Konzept zur automatischen Konfiguration von Feldgeräten in Steuerungssysteme der IEC 61131

Jason Li

25.05.2014

Aufgabenstellung

Titel der Bachelorarbeit:

Konzept zur automatischen Konfiguration von Feldgeräten in Steuerungssysteme der IEC 61131

Inv.-Nr.: 2014/39-BT
Verfasser: Jason Li
Ausgabe: 26.05.2014
Betreuer: Veit Hammerstingl
Abgabe: 27.11.2014

Ausgangssituation:
Mit steigender Variantenvielfalt und kleiner werdenden Losgrößen, müssen Unternehmen ihre Produktionsanlagen wandlungsfähiger gestalten. Hohe Investitionskosten bei der Inbetriebnahme werden dadurch zunehmend ein belastender Kostenfaktor. Dies resultiert aus komplexen Abläufen zur Vernetzung und Konfiguration von Systemen, was derzeit fast ausschließlich durch spezielles Fachpersonal durchgeführt werden muss. Falls Komponenten an einer Anlage getauscht werden müssen (z.B. Defekt oder Umbau), müssen die implementierten Prozessabläufe aufwendig angepasst und das System teil-weise neu konfiguriert werden. Einen Lösungsansatz hierfür bieten Systeme, welche sich selbstständig konfigurieren und hierbei anderen Systemen mitteilen welche Fähigkeiten sie ausüben können. Durch diese automatisierte Vernetzung sollen sich Anlagen in kürzester Zeit aufbauen und verändern lassen ("Plug & Produce"). Bestehende Forschungsansätze nehmen sich dieser Thematik an, basieren jedoch häufig auf nicht-standardisierten Sonderlösungen.

Zielsetzung:
Das Ziel dieser Arbeit ist es, eine automatisierte Vernetzung von einfachen Sensoren und Aktoren in ein übergeordnetes Steuerungssystem (SPS nach IEC 61131) zu ermöglichen. Fokus liegt hierbei auf einer Methodik, welche auf bereits industriell etablierten Komponenten und Architekturen beruht. Hierzu müssen die bisher manuell zur Konfiguration

durchgeführten Schritte analysiert und hieraus Anforderungen für eine Automatisierung abgeleitet werden. Die Automatisierung soll durch ein logisches System realisiert werden, welches mithilfe einer Schnittstelle zur Steuerung auf die Daten der angeschlossenen Peripherie zugreifen kann. Hierzu wird eine Software entwickelt, welche die Funktion einer Steuerung (SPS) anspricht, um an Informationen der Geräte zu kommen. Hierbei ist es unter anderem notwendig, dass die jeweiligen Geräte ihre bereitgestellten Funktionalitäten in das Programmiersystem der Steuerung integrieren.

Vorgehensweise und Arbeitsmethodik:

- Theoretische Einarbeitung in die Konfiguration von Produktionsanlagen
- Technische Einarbeitung in die bestehende Steuerungsarchitektur
- Bestimmung der Arbeitsschritte zur manuellen Integration von Geräten in Steuerungssysteme
- Ableiten von Anforderungen für eine automatisierte Konfiguration
- Analyse der auslesbaren Informationen von Geräten
- Konzept zur Datenbereitstellung und -verarbeitung bei heterogenen Komponenten in der Fabrik
- Entwurf einer Software-Referenzarchitektur zur automatisierten Konfiguration von Steuerungssystemen
- Implementierung der Referenzarchitektur in C# und TwinCat v3
- Erweiterung um herstellerübergreifende Austauschbarkeit von Sensoren
- Evaluation des Konzeptes und der Architektur
- Dokumentation der Arbeit

Vereinbarung:
Mit der Betreuung von Herrn cand.-Ing. Jason Li durch Herrn Dipl.-Ing. Veit Hammerstingl fließt geistiges Eigentum des *iwb* in diese Arbeit ein. Eine Veröffentlichung der Arbeit oder eine Weitergabe an Dritte bedarf der Genehmigung durch den Lehrstuhlinhaber.

Der Archivierung der Arbeit in der *iwb*-eigenen und nur für *iwb*-Mitarbeiter zugänglichen Bibliothek als Bestand und in der digitalen Studienarbeitsdatenbank des *iwb* als PDF-Dokument stimme ich zu.

Garching, den 22.05.2014

Prof. Dr.-Ing.	Dipl.-Ing.	cand.-Ing.
Gunther Reinhart	Veit Hammerstingl	Jason Li

1 Einleitung

1.1 Ausgangssituation

Die herstellende Industrie erlebt sich ändernde Marktverhältnisse. Niedrige Preise und gute Qualität alleine reichen nicht mehr aus, um sich im globalen Wettbewerb zu behaupten. Individualisierung, Variantenvielfalt und kürzere Lieferzeiten gewinnen immer mehr an Bedeutung ebenso wie Flexible Produktionsanlagen, die sich an neue Produkte, Losgrößen und Qualitätsanforderungen etc. anpassen können. Diesen Prozess nennt man Konfiguration. Moderne Produktionsanlagen sind sehr komplex. Dies ist begründet durch die große Vielfalt der Geräte. Jeder Hersteller setzt dabei auf eigene Entwicklungsumgebungen und Schnittstellen, sodass für jeden spezielles Fachwissen benötigt wird. Dadurch entstehen hohe Kosten für die Inbetriebnahme, die die Anschaffungskosten der Anlage übersteigen können, und über die Stückzahl amortisiert werden müssen. Zusätzlich zu den Kosten stellen Standzeiten eine Belastung für Unternehmen dar. (HEDELIND & JACKSON o.D.)

Einen Lösungsansatz hierfür bieten Systemen an, welche sich selbstständig konfigurieren können, um Entwicklungsaufwand und Produktionsausfälle während der Standzeit zu vermindern. Diesen Ansatz nennt man Plug&Produce. Moderne Produktionsanlagen müssen sich heutzutage in bereits existierende IT-Infrastrukturen der Unternehmen integrieren. Die Automatisierungspyramide stellt die verschiedene Ebene der Fertigung dar, die sich in den Anforderungen der Kommunikation unterscheiden.

Während in oberen Unternehmensebenen (Level 3 und 4) die Integration sehr gut gelingt, stellt die Vernetzung in den unteren Ebenen (Level 1 und 2) eine große Herausforderung dar. Die Integration von Sensoren und Aktoren auf der Feldebene ist weiterhin mit großem Aufwand verbunden. (REINHART et al. o.D.)

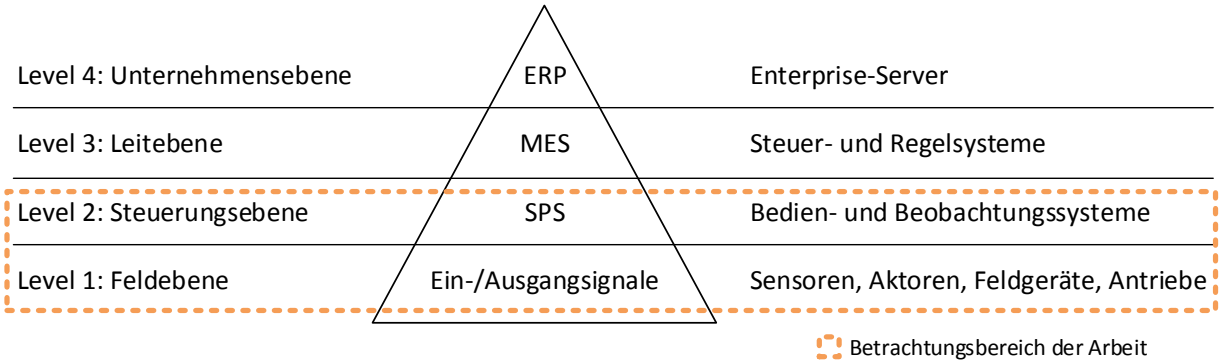


Abbildung 1.1: Automatisierungspyramide - Fokussierung auf die unteren zwei Ebenen

1.2 Zielsetzung

Das Ziel dieser Arbeit ist es, eine automatische Konfiguration und Erkennung von Sensoren und Aktoren an Steuerungssysteme zu entwickeln. Hierzu soll eine Software entwickelt werden, die die Funktion der Feldgeräte nach einer Identifikation in das Programmiersystem integriert. Der Fokus dieser Arbeit liegt auf der Verwendung von industriell bereits etablierten Komponenten und Architekturen. Dabei soll die Konfiguration herstellerunabhängig erfolgen. Ein weiteres Kriterium ist dass der Anwender dafür kein Expertenwissen benötigt.

Beispielhaft erfolgt die Umsetzung in dieser Arbeit anhand IO-Link Reflektionssensoren von Sick und Pepperl&Fuchs, die mit einer Beckhoff SPS vernetzt werden sollen.

Zusammenfassend lassen sich folgende Ziele definieren:

- Automatische Vernetzung von Sensoren und Aktoren mit der Steuerung (SPS)
- Gewinnung von Geräteinformationen nach Anweisung der SPS
- Umwandlung von Funktionen und gerätespezifischen Anweisungen
- Die Konfiguration soll vom Benutzer einfach und ohne technische Kenntnisse durchzuführen sein

1.3 Aufbau der Arbeit

Der Aufbau dieser Arbeit orientiert sich nach der oberen Abbildung dargestellten chronologischen Reihenfolge. In der Vorbereitungsphase wird eine solide Grundlage geschaffen und der aktuelle Stand der Technik beschrieben. In Kapitel zwei wurde das

Abbildung 1.2: Aufbau der Arbeit

der Arbeit vorliegende System genauer betrachtet und in der Funktionsweise sowie Bedienung analysiert. Die dadurch gewonnene Information zum Ablauf der manuellen Konfiguration wurde als Basis für das Konzept im nachfolgenden Kapitel verwendet. Dabei wurde untersucht, welche Möglichkeiten für eine Automatisierung bestehen und ob die Programmierumgebung bereits Schnittstellen besitzt. Die Umsetzung erfolgte anschließend in Kapitel 5. Abschließend wurde betrachtet, ob sich alle Ziele erfüllen ließen und welche Einschränkungen die entwickelte Lösung mit sich bringt (Kapitel 6: Bewertung).

2 Stand der Technik und Forschung

2.1 Stand der Technik

Die Konfiguration von Produktionsanlagen ist heutzutage mit sehr hohen Kosten verbunden und stellt eine interdisziplinare Herausforderung dar. Dies ist begründet durch die Vielfalt der Geräte und die daraus resultierenden Kombinationsmöglichkeiten. Der Techniker benötigt Wissen über Elektronik als auch Mechanik und muss Programmierfähigkeiten aufweisen können. Eine Lösungsmöglichkeit bieten Feldgeräte, die sich selbständig konfigurieren können, sodass der Benutzer keinerlei Fachkenntnisse benötigt (Plug&Produce). Die Idee ist angelehnt an den Mechanismus Plug&Play, das sich bereits in der PC-Welt etabliert hat. (ANTZOULATOS et al. 2014)

ZIMMERMANN et al. (2008) beschreibt drei Arten von Plug&Play:

Hot PnP: Das System ist die ganze Zeit im Betrieb. Geräte können ohne weiteres währenddessen angeschlossen und entfernt werden.

Cold PnP: Erst im ausgeschalteten Zustand können neue Geräte verbunden und entfernt werden. Anschließend wird das gesamte System hochgefahren.

Coordinated PnP: Geräte können nur zu bestimmten Zeiten verändert werden.

Eine andere Klassifizierung beschreibt den Grad der Automatisierung im Plug&Play:

Complete PnP: Das angeschlossene Gerät wird vom System erkannt. Die Gerätebeschreibungen sind in einer Datenbank gespeichert, Treiber automatisch geladen und in die Anwendung integriert. Es sind keine Benutzereingaben notwendig.

Semiautomatic PnP: Der Typ oder die Klasse des Gerätes wird erkannt. Der Benutzer muss Treiber laden. Ansonsten ist keine Konfiguration notwendig.

Configurable PnP: Der Typ oder die Klasse des Gerätes wird erkannt. Der Benutzer muss Treiber laden und manuell konfigurieren.

Die bekannteste Schnittstelle für Plug&Play heißt Universal Serial Bus (USB) und erlaubt hot plugging. Dort können Endanwender jederzeit Geräte wie Maus, Drucker oder Webcam mit dem Computer verbinden. Das Betriebssystem übernimmt alle Aufgaben von Identifizierung, Laden der Treiber bis Bereitstellung der Funktionen für das Programm.

Dieser Mechanismus vom Heimcomputer lässt sich jedoch nicht ohne weiteres auf Produktionsanlagen übertragen.

Der Unterschied von Plug&Produce zu Plug&Play ist die Komplexität. Während sich Computerhardware in einige Klassen einteilen lässt, besitzen Feldgeräte hingegen eine große Vielfalt und endlose Kombinationsmöglichkeiten. Ein Roboter zum Beispiel kann Bewegungen ausführen, und besitzt je nach Aufsatz verschiedene Werkzeuge. Außerdem laufen sie auf verschiedenen Plattformen, verwenden unterschiedliche Protokolle und spezielle Verkabelungen. Des weiteren besitzt jeder Hersteller eigene Programmierumgebungen, die meistens graphisch sind. Oft müssen noch einzelne Messwerte auf Bit-Ebene einem Typ zugeordnet werden, die dann im Programmkontext verknüpft werden. Dazu benötigt man nicht nur Kenntnisse über den Prozess, sondern auch über die Geräte, Sensoren und Informatik.

2.2 Stand der Forschung

Es existieren weitere Forschungsprojekte, welche sich ebenfalls mit dem Plug&Produce von Feldgeräten beschäftigen.

HODEK & SCHLICK (o.D.) beschreibt die automatische Integration von Geräten durch Profile und definiert die These, dass Geräte des gleichen Typs sich in Funktionen und Parametern ähnlich sind. Dies kann nur erreicht werden durch die Entwicklung eines standardisiertes (herstellerunabhängiges) Profil. Die Feldgeräte werden dazu mit einem Mikrocontroller ausgestattet, welche einen Webserver bereitstellen, damit das Gerät über Ethernet in den Produktionsablauf integriert werden kann.

Einen anderen Ansatz verfolgt REINHART et al. (o.D.) bei dem intelligente Sensoren einen Webserver besitzen und Geräteinformation sowie Funktionen in Ethernet verfügbar stellen. Diese Information werden von einem Konfigurationssoftware eingelesen und in die Applikation integriert.

Anstatt, dass jedes Feldgerät einen Webserver bereitstellt, fasst DURKOP et al. (o.D.) mehrere Geräte zu einer Modulgruppe zusammen, innerhalb denen Echtzeitkommuni-

kation besteht. Dabei erhält jedes Gerät einen OPC UA Server, die von den jeweiligen Modulsteuerungen erkannt und integriert werden. Die Verbindung der Modulgruppen nach außen erfolgt über einen Webserver.

Die hier dargestellten Konzepte bieten einen guten Ansatz für Plug&Produce. Allerdings werden die Feldgeräten modifiziert und mit zusätzlichen Komponenten, um neue Funktionen und Schnittstellen zu implementieren. Dies führt zu zusätzlichen Aufwand und wird aktuell in der Praxis kaum eingesetzt. Ein Konzept, welches diese Einschränkungen umgeht könnte Abhilfe schaffen. (HAMMERSTINGL & REINHART o.D.)

3 Analyse

3.1 Allgemein

Ein *Automat* ist eine Maschine, das selbstbestimmt arbeitet. Ein automatisierter Prozess ist demnach ein technischer Vorgang (z.B. Montage, Fertigung, Chemischer Prozess), der "mit elektrischen, mechanischen, pneumatischen oder hydraulischen Einrichtungen"(LAUBER & GÖHNER 1999) selbständig arbeitet.

Der *technische Prozess* findet dabei innerhalb einer *technischen Anlage* statt, welche von ihr beeinflusst wird. In der Chemieindustrie findet die Herstellung von Spülmittel als chemischer Mischvorgang in Reaktoren statt. Diese Reaktoren bestimmen maßgeblich den Herstellungsprozess, indem die Edukte im Reaktor vermischt und anschließend umgesetzt werden.

Eine technische Anlage kann in drei Komponenten aufgeteilt werden:

- Der technische Prozess, welcher in der Anlage abläuft
- Feldgeräte wie Sensoren und Aktoren, die den Prozess beobachten bzw. beeinflussen. Während Sensoren Messwerte über den Prozess liefern, in dem physikalische Eingangsgrößen (wie Temperatur) in Information für die Steuerung umgewandelt werden, führen Aktoren dem Prozess mechanische Energie zu.
- Steuerungssysteme sind Computersysteme, die ein Automatisierungsprogramm abarbeiten, welches aus Eingangssignalen entsprechende Ausgangssignale berechnen. Dies sind unter anderem Mikrocontroller, Speicherprogrammierbare Steuerung (SPS) oder PCs. Außerdem ist das Steuerungssystem mit der weiterführenden Infrastruktur im Unternehmen verbunden (z.B. mit dem Leitstand zur Überwachung der Anlage).

Im Rahmen dieser Arbeit wird die automatische Konfiguration von Steuerungssysteme bezüglich der Verbindung von Feldgeräten näher untersucht.

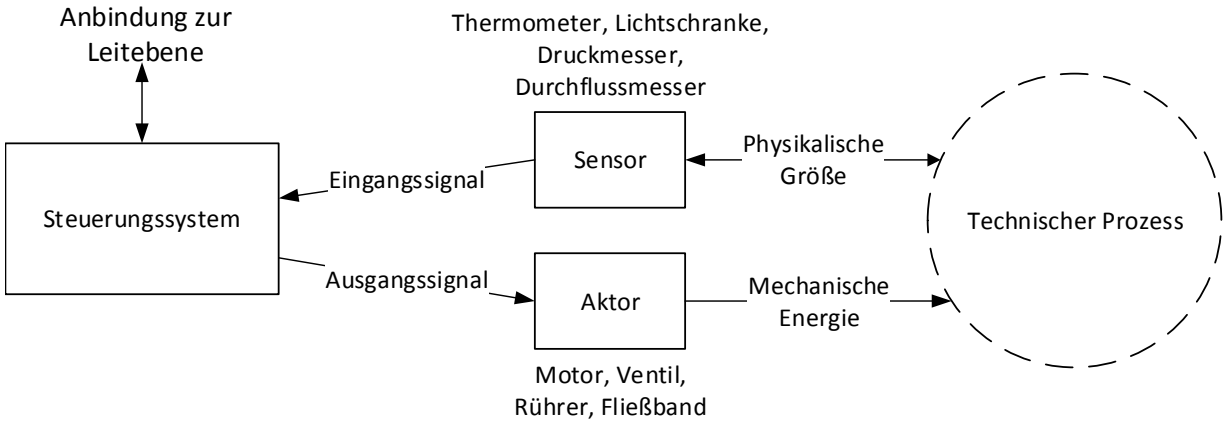


Abbildung 3.1: Übersicht der verschiedenen Komponenten einer technischen Anlage

3.2 Sensoren und Aktoren

Um einen technische Prozess durchführen zu können, sind Informationen über die Prozessgrößen (wie Druck, Temperatur, Durchfluss) notwendig. Mit Hilfe von *Sensoren* werden diese physikalischen Größen erfasst und in elektrische Signale umgewandelt, damit diese vom Steuerungssystem verarbeitet werden können.

Umgekehrt übermittelt das Steuerungssystem Signale an die *Aktoren*, um diese physikalische Größen zu verändern (z.B. öffnen und schließen eines Ventils).

Sensoren und Aktoren werden auch *Feldgeräte* genannt, da diese mit dem Prozess direkt in Kontakt stehen und sich im *Feld* befinden, was den Bereich außerhalb von Schaltschränken bezeichnet.

Zusätzlich zur Erfassung von physikalischen Größen kann ein Sensor auch das Signal aufbereiten oder vorverarbeiten. Dafür können drei Typen von Sensoren benannt werden:

Elementarsensor Nimmt eine physikalische Messgröße auf und bildet dieses als elektrisches Signal ab. Je ausgeprägter die Größe, desto höher die Spannung bzw die Stromstärke.

Integrierter Sensor Bereitet zusätzlich das Signal auf, indem dieses verstärkt, linearisiert, normiert oder gefiltert wird, sodass diese besser verarbeitet werden können.

Intelligenter Sensor Zusätzliche Intelligenz im Sensor erlaubt die Durchführung von Signalverarbeitung: z.B. Min und Max-Wert Überwachung, Umrechnung, Fehlerdiagnose. Die Intelligenz des Sensors gibt eine verarbeitete Größe aus, eine

direkte Steuerinformation. Außerdem sind sie oft in der Lage sich selber beim Steuerungsgerät zu identifizieren, was für die automatische Konfiguration sehr nützlich ist.

Geräte, die sich nicht selbständig identifizieren können, werden vom Entwickler identifiziert, was durch die Angabe von Hersteller, Modell und Revision geschehen kann. (LAUBER & GÖHNER 1999)

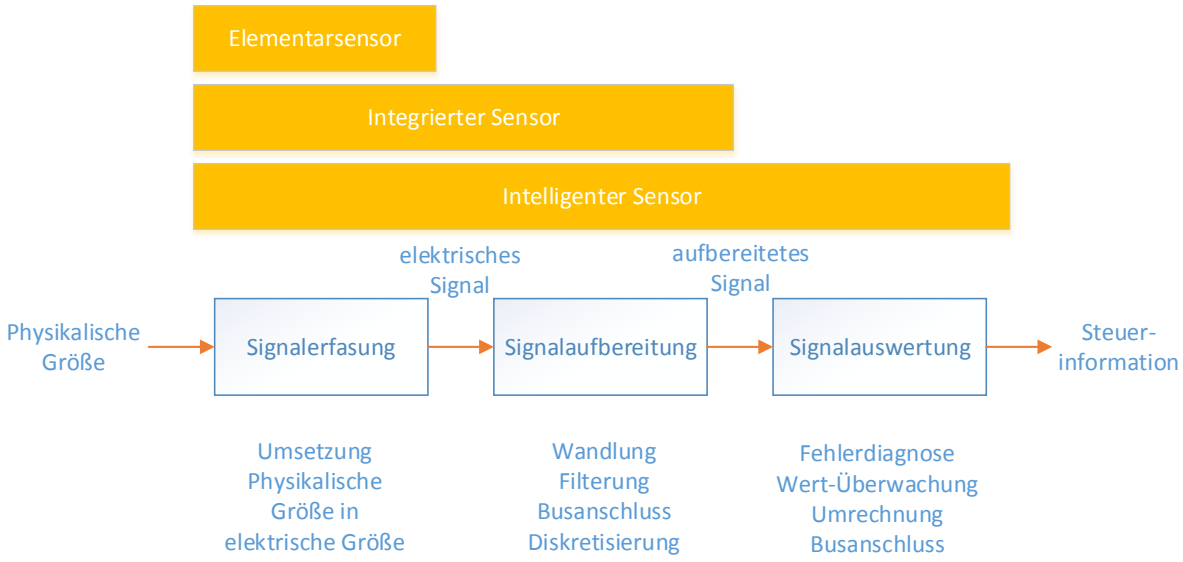


Abbildung 3.2: Vergleich verschiedener Sensorarten

3.3 Kommunikationssysteme

Je nach Typ werden diese Sensoren unterschiedlich an das Steuerungssystem angeschlossen. Im Gegensatz zu elementaren und integrierten Sensoren, die häufig über zwei Adern direkt an das Steuerungssystem angeschlossen werden, bieten intelligente Sensoren eine hohe Auswahl an Schnittstellen zur Kommunikation. Das können unter anderem aus dem PC-Bereich bekannten Anschlussarten wie USB, FireWire, Bluetooth sein, als auch Feldbussysteme wie CAN, Ethernet und Profibus sein oder andere Kommunikationssysteme.

Bei der direkten Verbindung an das Steuerungssystem (z.B. über zwei Adern), muss von jedem Feldgerät eine eigene Leitung verlegt werden (Punkt zu Punkt Verbindung). Mit zunehmender Automatisierung erhöht sich die Verkabelungsarbeit bei der parallelen

Verbindung aufgrund der wachsenden Anzahl von Feldgeräten. Der dadurch entstehende Aufwand und zeichnet sich durch hohe Kosten bei der Konfiguration aus. Um dem entgegenzuwirken, werden zunehmend Feldbussysteme eingesetzt, welche wesentlich kostengünstiger sind, welche multiple Anschließung von Feldgeräten an eine Leitung ermöglichen.

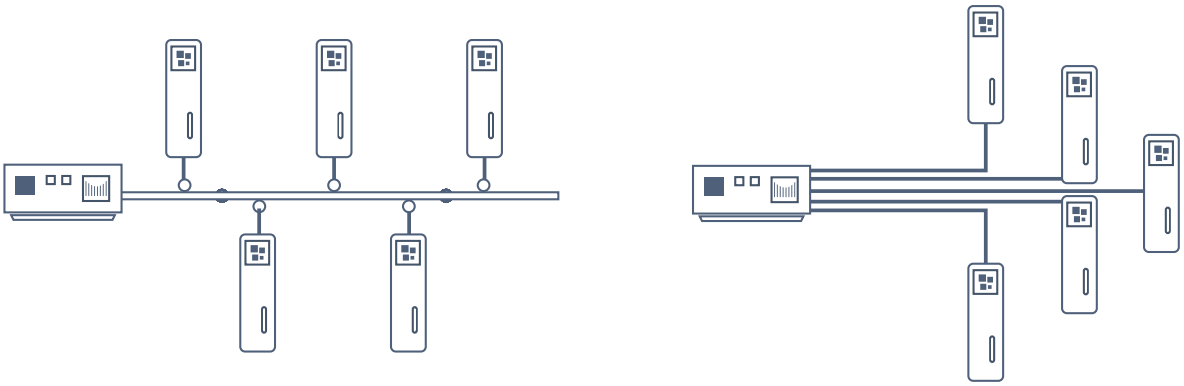


Abbildung 3.3: Vergleich Verkabelung von Feldgeräten: Seriell mit Feldbus (links) und Parallel (rechts)

Punkt-zu-Punkt-Verbindung Vom Steuerungsgerät geht zu jeden Feldgerät eine Leitung. (Parallele Verbindung)

Broadcast-Netz Vom Steuerungsgerät geht nur eine Leitung aus, welche an allen Feldgeräten vorbeigeführt wird. Sie teilen sich alle diese Leitung (Serielle Verbindung)

An diese Feldbusse werden hohe Anforderungen gestellt, da automatisierte Prozesse eine bestimmte Reaktionszeit (Echtzeitfähigkeit) verlangen. Deshalb ist es wichtig, dass genau definiert wird wer (Identifikation), wann (Initiative, Reihenfolge), was (Messwert, Befehl) über die Leitung senden darf und was passiert, wenn es zwei Geräte gleichzeitig senden. Dafür existieren je nach Einsatzbereich (für explosionsgefährdete Bereiche z.B. Profibus PA mit Leistungsbegrenzung) und Übertragungsleistung, eine Vielzahl an genormten Protokollen.

Ein Auszug von gängigen Feldbussen:

- Ethernet
- Profibus PA
- Profibus DP
- EtherCAT

- Profinet

Für die Produzenten stellt das eine große Herausforderung dar, da Sie mehrere Protokolle unterstützen müssen. Für die Konfiguration von Steuerungsgerten bedeutet das, dass mehrere Protokolle unterstützt und bei der Konfiguration nach Geräten durchsucht werden müssen. Im einen Netzwerk können auch unterschiedliche Protokolle verwendet werden, für die Umwandlung der Information sorgen dafür entsprechende *Master*.

3.4 Manuelle Konfiguration

Damit Feldgeräte in der technischen Anlage arbeiten können, müssen diese im Programm integriert und korrekt eingestellt sein. Eine Lichtschranke zum Beispiel liefert einen Messwert, ob sich ein Objekt davor befindet oder nicht. Der Wert des Sensors muss demnach mit einer Variablen im Programm verknüpft werden, damit diese auch verwendet werden kann. Zusätzlich muss die Lichtschranke eingestellt werden, welche Reichweite abgetastet wird, damit nur Objekte in einem bestimmten Bereich werden sollen (Parametrisierung). Im Folgenden werden alle nötigen Schritte vorgestellt. Sie zeigen, dass die Konfiguration mit sehr hohem personellen Aufwand verbunden ist.

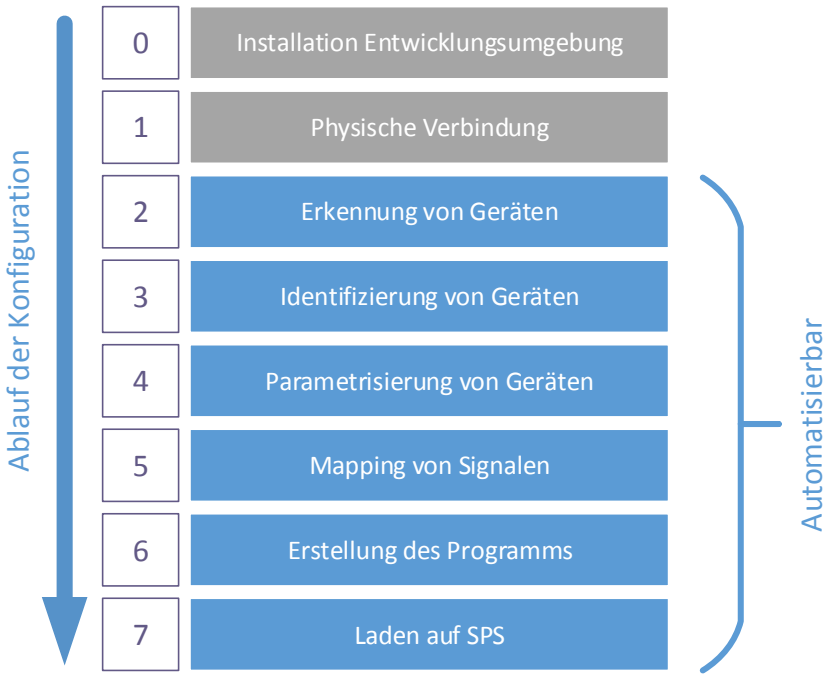


Abbildung 3.4: Übersicht der Arbeitsschritten bei einer manuellen Konfiguration

Entwicklungsumgebung
Steuerungssysteme werden in der Regel nicht direkt programmiert, sondern werden von einem Arbeitsrechner aus mit Programmen geladen. Dies geschieht mit Hilfe einer Entwicklungsumgebung, die sich je nach Steuerungssystemhersteller unterscheiden und andere Schnittstellen mit sich bringen. Auf dem Arbeitsrechner wird dafür die korrekte Software installiert, da in der Regel nur diese auf das Steuerungssystem zugreifen kann.

Physische Verbindung
Zunächst werden die Geräten mit dem Steuerungssystem verbunden. Das kann entweder durch eine analoge elektrische Verbindung über zwei Adern sein oder einen Feldbus Anschluss.

Geräteerkennung
Bevor eine Kommunikation stattfinden kann, wird das Gerät vorher vom Steuerungssystem erkannt. Konventionell trägt der Entwickler die Adresse im Netzwerk ein oder gibt, an welchen Anschlüssen sich Geräte befinden. Moderne Feldbusse sind allerdings in der Lage, nach Geräten zu suchen.

Kommunikation herstellen
Nachdem das Gerät identifiziert ist, wird die Signalkodierung eingestellt, damit sich Steuerungssystem und Gerät verstehen. Hierfür werden Datentyp und Bit-Länge angegeben. Hersteller bieten dafür Beschreibungsdateien, die diese Informationen enthalten. Die Gerätebeschreibungsdatei für die passende Verbindung und Modell werden in die Entwicklungssoftware geladen.

Parametrisierung
In diesem Schritt werden die Geräte für den Anwendungszweck konfiguriert, wie z.B. die Empfindlichkeit eines Reflektionssensors.

Mapping Signale
In diesem Schritt verknüpft der Entwickler die Ein-/Ausgangssignale der Feldgeräte mit Variablen im Automatisierungsprogramm.

Programmerstellung
Liegen alle benötigten Informationen vor, wird das eigentliche Programm erstellt. Das kann durch den Einsatz verschiedener Programmiersprachen und Ablaufplänen erfolgen. Aus logischen Funktionen werden die Ein- und Ausgangssignale gekoppelt.

Laden der Konfiguration
Im letzten Schritt wird die vollständige Konfiguration mit Hilfe der Entwicklungsumgebung auf das Steuerungssystem geladen.

3.5 Anforderungen an die Automatisierung

Die Automatische Konfiguration soll die Aufgaben des Automatisierungstechnikers übernehmen. Dafür muss sie zum einen robust sein und Fehlern erkennen können. Dazu gibt es verschiedene Methoden, automatisierte Befehle durchzuführen, die sich jedoch in der Funktionsmöglichkeit und Robustheit unterscheiden. Beckhoff TwinCAT bietet zum einen das Automation Interface und ADS an, mit der man Feldgeräte konfigurieren kann. In der aktuellen Version sind diese von der Funktionalität sehr eingeschränkt. Viele Funktionen, die man manuell über die GUI ausführen kann sind nicht möglich, sodass bei dieser Arbeit auf zusätzliche Methoden zurückgegriffen werden muss. Die physikalische Verbindung muss erfolgt sein.

4 Konzept

4.1 Allgemein

In diesem Kapitel werden die einzelnen Schritte der manuelle Konfiguration näher betrachtet und ein Konzept entwickelt, um sie automatisiert durchzuführen. Auf zwei Möglichkeiten lässt sich eine Automatisierung naheliegend umsetzen:

- Die Verwendung von Programmschnittstellen in der Entwicklungsumgebung des Herstellers. Im Optimalfall direkten Zugriff auf Programmfunktionen. Diese sind bevorzugt zu verwenden, da sie vom Benutzersystem unabhängig sind und vom Hersteller bereitgestellt werden, sodass sie am ehesten über verschiedene Versionen aufrechterhalten werden.
- Falls Programmfunktionen nur über die graphische Benutzeroberfläche (GUI) zu erreichen sind, werden diese über simulierte Benutzereingaben angesteuert. Maus und Tastatur werden von einem Programm verwendet, als ob sie ein Entwickler steuern würde. Bei einer Veränderung der GUI (z.B. bei einer neuen Softwareversion) muss die Automatisierung ebenfalls angepasst werden.

4.2 Programmstruktur

Das Ziel dieser Arbeit ist es, die automatische Erkennung von Geräten und die Bereitstellung deren Funktionen zu ermöglichen, unabhängig von Hersteller und Typ. Die aus der Analyse gewonnenen Handlungsfolge ist universal anwendbar. Aufgrund der Vielfalt von Steuerungssystemen und Verbindungen mit eigener Kodierung und Schnittstellen ist es notwendig, diese universale Handlungsfolge in gerätespezifische Befehle zu übersetzen. Diese Übersetzung wird durch sogenannte *Treiber* durchgeführt. Die Gerätetreiber wandeln allgemeine Funktionen, die in dem wiederverwendbaren Ablauf des Programms definiert sind, in die Befehle um, die das jeweilige Gerät verstehen kann. Es erfolgt eine *Abstraktion*. Der Anwender benötigt in diesem Fall keine tiefgreifende

Fachkenntnisse in die spezifische Entwicklungsumgebung oder Gerätearchitektur. Diese Arbeit wird durch das Plug&Produce Programm erledigt.

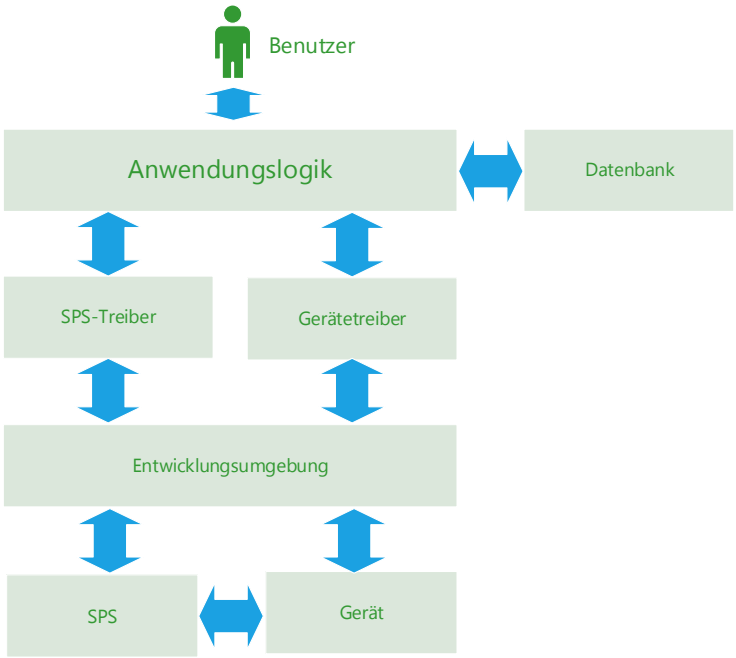


Abbildung 4.1: Strukturelle Aufbau des Programms im Interaktion mit dem Benutzer

Folglicherweise wird das Programm in verschiedene Module unterteilt. Die allgemeingültigen Befehle des Benutzers werden von der Anwendungslogik des Programms verarbeitet. Um mit den Geräten zu kommunizieren, werden diese mit Hilfe einer Datenbank identifiziert. Passende Treiber für die Steuerung (SPS) sowie dem angeschlossenen Gerät werden geladen, welche die Befehle des Benutzers umwandeln und über die Entwicklungsumgebung an die Hardware weiterleiten.

4.3 Programmablauf

Der Ablauf der Automatisierung, angelehnt an HAMMERSTINGL & REINHART (o.D.), orientiert sich an denen der manuellen Konfiguration. Nach der Strukturierung des Programms ist es möglich, detaillierte Verläufe zwischen den Komponenten zu definieren. Hinzugekommen ist dabei die Suche nach der SPS, welcher im Unternehmensnetzwerk verbunden ist, und zuerst als Zielsystem eingestellt werden muss.

Um mit der SPS kommunizieren zu können, wird aus der Datenbank der benötigte Treiber geladen. Darüber kann die Geschäftslogik die SPS befehlen nach Geräten zu suchen, welche mit der Datenbank abgeglichen werden und entsprechende Treiber zur

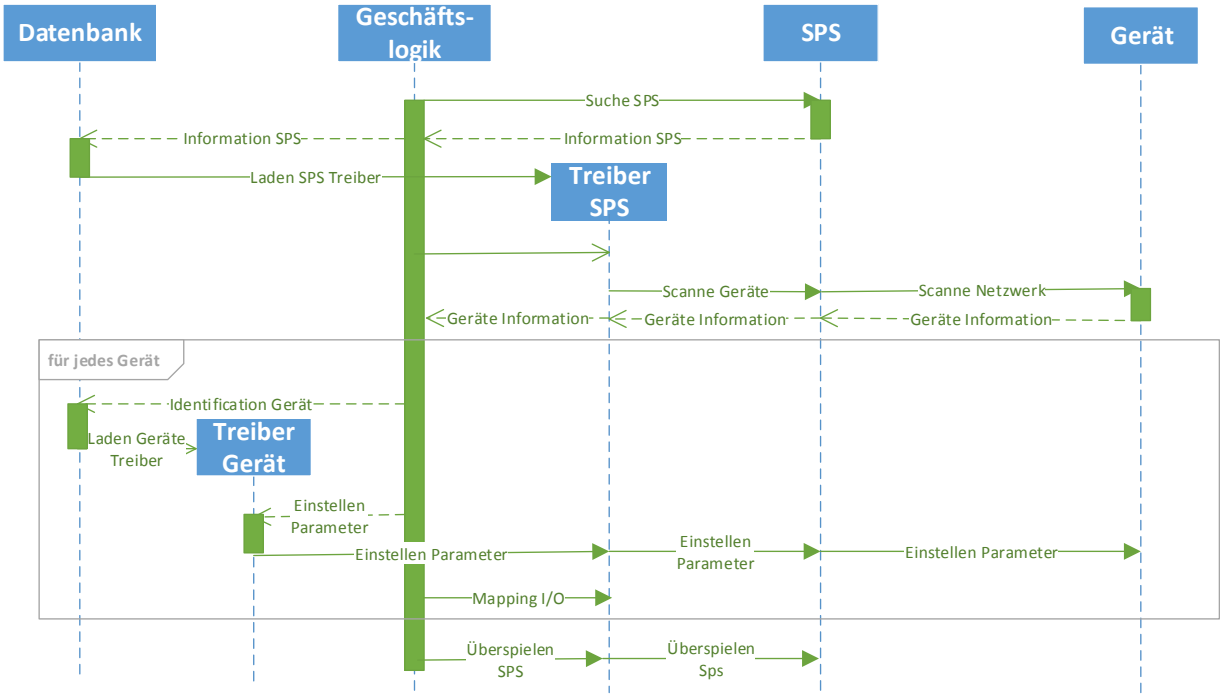


Abbildung 4.2: Allgemeiner Ablauf einer automatisierten Konfiguration

Integration und Parametrisierung geladen. Zuletzt wird die Konfiguration über den SPS Treiber überspielt.

5 Umsetzung

5.1 Allgemein

Das Konzept aus Kapitel 4 wird nun beispielhaft anhand einer Beckhoff SPS und IO-Link Sensoren als eine C# Anwendung umgesetzt. Einführend werden die verwendete Hardware und physikalische Verknüpfung benannt. Beispielhaft wird die manuellen Konfiguration die benötigten Schritte für die Automatisierung abgeleitet. Da diese Arbeit auf die Umsetzung der Konfigurationsmethoden fokussiert ist, werden im darauffolgenden Teil verschiedene Möglichkeiten dafür vorgestellt und ihre Vor- und Nachteile ausgearbeitet. Zum Schluss wird das fertige Graphical User Interface (GUI) vorgestellt und die Bedienung kurz erklärt.

Die der Arbeit vorliegende Hardware Konfiguration ist wie folgt:

- Beckhoff CX2030 - Speicherprogrammierbare Steuerung (SPS). Betrieben mit Windows 7 und TwinCAT Runtime
- Beckhoff IO-Link Master EP6224-2022 - Anschluss für IO-Link Sensoren
- Sick WTB4SC-3P3452V - Reflexionssensor
- Pepperl&Fuchs MLV41-8-H-500-RT - Reflexionssensor

Das IO-Link Master ist dabei über dem EtherCAT Feldbus mit dem Steuerungssystem verbunden.

5.2 Manuelle Konfiguration

Es gilt nun, die manuelle Konfiguration mit der vorliegenden Hardware durchzuspielen.

Entwicklungsumgebung

Die Entwicklungsumgebung *TwinCAT XAE (eXtended Automation Engineering)* ist vollständig in *Microsoft Visual Studio* integriert. Die TwinCAT Konfiguration besitzt eine

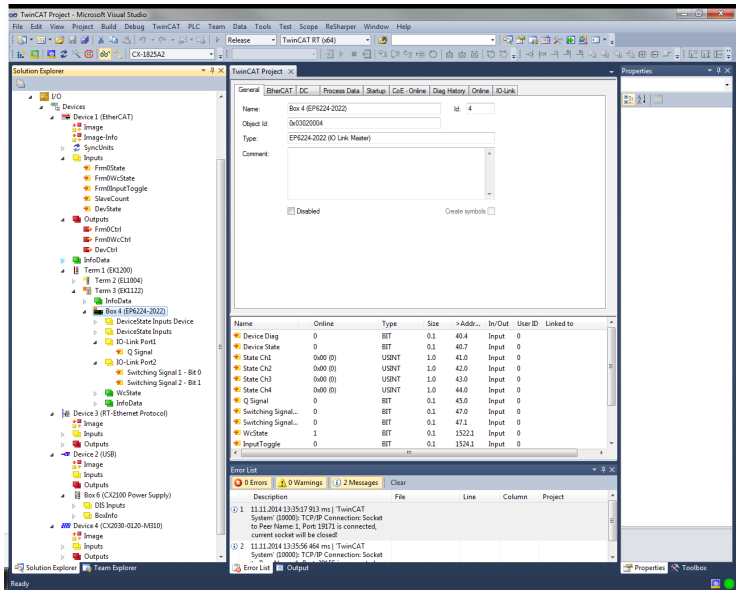


Abbildung 5.1: Screenshot von TwinCAT XAE integriert in VS 2010

Baumstruktur, in der alle Informationen wie Logik-Programme und angeschlossene Geräte gespeichert werden.

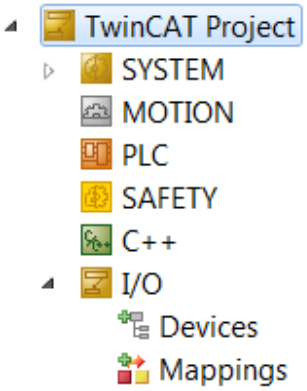


Abbildung 5.2: Informationsbaum des TwinCAT Projekts

Auf der SPS läuft das Gegenstück TwinCAT Runtime, welche in der Lage ist die von TwinCAT XAE generierte Konfiguration auszuführen. Für die Programmierung stehen die Sprachen, die in IEC 61131 definiert sind, sowie C++ und Matlab/Simulink zur Verfügung.

Auf dem Arbeitscomputer ist TwinCAT mit Visual Studio 2010 in englischer Sprache installiert, da dies für die Funktion notwendig ist. Nach der Installation werden die EtherCAT Gerätebeschreibungsdateien von der Website des Herstellers aktualisiert, da ansonsten Fehler in der Kommunikation auftreten können.

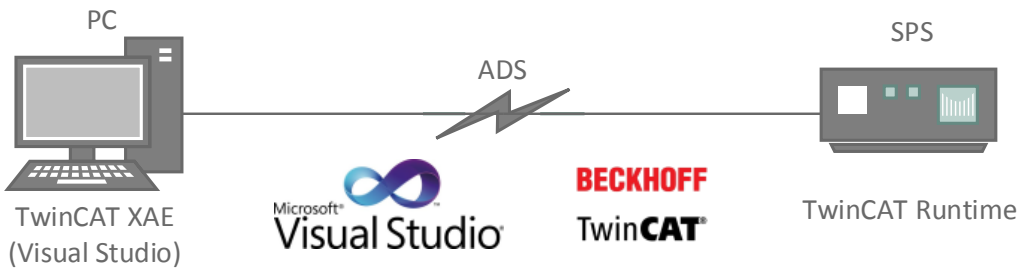


Abbildung 5.3: Aufbau Beckhoff Automatisierungslösung

Der Informationsaustausch zwischen den einzelnen Komponenten (Steuerung, Feldgeräte, Arbeitsrechner) erfolgt über die universale Kommunikationsplattform ADS, welches das Auslesen und Schreiben von Parametern und Betriebsdaten ermöglicht.

Physische Verbindung
Bevor die eigentliche Konfiguration beginnen kann, muss die SPS mit Strom versorgt und die benötigten Anschlusskarten und IO-Link Master sowie Sensoren miteinander verbunden werden.

Kommunikation herstellen
Zuerst wird die SPS als Zielsystem für die Konfiguration eingestellt. Die Netzwerkanbindung der SPS erfolgt über eine Ethernet-Verbindung. Über eine Suche werden alle im Netzwerk befindlichen TwinCAT Geräten angezeigt. Nach der Auswahl der SPS und der Eingaben von Kennungsdaten ist sie als Route am Entwicklungsrechner spezifiziert.

Abbildung 5.4: Toolbar Funktionen der manuellen Konfiguration (1) Konfiguration aktivieren, (2) Neustart (Run-Mode), (3) Neustart (Config-Mode), (4) Geräte neu laden, (5) Geräte scannen, (6) FreeRun aktivieren, (7) Online Daten anzeigen, (8) Zielsystem Auswahl

Erkennung von Geräten
Die Erkennung der Hardware erfolgt bei TwinCAT vollautomatisch. In Visual Studio wird dazu die Scan-Funktion aktiviert. Voraussetzung dafür ist, dass sich die SPS sich im Config-Mode befindet. TwinCAT unterscheidet zwischen zwei Betriebsmodi, Config- und Run-Mode:

Config-Mode: Im Config Mode befindet sich die SPS im Wartungsmodus. ADS Server sind nicht verfügbar. Für die Scan-Funktion wird Config Mode vorausgesetzt. Nach

der erfolgreichen Erkennung kann zusätzlich der Free-Run aktiviert werden und IOs geladen werden. Dabei können Sensoren und Aktoren geprüft und eingestellt werden.

Run-Mode: Im Run-Mode befndet sich die SPS im Betrieb. SPS Programme können nun gestartet werden, alle verwendeten Komponenten sind aktiv. Sie stellen außerdem einen ADS Server für Informationsaustausch zur Verfügung.

Erkennung und Parametrisierung von IO-Link Sensoren

Wird der IO-Link Master erkannt, so kann man diesen im Strukturbaum auswählen und nach angeschlossenen Sensoren suchen. Dazu wird der *Scan* Button getätigt und anhand der Vendor- und Device-IDs die Sensoren identifiziert. Die Gerätebeschreibungsdateien werden von TwinCAT geladen, wenn sich diese im Repository der Entwicklungsumgebung befinden.

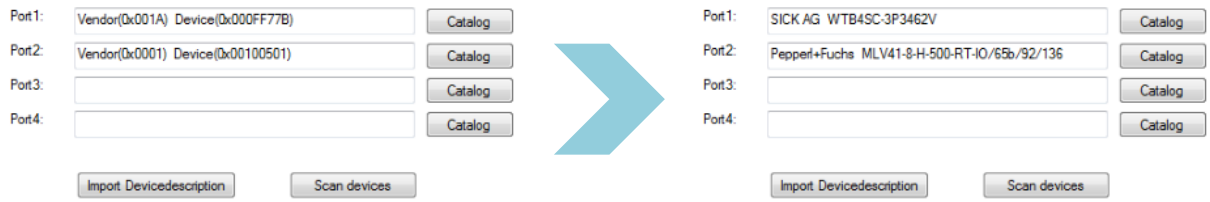


Abbildung 5.5: Scan der IO-Link Sensoren ohne passende Beschreibungsdatei (links) und mit (rechts)

Die Parametrisierung erfolgt ebenfalls über die grafische Oberfläche beim IO-Link Master.

Mapping von Variablen

Beim Mapping entfällt die Typisierung auf Bit-Ebene, da IO-Link Feldgeräte intelligente Sensoren sind und somit diese Arbeit durch die Beschreibungsdatei abgenommen wird. Der Entwickler muss nur noch die Eingängen des Sensors mit Programmvariablen verknüpfen.

SPS Starten

Nach Beendigung der Konfiguration und die Anlage wird über *Konfiguration aktivieren* gestartet.

5.3 Schnittstellen

Um die Konfiguration durchzuführen werden nun die Möglichkeiten für die Automatisierung bei TwinCAT näher untersucht. Der Hersteller bietet bereits eine Schnittstelle für die automatische Erstellung von Konfigurationen an, die aufgrund des begrenzten Funktionsumfangs ergänzt wird. Hierfür wird einerseits die Kommunikationsplattform ADS zum Parametrisieren verwendet. Mit Hilfe von simulierten Benutzereingaben werden Funktionen implementiert - analog zur manuellen Konfiguration über die grafische Oberfläche. Dazu werden Macros in Visual Studio ergänzend mit AutoIt verwendet.

5.3.1 Automation Interface

Automation Interface ist Schnittstelle für die automatische Erstellung von TwinCAT Konfigurationen via Skript. Sie bietet Bibliotheken für diverse Plattformen an. Im Rahmen dieser Arbeit wird die C# Bibliothek verwendet, die ebenfalls in Visual Studio integriert ist. Neben einigen direkten Grundfunktionen (Erstellen der Konfiguration, Starten der SPS, Verlinken von Variablen, Laden von POUs) besteht die hauptsächliche Aufgabe im Lesen und Schreiben des Datenbaums in TwinCat. Somit lassen sich nahezu alle Informationen extrahieren.

Insgesamt kann man Funktionen auf zwei Arten aufrufen:

- Einbinden der Klasse und direkten Aufruf der Funktion über die Bibliothek

```
1  _SysManager.ActivateConfiguration();
```

- Aktivieren (auf Eins setzen) eines Trigger-Elementes im Datenbaum durch XML Eingabe

```
1  <TreeItem><DeviceDef><ScanBoxes>1</ScanBoxes></DeviceDef></TreeItem>
```

Das Automation Interface ist für die Verwendung vor der Laufzeit vorgesehen. Der Aufgabenbereich erstreckt sich über Geräte scannen/erstellen, SPS Programme laden, Variablen verknüpfen und SPS (neu)starten. Vorteilhaft ist, dass diese vom Hersteller angeboten werden und somit über lange Zeit erhalten bleiben. Die aktuelle Version ist jedoch relativ begrenzt in der Funktionalität.

5.3.2 ADS

ADS (Automation Device Specification) ist ein universell geräteunabhängiges Nachrichtenprotokoll. Jedes mit der Steuerung verbundenes Gerät bekommt eine sogenannte *NetId* zugewiesen. Unter dieser Adresse wird vom Gerät zur Laufzeit jeweils ein ADS Server zum Informationsaustausch bereitgestellt. Auf diese Weise lassen sich Variablen auslesen und Parameter ändern. Diese Schnittstelle ist unabhängig von Visual Studio.

5.3.3 Makro und AutoIt

Alle andere Funktionen, die nicht über Automation Interface oder ADS verfügbar sind, werden in dieser Arbeit mit simulierten Eingaben über die GUI bedient. In Visual Studio können einige Buttons und Funktionen über Makros aufgerufen werden. Diese Variante übernimmt die direkte Aktivierung von Visual Studio Funktionen, falls diese direkt aufrufbar sind. Wenn nicht, werden diese Funktionen über den manuellen Zugriff per Button und Mausklick aktiviert, wie sie beim Scannen von IO Link Geräten der Fall ist.

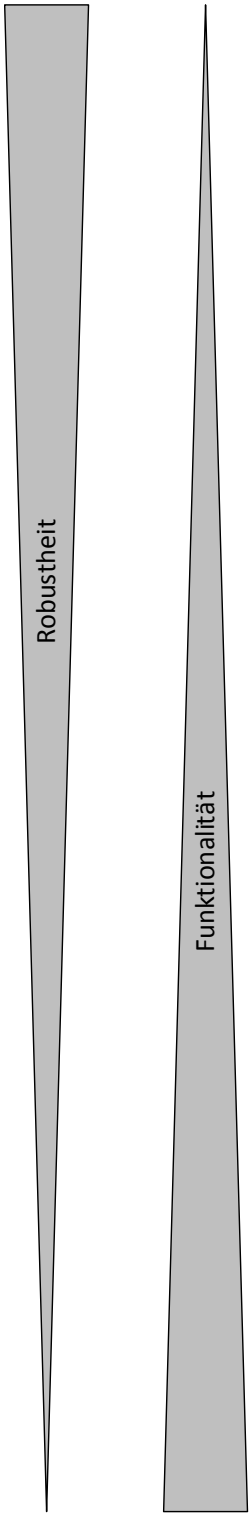
5.4 Programmstruktur

Das Program, genannt *Javelin*, zielt darauf ab eine TwinCAT Konfiguration zu erstellen. Sie manipuliert Visual Studio auf drei verschiedenen Wegen: (1) Automation Interface, (2) VS Makro und (3) AutoIt.

Treiber und Geschäftslogik werden hier zusammenfassend als die Javelin Anwendung dargestellt. Sie erstellt utner Verwendung von AutoIt, Makros sowie Automation Interface über Visual Studio eine TwinCAT Konfiguration und überspielt diese auf die SPS.

5.4.1 Multithreading

Um die Bedienbarkeit der Oberfläche zu gewährleisten ist die Software in mehreren Threads (Anwendungsstränge) aufgeteilt. Die grafische Oberfläche läuft somit unabhängig von der Ausführung der Konfiguration. Deren Ausführung in einem gemeinsamen Thread hätte ein Einfrieren der Oberfläche zur Folge.



Automation Interface	Generierung der TwinCAT Konfiguration Anlegen/Speichern von Projekten Suchen von Zielrechner Scannen nach Geräten Auslesen der Geräteinformation + Direkte Steuerung von TwinCAT XAE + Hersteller API, versionsübergreifend + Sehr robust − Geringer Funktionsumfang
ADS	Austausch von Laufzeitinformationen zwischen Komponenten Schreiben/Lesen IO-Link Parametern Schreiben/Lesen von SPS Variablen Schreiben/Lesen von Zuständen + Versionsübergreifend − Erst während Laufzeit verfügbar
Visual Studio Makro	Zugriff zu TwinCAT XAE Funktionen, die nicht über AI direkt verfügbar sind Neustarten von SPS Navigation in der VS GUI (TreeView) + Robuste Steuerung von Visual Studio − Sprachabhängig − Versionsabhängig
AutoIT	Simulation von Benutzereingaben Navigation in der VS GUI Verwerfen von Bestätigungsdialoge Scannen von IO-Link Geräten − GUI abhängig − Benutzereingaben während Programmablauf gefährden Funktion + Sehr flexibel

Abbildung 5.6: Übersicht und Vergleich der verfügbaren Schnittstellen

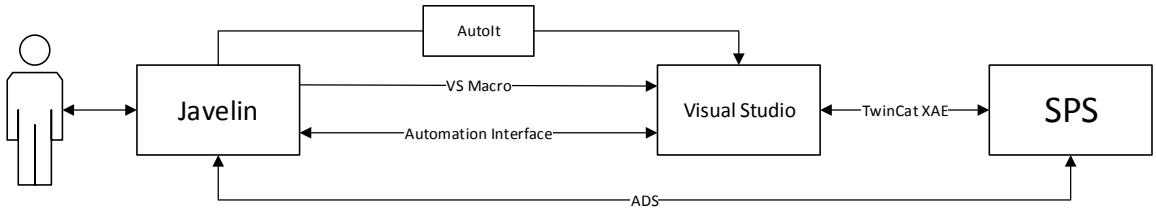


Abbildung 5.7: Funktionelle Struktur von Javelin

Für die Abarbeitung der Automatisierung wurde ein Hintergrunds-Thread implementiert, der sogenannte *JavelinBackgroundWorker*. Es handelt sich um einen Thread des Typs STA (Single Thread Apartment). Somit wird Visual Studio, einem eigenständigen Prozess, die Möglichkeit gegeben, ber die COM-Schnittstelle die Oberfläche über den Stand der Ausführung zu informieren. Falls benötigt werden einzelne AutoIt Prozesse aufgerufen, um Benutzereingaben zu simulieren.

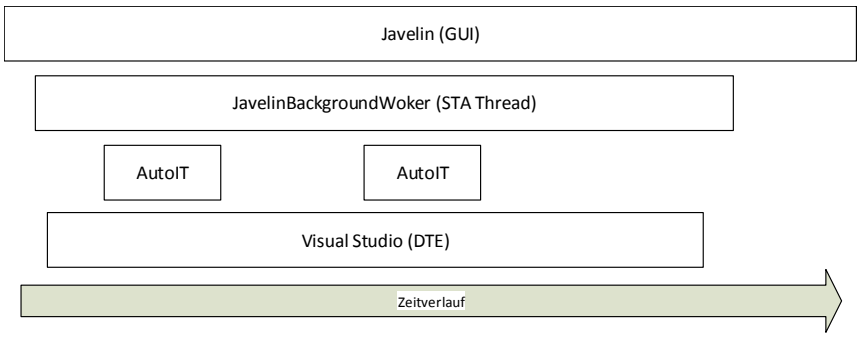


Abbildung 5.8: Aktive Threads über Zeit

5.5 Automatische Konfiguration

5.5.1 Übersicht

Nachdem die physikalische Verbindung abgeschlossen ist, übernimmt die Javelin Software die Einrichtung. Im Vergleich zur manuellen Konfiguration wird das System erstmals initialisiert und ein Zielsystem ausgewählt. Die Parametrisierung erfolgt nun im letzten Schritt aufgrund der Verwendung von ADS, welche erst während der Laufzeit verfügbar ist.

1. Initialisierung: Am Anfang wird ein neues Projekt in Visual Studio anhand der TwinCAT Vorlage erstellt. Die SPS wird in den Config Mode geschaltet und die

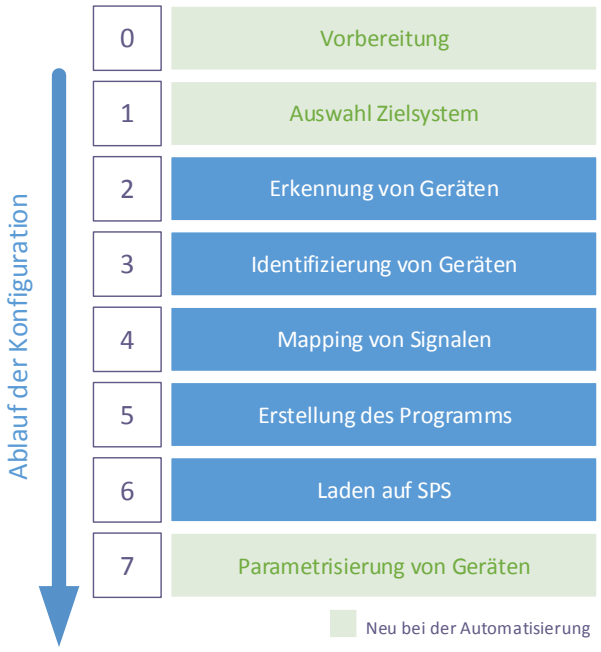


Abbildung 5.9: Ablauf der automatischen Konfiguration

GUI zurückgesetzt, um einen bekannten Zustand für spätere simulierte Benutzereingaben zu gewährleisten. Eine Überprüfung des Aufbaus der GUI findet allerdings nicht statt.

2. Broadcast Search: Nachdem die nötige Infrastruktur erstellt wurde, aktiviert Javelin über AI einen Broadcast-Search. Alle im Netzwerk angeschlossene TwinCAT Geräten werden angezeigt. Der Benutzer wählt seinen Zielrechner aus und trägt die Anmeldedaten ein.
3. Geräte-Scan: Erkennung der angeschlossenen Geräte
4. IO-Link Erkennung: Werden IO-Link Master gefunden, scannen diese nach IO-Link Sensoren, welche anhand einer Datenbank identifiziert und dafür geeignete Treiber bereitgestellt werden.
5. Mapping der Variablen: Sensor-Variablen werden mit Programm-Variablen verknüpft
6. Laden eines vorbereiteten SPS Programms
7. Konfiguration laden: Das TwinCAT Projekt wird auf die SPS übertragen und aktiviert

8. Parametrisierung: Das Einstellen der Sensorparameter erfolgt am Ende aufgrund der Verwendung von ADS

5.6 Ablauf

5.6.1 Initialisierung

Für den erfolgreichen Ablauf der Automatisierung ist eine Initialisierung notwendig, damit der Zustand des Systems den Erwartungen des Programms entspricht. Folgende Schritte werden unternommen:

- Die GUI wird zurückgesetzt. Dies ist notwendig für die korrekte Funktionalität der AutoIt Skripte, indem alle Standard-Bedienelemente zugänglich sind.

```
public void ResetUI()
{
  System.Diagnostics.Process.Start(@"AutoIt\resetUI.exe");
  _autoit.WinActivate(_dte_hwnd);
  _dte.ExecuteCommand("Window.ResetWindowLayout");

}
```

Quellcode 5.1: Funktion zum Zurücksetzen der GUI aus TcToolbox

  Zuerst wird resetUI.exe gestartet, welches den Bestätigungsdialog schließt. Per AutoIt das Visual Studio Fenster fokussiert und über Makro den GUI-Reset ausgelöst.

```
$hWnd = WinWait("[TITLE:Microsoft Visual Studio; CLASS:#32770]", "", 10)
ControlClick ( $hWnd, "", 6)
```

- Die SPS wird neu in Config Mode gestartet. Alle laufenden SPS Programme werden gestoppt und die automatische Geräteerkennung ermöglicht
- Zurücksetzen des IODD Repositories von TwinCAT, sodass alle Gerätetreiber neu geladen werden. Dazu werden alle Beschreibungsdateien gelöscht aus

  `C:\TwinCAT\3.1\Config\Io\IOLink`

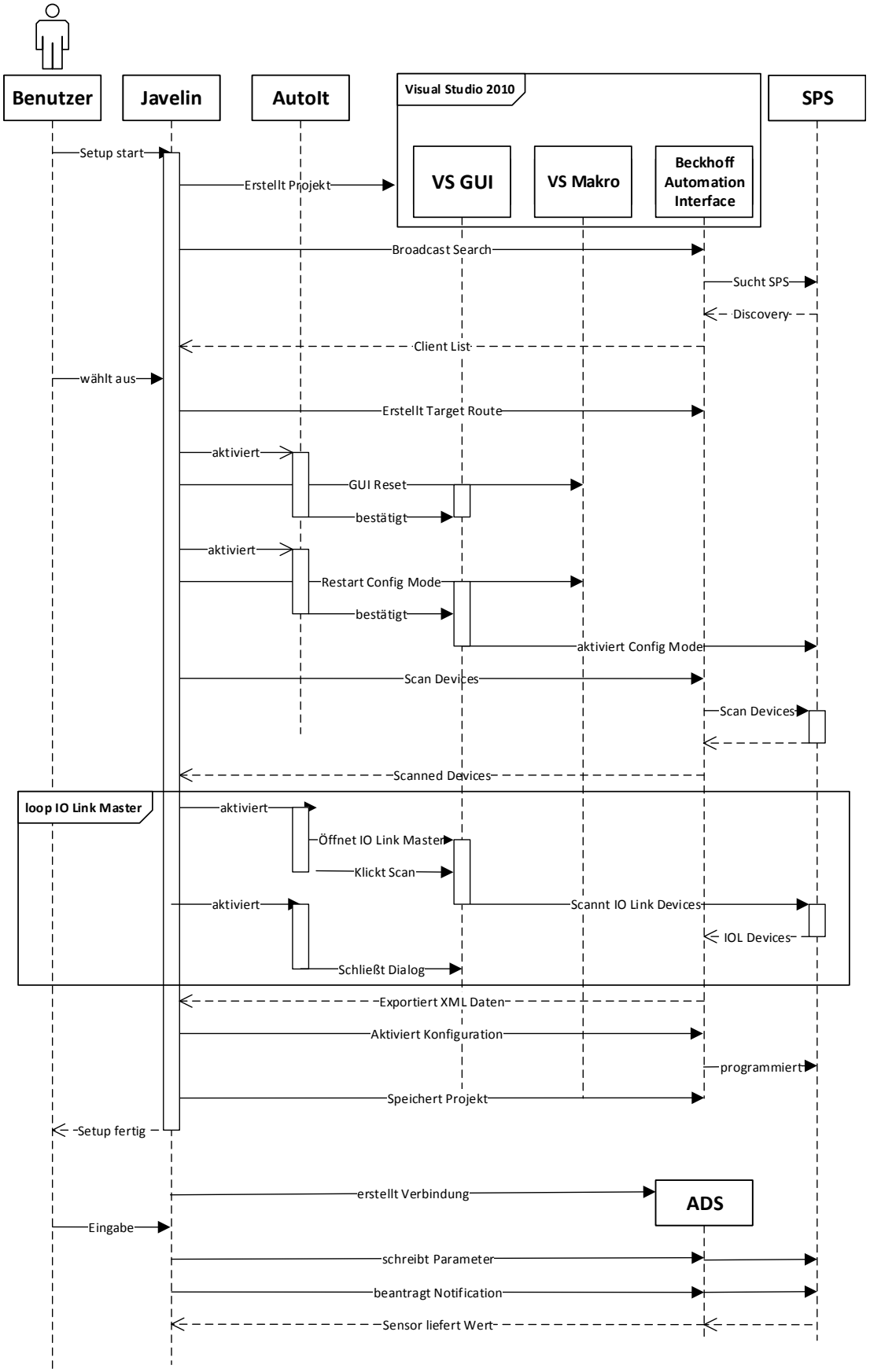


Abbildung 5.10: Detaillierter Ablauf der Automatisierung

5.6.2 Herstellung der Kommunikation

Zur Herstellung der Kommunikation wird über Automation Interface ein Broadcast-Search durchgeführt. Diese Aktion wird durch das Ändern eines Trigger-Elements

```
<BroadcastSearch>true</BroadcastSearch>
```

im Datenbaum unter *TIRR* (System/Routes) ausgelöst.

```
public IEnumerable<XElement> BroadcastSearch()
{
  string xmlString = "<TreeItem>" +
                "<RoutePrj>" +
                  "<TargetList>" +
                   "<BroadcastSearch>true</BroadcastSearch>" +
                  "</TargetList>" +
                "</RoutePrj>" +
              "</TreeItem>";

  ITcSmTreeItem routesNode = _sysManager.LookupTreeItem("TIRR");
  routesNode.ConsumeXml(xmlString);

  string result = routesNode.ProduceXml();
  XElement search_xml = XElement.Parse(result);

  IEnumerable<XElement> routes = from el in search_xml.Descendants("TargetList").Elements("
      Target") select el;
  return routes;
}
```

Quellcode 5.2: Durchführung des Broadcast-Searches

(Beckhoff Automation GmbH)

Anschließend werden die Ergebnisse als XML ausgegeben und in der GUI zur Auswahl präsentiert.

Nach der Eingabe der Benutzerkenndaten wird der Zielrechner als Target-System gespeichert und eine Route auf dem TwinCAT XAE erstellt.

5.6.3 Erkennung von Geräten

Die Erkennung von Geräten erfolgt bei TwinCAT im Grunde automatisch. Hierzu wird das Scannen indirekt ausgelöst über *ProduceXML* am Geräte-Knoten.

Im Gegensatz zum Scan via GUI werden die gefundenen Geräte nicht selbständig in den Strukturbaum eingetragen. Dies erfolgt nun durch das Einlesen von XML, welche die Information des Scans enthält.

Nachdem die Geräte gespeichert wurden, wird die Suche nach Boxen (weitere Hardware die an Geräten angeschlossen sind) ausgelöst, durch das Aktivieren der Trigger-Elemente bei den jeweiligen Geräten.

```
public void ScanDevices(bool scanBoxes)
{
  ITcSmTreeItem ioDevicesItem = _sysManager.LookupTreeItem("TIID"); //TIID is path for devices
  string scannedXml = ioDevicesItem.ProduceXml(false);

  XmlDocument xmlDoc = new XmlDocument();
  xmlDoc.LoadXml(scannedXml); // Loads the Xml data into an XML Document
  XmlNodeList xmlDeviceList = xmlDoc.SelectNodes("TreeItem/DeviceGrpDef/FoundDevices/Device");
  List<ITcSmTreeItem> devices = new List<ITcSmTreeItem>();

  int deviceCount = 0;

  // Add all found devices to configuration
  foreach (XmlNode node in xmlDeviceList)
  {
    // Add a selection or subrange of devices
    int itemSubType = int.Parse(node.SelectSingleNode("ItemSubType").InnerText);
    string typeName = node.SelectSingleNode("ItemSubTypeName").InnerText;
    XmlNode xmlAddress = node.SelectSingleNode("AddressInfo");
    ITcSmTreeItem device = ioDevicesItem.CreateChild(string.Format("Device_{0}", ++deviceCount)
        , itemSubType, string.Empty, null);

    if (xmlAddress != null)
    {
      string xml = string.Format("<TreeItem><DeviceDef>{0}</DeviceDef></TreeItem>", xmlAddress.
          OuterXml);
      device.ConsumeXml(xml); // Consume Xml Parameters (here the Address of the Device)
    }
    devices.Add(device);
  }

  if (scanBoxes)
  {
    // Scan all added devices for attached boxes
    foreach (ITcSmTreeItem device in devices)
    {
      string xml = "<TreeItem><DeviceDef><ScanBoxes>1</ScanBoxes></DeviceDef></TreeItem>";
      // Using the "ScanBoxes XML trigger-Method"
      try
      {
        device.ConsumeXml(xml);
        // Consume starts the ScanBoxes and inserts every found box/terminal into the
            configuration
      }
      catch (Exception ex)
```

```
43          {
44            Console.WriteLine("Warning: {0}", ex.Message);
45          }
46        }
47      }
48  }
```

Quellcode 5.3: Scannen nach Geräten und Boxen

(Beckhoff Automation GmbH)

5.6.4 Erkennung von IO-Link Sensoren

Die Funktion zur Erkennung von IO-Link Sensoren ist nur über die GUI zu erreichen. Dafür wird eine Kombination von Makros und AutoIT verwendet.

- Nachdem der Geräte-Scan beendet ist wird im Informationsbaum nach IO-Link Master anhand des Namens gesucht.
- Per Makro wird der Knoten des IO-Link Masters ausgewählt

```
1 public void GoToTreeItem(string path)
2 {
3
4   string macro_path = path.Replace(@"TIID", _context.SolutionName + @"\" + _context.
        ProjectName + @"\I/O\Devices").Replace(@"^", @"\");
5   _autoit.WinActivate(_dte_hwnd);
6   _dte.Windows.Item(EnvDTEConstants.vsWindowKindSolutionExplorer).Activate();
7   _dte.ActiveWindow.Object.GetItem(macro_path).Select(EnvDTE.vsUISelectionType.
        vsUISelectionTypeSelect);
8   _dte.ActiveWindow.Object.DoDefaultAction();
9 }
```

Quellcode 5.4: Navigieren im Visual Studio mithilfe des Pfades aus Automation Interface

Visual Studio wird als aktives Fenster ausgewählt und auf den *SolutionExplorer* (durch die ID *EnvDTEConstants.vsWindowKindSolutionExplorer*) fokussiert. Aus dem von Automation Interface gelierten Pfad

```
1     <PathName>TIID^Device 1 (EtherCAT)^Term 1 (EK1200)^Term 3 (EK1122)^Box 4
```

wird ein Pfad, das dem Strukturbaum in Visual Studio entspricht,

```
1     TwinCAT Project\TwinCAT Project\I/O\Devices\Device 1 (EtherCAT)\Term 1 (EK1200)\Term
          3 (EK1122)\Box 4
```

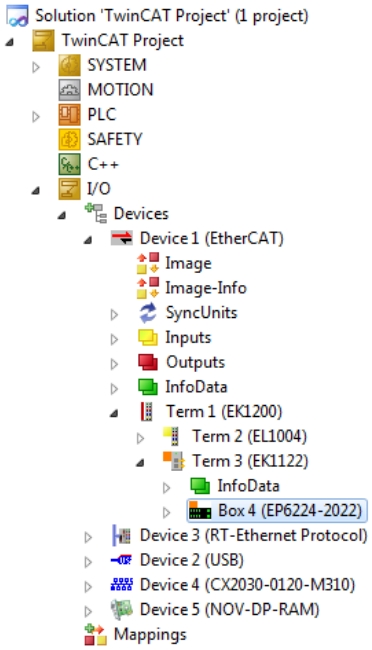


Abbildung 5.11: Pfad gibt die Position im Baum an

für das Makro nutzbar gemacht. Dabei ist zu beachten, dass Automation Interface alle Pfade auf English ausgibt. Das führt dazu, dass das Visual Studio ebenfalls auf englischer Sprache vorliegen muss.

- AutoIT wechselt zum IO-Link Tab, klickt auf den Button Scan Devices und quittiert den Benachrichtigungs-Dialog
- Erkennung der Geräte anhand von Vendor- und Device ID. Diese werden aufgrund der fehlenden IODDs angezeigt
- Kopieren der IODDs aus der Datenbank
- Erneutes Scannen der Geräte führt zur Idenfizierung dieser durch TwinCat, da benötigte IODDs vorhanden sind

5.6.5 Laden des Programms

Bei dieser Arbeit wird für das Programm lediglich eine Globale Variablenliste (GLV) mit Hilfe von Automation Interface erstellt, sodass die Eingangssignale verknüpft werden können. Eine Logik wird nicht implementiert.

```
/// <summary>
/// Creates Global Variable list
/// </summary>
/// <param name="name">Name of Global Variable list</param>
/// <param name="declarationCode">Content of the List</param>
/// <param name="project">Name of the PLC Project the list should be created in</param>
/// <returns></returns>
public ITcSmTreeItem CreateGLV(string name, string declarationCode, string project)
{
  ITcSmTreeItem glvsItem = _sysManager.LookupTreeItem("TIPC^" + project + "^" + project + "
      Project^GVLs");
  ITcSmTreeItem item = glvsItem.CreateChild(name, TreeItemType.PlcGlobalVariablesList.AsInt32()
      , "", declarationCode);

  var declaration = (ITcPlcDeclaration)item;
  declaration.DeclarationText = declarationCode;
  return item;

}
```

Quellcode 5.5: Funktion zur Erstellung von Globale Variablen Listen (GLV)

5.6.6 Signal Mapping

In diesem Schritt werden die Sensorvariablen mit den Variablen aus der GLV verknüpft. Dazu wird eine Funktion von Automation Interface unter Angaben von zwei Variablen-Pfade (Pfad des Signaleingangs und Pfad der Programm-Variable) aufgerufen.

```
  _sysManager.LinkVariables(Variable1, Variable2);
```

5.6.7 Starten der SPS

```
  _sysManager.ActivateConfiguration();
  _sysManager.StartRestartTwinCAT()
```

5.6.8 Parametrisierung von IO-Link Sensoren

Die Parametrisierung der IO-Link Sensoren erfolgt durch ADS nachdem die SPS gestartet ist, da das ADS System erst während der Laufzeit zur Verfügung steht. Dazu wird ein

ADS Stream an die NetId des IO-Link Masters geschrieben. Der Port entscheidet welches Gerät angesprochen wird.

Es wird zuerst ein ADS Objekt erstellt und eine Verbindung zum Gerät hergestellt. Hier ist *_AoENetId* die Adresse vom IO-Link Master. Der Sensor wird über den Port definiert: 4066 für Port 1 und 4067 für Port 2...

```
1 private TcAdsClient _adsClient = new TcAdsClient()
2 _adsClient.Connect(_AoENetId, 4095 + port_id); //0x0100
```

Quellcode 5.6: Erstellung eines TcAdsClient Objektes und die Verbindung zum Sensor angeschlossen an Port port_id

Um einen Parameter zu schreiben erstellt man hierfür einen *AdsStream* mit der Länge des Parameters, in welchen der Wert geschrieben wird. Anschließend wird die Nachricht per ADS *Write* Befehl verschickt.

```
1 AdsStream writeStream = new AdsStream(length);
2 BinaryWriter writer = new BinaryWriter(writeStream);
3 writer.Write(value);
4
5 _adsClient.Write(index_group, index_offset, writeStream, bit_offset, length);
```

Für den IO-Link Bedarfsdatenkanal ist der index_group auf 62210 festgelegt. Die restlichen vier Parametern sind abhängig vom Sensor und den zu ändernden Parametern.

5.7 Programmoberfläche

Die Programmoberfläche ist schlicht gehalten und enthält alle notwendige Informationen für die Parametrisierung von IO-Link Geräten.

Durch das Betätigen des *Start* Buttons wird die Automatisierung ausgeführt. Nachdem Visual Studio im Hintergrund gestartet worden ist, bekommt der Benutzer die Möglichkeit ein Zielsystem auszuwählen und die Kennungsdaten einzugeben. Darauf folgt die restliche Konfiguration und am Ende werden die gefundenen IO-Link Master und Sensoren präsentiert. Mit dem Auswählen des Sensors werden Bilder aus der IO-Link Gerätebeschreibungsdatei und die Parametern angezeigt.

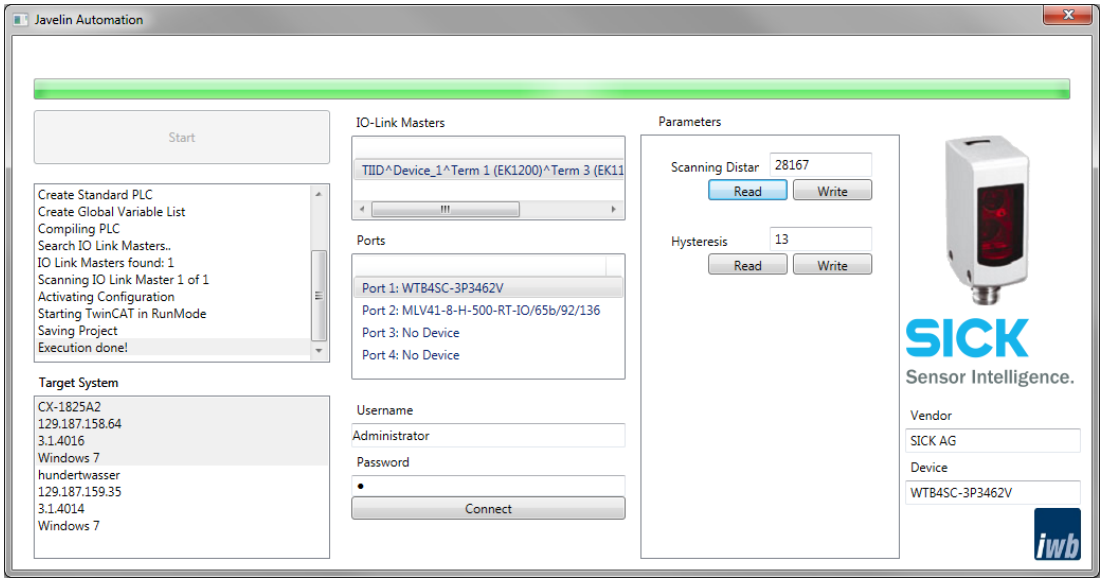


Abbildung 5.12: Screenshot von Javelin nach erfolgreichen Durchlauf

6 Bewertung

Die Konfigurationssoftware ist in der Lage die der Arbeit vorliegenden Geräte zu konfigurieren. Dazu werden neben Hersteller-Schnittstellen auch Visual Studio Makros und AutoIt verwendet, die Benutzereingaben nachbilden und somit stark systemabhängig sind. Da Macros verwendet werden, um innerhalb der Baumstruktur in TwinCAT zu navigieren, setzt die aktuelle Umsetzung die englische Version von Visual Studios voraus. Dies ist dadurch begründet, dass Automation Interface den Pfad auf englisch liefert.

Auch darf während dem Durchlauf der Automatisierung der Benutzer keine Tastatur- und Mauseingaben tätigen, da diese den Vorgang stören können. Zum jetzigen Zeitpunkt ist das allerdings die einzige Möglichkeit den Ablauf zu automatisieren, da entsprechende Schnittstellen nur über die GUI zu erreichen sind.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Durch sich ändernde Marktverhältnisse stehen die Hersteller vor großen Herausforderungen. Immer kleinere Serien und spezifischere Kundenwünsche verlangen flexible Produktionsanlagen, die sich an neue Produkte, Losgrößen und Qualitätsanforderungen anpassen können. Diesen Prozess nennt man Konfiguration. Moderne Produktionsanlagen sind sehr komplex. Grund dafür ist die große Vielfalt der Geräte und dem benötigten Fachwissen.

In dieser Arbeit wurde die automatische Erkennung und Konfiguration von Feldgeräten beispielhaft mit Beckhoff und IO-Link umgesetzt. Um zunächst das Verständnis für diese Systeme aufzubauen, erfolgt eine manuelle Konfiguration der Geräte. Die daraus ermittelten Handlungsschritten werden in Anbetracht der angebotenen Schnittstellen und Möglichkeiten umgesetzt. Die vom Hersteller bereitgestellten Kommunikationsplattform ADS und Funktionsbibliothek Automation Interface besitzen einen eingeschränkten Funktionsumfang. Daher werden diese durch Visual Studio Makros und AutoIt ergänzt, um die restlichen Arbeitsschritte per Simulation von Benutzereingaben durchführen zu können.

Nachdem die Geräte und Sensoren physisch miteinander verbunden sind, erfolgt die Erkennung der Geräte. Bei den erkannten IO-Link Mastern wird ebenfalls eine Geräteerkennung durchgeführt und passende Treiber für die angeschlossenen intelligenten Sensoren aus einer Datenbank geladen. Sobald die Sensorsignale mit Programmvariablen verknüpft sind, wird die SPS gestartet, sodass die Parametrisierung über ADS unternommen werden kann.

7.2 Ausblick

Die in dieser Arbeit umgesetzte Software stellt einen Rahmen für zukünftig weitere Erweiterungen in der automatischen Konfiguration von Feldgeräten. Sie könnte nach

der Analyse von Produkten anderen Hersteller in einen Framework umgeschrieben werden, welches Schnittstellen bereitstellt, und eventuell auch mit weiteren Treibern auf andre Geräte ausgeweitet werden könnte. Aktuell werden nur Systeme von Beckhoff betrachtet, sodass zu diesem Zeitpunkt eine weitere Abstraktion schwer ist, da die Definition der Programm Interfaces ohne Kenntnis der Funktionalität anderer Hersteller nicht möglich ist.

Im Laufe der Zeit ist zu erwarten, dass die Automatisierungsmöglichkeiten mit Herstellerschnittstellen deutlich zunehmen werden. Die Verwendung von Klick-Bots (Programme die Benutzereingaben simulieren) kann dadurch verringert und die Robustheit des Ablaufs erhöht werden. Beckhoff hat bereits angekündigt unter anderem das Starten der SPS in Config-Mode in das Automation Interface aufzunehmen.

Aufgrund der Abstraktion wirken sich Veränderungen auf Herstellerseite nicht auf das Benutzerprogramm aus, da Javelin Automation später solche Änderungen in einem Update berücksichtigt und anstatt Macros und AutoIt den direkten Funktionsaufruf von Automation Interface verwendet könnte.

Literaturverzeichnis

ANTZOULATOS et al. 2014
Antzoulatos, N.; E. Castro; D. Scrimieri; S. Ratchev: A multi-agent architecture for plug and produce on an industrial assembly platform. *Production Engineering* (2014). ISSN: 0944-6524. DOI: 10.1007/s11740-014-0571-x.

Beckhoff Automation GmbH
Beckhoff Automation GmbH, Hrsg.: *Online Dokumentation Wie kann ich einem Remote-ADS-Teilnehmer Routen hinzufügen? Wie kann ich einem Remote-ADS-Teilnehmer Routen hinzufügen?* URL: http://infosys.beckhoff.com/index.php?content=../content/1031/tc3%5Ctextunderscore%20automationinterface/27021598007162763.html%5C&id=16219.

Beckhoff Automation GmbH
Beckhoff Automation GmbH, Hrsg.: *Online Dokumentation Wie kann ich nach Geräten und Boxen suchen? Wie kann ich nach Geräten und Boxen suchen?* URL: http://infosys.beckhoff.com/index.php?content=../content/1031/tc3%5Ctextunderscore%20automationinterface/36028797261900683.html%5C&id=16217.

DURKOP et al. o.D.
Durkop, L.; H. Trsek; J. Otto; J. Jasperneite: A field level architecture for reconfigurable real-time automation systems. In: *2014 10th IEEE Workshop on Factory Communication Systems (WFCS)*. (Toulouse, France), S. 1–10. DOI: 10.1109/WFCS.2014.6837601.

HAMMERSTINGL & REINHART o.D.
Hammerstingl, V.; G. Reinhart: „Unified Plug&Produce Architecture for Automatic Integration of Field Devices in Industrial Environments“.

HEDELIND & JACKSON o.D.
Hedelind, M.; M. Jackson: The Need for Reconfigurable Robotic Systems. In: *Proceedings of CARV 2nd International Conference on Agile, Reconfigurable and Virtual Production*.

HODEK & SCHLICK o.D.
Hodek, S.; J. Schlick: Ad hoc field device integration using device profiles, concepts for automated configuration and web service technologies: Plug&Play field device integration concepts for industrial production processes. In: *2012 IEEE 9th International Multi-Conference on Systems, Signals and Devices (SSD)*. (Chemnitz, Germany), S. 1–6. DOI: 10.1109/SSD.2012.6197997.

LAUBER & GÖHNER 1999
Lauber, R.; P. Göhner: *Automatisierungssysteme und -strukturen, Computer- und Bussysteme für die Anlagen- und Produktautomatisierung, Echtzeitprogrammierung und Echtzeitbetriebssysteme, Zuverlässigkeits- und Sicherheitstechnik*. 3., völlig neubearb. Aufl. Bd. / Rudolf Lauber; Peter Göhner ; 1. Prozessautomatisierung. Berlin [u.a.]: Springer. 1999. 440 S. ISBN: 978-3-540-65318-9.

REINHART et al. o.D.
Reinhart, G.; S. Krug; S. Huttner; Z. Mari; F. Riedelbauch; M. Schlogel: Automatic configuration (Plug & Produce) of Industrial Ethernet networks. In: *2010 9th IEEE/IAS International Conference on Industry Applications - INDUSCON 2010*. (Sao Paulo, Brazil), S. 1–6. DOI: 10.1109/INDUSCON.2010.5739892.

ZIMMERMANN et al. 2008
Zimmermann, U.; R. Bischoff; G. Grunwald; G. Plank; D. Reintsema: COMMUNICATION, CONFIGURATION, APPLICATION: The three layer concept for Plug-and-Produce. In: *ICINCO 2008. Proceedings*. [Setúbal]: Insticc Press. 2008. ISBN: 978-989-8111-35-7.

Eidesstattliche Erklärung

Ich erkläre hiermit eidesstattlich, dass ich die vorliegende Arbeit selbstständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt.

Garching, den 27.11.2014

(Jason Li)