden. Im Bereich von USB ist dies gelungen – dort wird jedoch mit Endverbrauchern eine andere Zielgruppe adressiert: Wäre im Konsumentenbereich die Inbetriebnahme von PCs mitsamt ihrer Peripherie nur mit Expertenwissen möglich, würde dies viele Verbraucher vor Investitionen abschrecken und der Markt für IT-Technik wäre wesentlich kleiner. Demgegenüber muss bei automatisierungstechnischen Komponenten eine entsprechende Nachfrage nicht erst geschaffen werden, diese ergibt sich durch die Notwendigkeit zur Produktions- und Prozessautomatisierung. Würden die Hersteller hier ihre Produkte standardisieren und damit leichter austauschbar machen, könnte dies zu einer starken Wettbewerbszunahme führen, welche nicht im Interesse der etablierten Hersteller sein dürfte. Im Gegenteil findet eher eine Individualisierung durch herstellerspezifische Funktionalitäten statt.

  Weiterhin gibt es aus technischer Sicht Hindernisse bei einer direkten Übertragung des Klassentreiber-Prinzips auf die Automatisierungstechnik: Bei PCs erfüllen die meisten Peripheriegeräte eindeutige Funktionalitäten (z. B. Maus, Tastatur, Monitor) – in der Automatisierungstechnik gibt es jedoch auch Feldgeräte wie digitale Eingabe/Ausgabe-Komponenten, welche erst durch ihre Integration in den Prozess einer bestimmten Funktionalität zugeordnet werden können.

  Allerdings gibt es auch in der Automatisierungstechnik in der Form von Profilen Ansätze zur einheitlichen Beschreibung von Feldgerätefunktionalitäten. So definiert die Normenreihe IEC 61800-7 [IEC07] beispielsweise eine universelle Schnittstelle für die Antriebstechnik und deren Umsetzung auf verschiedene Kommunikationssysteme und Applikationsprotokolle wie CIP (CIP Motion), CANopen (CiA 402) oder Profinet (PROFIdrive). Eine Abdeckung der meisten auf dem Markt befindlichen Feldgeräte durch entsprechende Profile ist in der Automatisierungstechnik jedoch –auch aufgrund der Vielzahl unterschiedlicher Geräte– jedoch nicht in Sicht.

  *Fazit: bedingt übertragbar*

### 3.2 UPnP

- Adressierung: Eine automatische Adressvergabe wäre bei der automatischen RTE-Konfiguration prinzipiell möglich. Allerdings ist zu berücksichtigen, dass während der herkömmlichen Inbetriebnahme eines RTEs die Prozessdaten der Steuerung fest mit den Adressen der Feldgeräte verknüpft werden. Werden die Adressen erst während des Hochlaufens des Netzwerkes vergeben, so muss eine Lösung für eine dynamische Prozessdatenzuordnung gefunden werden.

  *Fazit: bedingt übertragbar*

- Erkundung, Beschreibung, Kontrolle, Benachrichtigung: Bei UPnP werden die vorhandenen Geräte und ihre Dienste dynamisch zur Laufzeit erkundet. Anschließend werden zwischen Kontrollpunkt und Dienst Details ausgetauscht, welche zum Aufrufen des Dienstes erforderlich sind. Dieses Prinzip ließe sich auch auf die Automatisierungstechnik zum Ersatz bzw. zum Vereinfachen der statischen Prozessdatenzuordnung übertragen, indem man die Steuerungskomponente als Kontrollpunkt auffasst, welcher Dienste der Feldgeräte (z. B. Einlesen eines bestimmten Prozesswerts) aufruft.

  Ähnlich wie bei der notwendigen Definition von Geräteklassen in USB benötigt allerdings auch dieses Konzept einen industrieweiten Standardisierungsprozess für die einheitliche Beschreibung von Diensten, um herstellerunabhängig Interaktionen zwischen der Steuerung und den von den Feldgeräten angebotenen Dienste zu gewährleisten. Weiterhin müssen die Echtzeitanforderungen der Automatisierung berücksichtigt werden, welche von den TCP/IP-basierten UPnP-Protokollen nicht erfüllt werden können.

  *Fazit: übertragbar*

- Darstellung: Ähnlich wie bei UPnP könnten auch Feldgeräte die URL ihres Web-Interfaces im Netzwerk propagieren. Dieses könnte von Technikern genutzt werden, um das Feldgerät zu konfigurieren oder Diagnoseinformation abzurufen. Entsprechende Web-Server sind heute bereits auf einer Vielzahl von Feldgeräten enthalten – neu wäre die automatische Bekanntgabe der URL, sodass der Techniker beispielsweise nicht erst die IP-Adresse des Geräte herausfinden muss.

  *Fazit: übertragbar*

## 4 Zusammenfassung

Für den möglichst konfigurationslosen Einsatz von Peripheriegeräten oder Netzwerkkomponenten hat sich in der Informationstechnik (IT) das „Plug&Play“ (PnP)-Prinzip bewährt. Zwei weit verbreitete IT-Standards, welche diesem Prinzip folgen, sind USB und UPnP. In Hinblick auf PnP zeichnet sich USB dadurch aus, dass der Kommunikations-Bus automatisch konfiguriert wird und für die applikationsseitige Geräteintegration auf standardisierte Profile zurückgegriffen werden kann. UPnP setzt auf TCP/IP als Protokoll auf und beinhaltet zusätzliche Protokolle für die automatische Geräteerkundung im Netzwerk. Auch UPnP nutzt Profile für den standardisierten Applikationszugriff.

In diesem Beitrag wurde geprüft, welche PnP-Mechanismen aus der IT auf die Automatisierungstechnik (AT) übertragen werden könnten. Die Integration von Feldgeräten in die Steuerungsapplikation erfordert hier vielfältige manuelle Inbetriebnahmeschritte wie die Prozessdatenzuordnung und die Konfiguration des Echtzeit-Netzwerkes. Es hat sich gezeigt, dass insbesondere die Erkundungsfunktionalitäten von UPnP auch in der AT zur Unterstützung des Inbetriebnahmeprozesses eingesetzt werden können. Die Übertragbarkeit eines der wesentlichen Grundlagen für die erfolgreiche Realisierung von PnP in der IT –der Einsatz von standardisierten Geräteprofilen– auf die AT wird derzeit durch die fehlende Standardisierung gehemmt. Ob sich Profile in der AT im breiten Maße durchsetzen werden, ist fraglich.

## Danksagung

Diese Arbeit wurde mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) im Rahmen des Spitzenclusters „Intelligente Technische Systeme OstWestfalenLippe (it’s OWL)“ gefördert.

## Literatur


[Dür16] Lars Dürkop. *Automatische Konfiguration von Echtzeit-Ethernet*. Dissertation, Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg, 2016.

[IEC07] IEC 61800-7: Adjustable speed electrical power drive systems – Part 7: Generic interface and use of profiles for power drive systems. Norm, International Electrotechnical Commission, 2007.

[RFC97] RFC 2131: Dynamic Host Configuration Protocol. Norm, Internet Engineering Task Force, 1997.

[RFC05] RFC 3927: Dynamic Configuration of IPv4 Link-Local Addresses. Norm, Internet Engineering Task Force, 2005.

[RFC14] RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. Norm, Internet Engineering Task Force, 2014.

[UPn08] UPnP Device Architecture 1.1. Norm, UPnP Forum, 2008.

[USB00] Universal Serial Bus Specification. Norm, USB Implementers Forum, 2000.