Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Carina Wüst

Spezifikation und Realisierung eines Standard-Interfaces zur Anlagenvisualisierung und Steuerung

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Carina Wüst

Spezifikation und Realisierung eines Standard-Interfaces zur Anlagenvisualisierung und Steuerung

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Andreas Suhl
Zweitgutachter : Prof. Dr.-Ing. Michael Röther

Abgegeben am 21. Juni 2012

Carina Wüst

Thema der Bachelorthesis

Spezifikation und Realisierung eines Standard-Interfaces zur Anlagenvisualisierung und Steuerung

Stichworte

Schnittstellen, Kommunikationsprotokolle, Speicherprogrammierbare Steuerungen, Steuerungssoftware

Kurzzusammenfassung

Diese Arbeit beschreibt die Spezifikation und Realisierung eines Interfaces im Zuge einer Standardisierung. Es werden verschiedene Schnittstellen sowohl theoretisch als auch praktisch auf ihre Eigenschaften untersucht und überprüft.

Carina Wüst

Title of the paper

Specification and implementation of a standard interface for the visualisation of equipment and control

Keywords

Interfaces, communication protocol, programmable logic controller, controlsoftware

Abstract

This paper describes the specification and realisation of an interface in the course of standardisation. Different interfaces are analyzed and tested in their characteristics in theory and practise.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich bei der Anfertigung dieser Bachelorthesis unterstützt haben.

Ein besonderer Dank gilt meinen beiden Betreuern. Herrn Prof. Dr.-Ing. Andreas Suhl, der mich mit seiner netten und flexiblen Betreuung und seinen beruhigenden Worten immer unterstützt hat, sowie Herrn Prof. Dr.-Ing. Michael Röther für die Zweitkorrektur.

Weiterhin bedanke ich mich bei der YXLON International GmbH, dass sie mir das Thema zur Verfügung gestellt und die Möglichkeit gegeben haben, diese Arbeit im Unternehmen anzufertigen. Besonders hervorheben möchte ich Jan Spalding sowie Rainer Steinfeld für ihre Unterstützung und ihre stete Bereitschaft mir mit fachlichem Rat zur Seite zu stehen.

Selbstverständlich möchte ich mich auch bei meinen Freunden und meiner Familie bedanken, die mich in allen Phasen der Arbeit und des Studiums unterstützt haben; besonders aber bei Christoph Schlüter und Torben Ibs, ohne die ich dieses Studium niemals begonnen hätte.

Inhalt

1 Einleitung

Die Verwendung automatisierter Anlagen in der Produktion nimmt stetig zu. Durch höhere Anforderungen an diese Anlagen, steigt deren Komplexität und Leistungsfähigkeit. Deshalb müssen die einzelnen Komponenten dieser Anlagen perfekt aufeinander abgestimmt sein, um die Produktion so effektiv wie möglich zu gestalten.
Fehlende Schnittstellen und standardisierte Protokolle bilden Kommunikationsbarrieren zwischen den Automatisierungskomponenten und anderen Bereichen. Selbst zwischen unterschiedlichen Automatisierungskomponenten können die Daten teilweise nicht effektiv ausgetauscht werden. Mit Hilfe von Standardschnittstellen, sowohl in der Hardwaretechnik als auch in der Software, und vereinheitlichten Kommunikationsprotokollen werden unterschiedliche Systeme miteinander verbunden und sind so der Schlüssel für eine erfolgreiche Datenübertragung.
In der Automatisierungstechnik muss eine Vernetzung von Geräten, wie z.B. Steuerungen, PCs oder Robotern erfolgen. Das jeweilige Übertragungsmedium muss für die Bedingungen in der Industrie geschaffen sein, so muss es beispielsweise rauen Umgebungen standhalten, gute Möglichkeiten zur Fehlerdiagnose und schnelle Wartbarkeit bieten, zuverlässig arbeiten und eine, auf den Einsatzbereich zugeschnittene, Übertragungsgeschwindigkeit gewährleisten.
Der Austausch von Daten zwischen diesen Komponenten wird durch sogenannte Transportprotokolle realisiert. Hier wird die Kommunikation der Anlagenkomponenten definiert.

1.1 Aufgabe

Diese Bachelorthesis wurde im Rahmen der Standardisierung der YXLON International GmbH angefertigt.
In den Röntgen- und CT-Anlagen der YXLON International GmbH werden zum aktuellen Zeitpunkt verschiedene Schnittstellen zwischen PC und Steuerung verwendet. Die Anforderungen an die Schnittstellen sind anlagenübergreifend aber doch die gleichen. Um die Produktion noch wirtschaftlicher zu gestalten, ist es notwendig, die Anlagenfamilien weitestgehend zu standardisieren. Im Zuge der Entwicklung einer neuen Anlage, die unter dem Projektnamen „Asterix“ läuft, soll eine Analyse aller bisher verwendeten Schnittstellen vorgenommen werden.
Das Entwicklungsprojekt Asterix ist die Entwicklung eines Computertomographen, der zur zerstörungsfreien Materialprüfung verwendet wird.

Im Rahmen dieses Entwicklungsprojektes entsteht eine neue Automatisierungslösung zur Steuerung des Prüfteilmanipulators. Die Spezifikation dieser Aufgabe umfasst zusätzlich dem Pflichtenheft des Projektes Asterix auch die Anforderungen aller Anlagenfamilien bei YXLON. Mit dieser Manipulatorsteuerung soll eine schlanke, erweiterbare und universelle Automatisierungslösung entstehen, die das Fundament zukünftiger Anlagenneuentwicklungen darstellt.
Es soll also eine Schnittstelle gefunden werden, die sowohl für das Projekt „Asterix“ als auch als Standard-Interface einsetzbar ist.

2 Theoretische Grundlagen

Kommunikation besteht aus mindestens zwei oder mehreren Teilnehmern. Wird der einfachste Fall betrachtet, also zwei Teilnehmer, muss einer von beiden der Sender und der andere der Empfänger sein, wobei beide Parteien während der Unterhaltung die Rollen tauschen können. [4b]
Wird diese allgemeine Definition einer Kommunikation nun auf die Netzwerktechnik übertragen, müssen Kommunikationsregeln definiert werden, damit Informationen so zuverlässig wie möglich übertragen werden können. Dazu werden Protokolle definiert.
Ein Protokoll gibt an, wie zwei Parteien miteinander kommunizieren. Diese Parteien werden Master und Slave genannt, oder auch einfach Sender und Empfänger. Das Protokoll legt fest, wer die Kommunikation beginnt, wer antwortet und wann die Kommunikation beendet ist. Es kann also gesagt werden, dass ein Protokoll die Syntax, Semantik und Synchronisation einer Kommunikation definiert. [12]
In der Netzwerktechnik finden verschiedene Protokolle Anwendung, doch ihre allgemeine Funktionsweise, nämlich die Signalübertragung zwischen zwei Punkten, ist die gleiche. In einem digitalen Übertragungssystem repräsentieren die Signale binäre Symbole oder Bits.

2.1 OSI-Referenzmodell

Um die Aufgaben und Anwendungen dieser Protokolle besser spezifizieren und anwenden zu können, wurde das sogenannte OSI-Modell (Open Systems Interconnection), welches das ISO (International Standardization Organisation) Referenzmodell ist, definiert. Das Modell besteht aus sieben Schichten, auch Ebenen oder Layer genannt, wobei jede Schicht einen Dienst in der Kommunikation symbolisiert. Der gesamte Kommunikationsweg wird allgemeingültig und hardwareunabhängig dargestellt. Der Datenstrom läuft in senkrechter Richtung durch alle Schichten, dies wird vertikale Kommunikation genannt. Horizontale

Kommunikation ist die Kommunikation, die auf einer Ebene zwischen zwei gleichnamigen Schichten der kommunizierenden Systeme stattfindet.

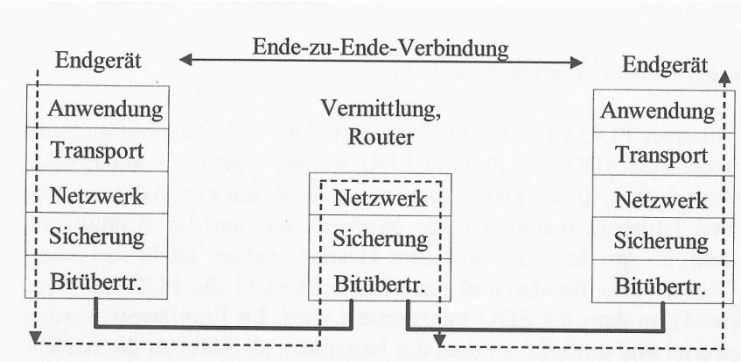

Abbildung 1 Informationsfluss in einem Kommunikationsnetz im OSI-Modell Quelle: [3]

1. Schicht: Bitübertragung (Physical Layer)

In der physikalischen Schicht wird das Senden und Empfangen der „rohen" Datenbits geregelt. Die Datenbits werden in elektrische, elektromagnetische, akustische oder optische Signale umgewandelt und über das Übertragungsmedium gesendet und empfangen. Es wird definiert, welche Spannungen welchem logischen Pegel entsprechen, außerdem werden Periodendauern, Taktzyklen, Impulsbreiten sowie mechanische Charakteristika wie Pinbelegungen, Kabeltypen etc. umgesetzt.

2. Schicht: Sicherung (Data Link Layer)

Diese Schicht verbindet zwei Stationen innerhalb eines Netzsegmentes miteinander. Sie legt den Zugang zum physikalischen Netz und kontrolliert die Bitübertragung (Quittierungsfunktionen, Überlastkontrolle). Die Verbindungsschicht fügt beim Senden der Daten entsprechende Prüfsummen und/oder Korrekturbits (CRC - cyclic redundancy check, Hamming-Bits) hinzu. Wenn beim Empfangen Übertragungsfehler auftreten, die nicht korrigiert werden können, wird ein erneutes Senden der Daten veranlasst.

Die Verbindungsschicht ist in zwei Unterschichten unterteilt, die MAC (Media Access Control)-Schicht regelt den Zugriff auf das Übertragungsmedium und die LCC (Logical Link Control)-Schicht stellt Funktionen bereit, die vom Übertragungsmedium abhängig sind. [1]

3. Schicht: Vermittlung (Network Layer)

Die Vermittlungsschicht, auch Netzwerkschicht genannt, bestimmt das Routing, also den Weg der Datenpakete, und regelt die Flusskontrolle, d.h. die Kommunikation zwischen langsamen und schnellen PCs, Überlastungsschutz und Fehlerbehebung (Paketdublikate, Irrläufer, o.ä.) zwischen den Endpunkten einer Verbindung. Im Wide Area Network (WAN) findet die Protokollumsetzung (Internetworking) statt. Hierzu lassen sich drei Teilschichten bilden:

Subnetwork Access, Abwicklung der Protokolle der jeweiligen Teilnetze.
Subnet Enhancement, Ergänzungen um Funktionen der Teilnetze, sodass die Anforderungen zur Protokollumsetzung erfüllt werden.
Internetworking, Abwickeln teilnetzunabhängiger Protokolle (Routing, globale Adressierung). [1]

4. Schicht: Transport (Transport Layer)
Diese Schicht ist sozusagen die jeweilige Endstelle einer Ende-Ende-Verbindung. Pakete, die aus der Zerlegung eines Datenstroms entstehen, werden an die Netzwerkschicht übergeben bzw. von dort empfangene Pakete werden zur Weitergabe an die darüber liegende Schicht wieder zu einem Datenstrom zusammengesetzt. Außerdem ist die Transportschicht für den Auf- und Abbau und für die Überwachung der Verbindung verantwortlich. Sie stellt fünf Dienstklassen zur Verfügung:
Klasse 0, gegenüber Schicht 3 keine Fehlerkontrolle. Eine Transportverbindung steht genau einer Netzwerkverbindung gleich.
Klasse 1, wie Klasse 0, allerdings mit Behebung von Fehlern aus Schicht 3, nach einer Unterbrechung der Verbindung wird z.B. versucht diese wieder herzustellen, sodass die Unterbrechung in höheren Schichten nicht bemerkt wird.
Klasse 2, Multiplexverbindung, also der Aufbau mehrerer Verbindungen. Die Netzwerkverbindung darf erst dann getrennt werden, wenn alle Verbindungen abgebaut sind.
Klasse 3, Zusammenfassung aus den Klassen 1 und 2.
Klasse 4, wie Klasse 3, allerdings ist diese um zusätzliche Funktionen zur Fehlererkennung und Fehlerbehandlung erweitert. [1] [7b]

5. Schicht: Sitzung (Session Layer)
In der Sitzungsschicht wird der Aufbau einer Übertragungssitzung zu einem entfernten System definiert. Dienste zur Kommunikationssteuerung bezüglich des Aufbaus, der Durchführung und der Beendigung der Verbindung werden bereitgestellt. Auch Spezifikationen von Sicherheitstechniken z.B. Passwörter oder die Überwachung der Betriebsparameter und Datenflusssteuerung gehören zu dieser Schicht. Unter Datenflusssteuerung versteht man z.B. die Bereitstellung von Übertragungstoken und Synchronisationspunkten, sodass im Fall einer Unterbrechung der Verbindung die Übertragung nicht komplett neu definiert werden muss. [1] [7]

6. Schicht: Darstellung (Presentation Layer)
In dieser Schicht wird die Darstellung von Daten durch Protokolle definiert. Die Protokolle dieser Schicht übersetzen die Datendarstellung eines Rechnertyps in die eines anderen, da verschiedene Rechnertypen Ganzzahlen und Zeichen intern anders darstellen. [7]

7. Schicht: Anwendung (Application Layer)
Die Anwendungsschicht unterscheidet sich von den anderen Schichten, da sie die oberste Schicht des Modells ist und ihre sogenannten Benutzer nicht Instanzen der nächsthöheren Schicht sind, sondern Prozesse, z.B. Softwarepakete oder Programme, die ihre Dienste bereitstellen. Den Benutzern oder Benutzerprogrammen werden also systemunabhängige Anwendungsdienste zur Verfügung gestellt. Die Anwendungsschicht selbst verwendet die Dienste der Darstellungsschicht. [6]
Den Transport- und Sicherungsschichten wird in dieser Arbeit im Zuge der Aufgabenstellung die größte Aufmerksamkeit gewidmet.

2.2 Transmission Control Protocol

Das Transmission Control Protocol (TCP) kann man als eine Realisierung der Schicht 4, also der Transportschicht, des OSI-Modells interpretieren. TCP nutzt den vom IP (Internet Protocol) bereitgestellten unzuverlässigen Datagrammdienst beim Austausch von Daten zwischen zwei Computern, bietet aber hierbei den Anwendungsprogrammen einen zuverlässigen Datendienst. In der Verantwortung des Transportprotokolls liegt die korrekte und vollständige Übertragung der Daten. Beim Datenaustausch mit einem Transportdienst interagieren die Anwendungen. TCP ist ein Ende-zu-Ende-Protokoll, da es eine direkte Verbindung zwischen einer Anwendung eines lokalen Computers zu einer Anwendung eines entfernten Computers bietet. Von den Anwendungen kann die Forderung an TCP gestellt werden, eine Verbindung aufzubauen, Daten auszutauschen und die Verbindung wieder zu trennen. Diese Verbindungen sind allerdings nur virtuell zu verstehen, da sie von der Software realisiert werden und nicht von der Hardware. Die TCP-Softwaremodule tauschen auf beiden Computern Nachrichten aus, um die Illusion einer Verbindung zu schaffen. [1]
Zur Vermittlung von Nachrichten wird IP genutzt. Die TCP-Nachrichten werden jeweils in einem IP-Datagramm zusammengepackt und im Netzwerk übertragen. Das Internet Protokoll gibt den Inhalt des Datagramms ab, sobald das Datagramm am Ziel-Host ankommt. Die Nachrichten werden von IP aber weder gelesen noch interpretiert. Wie schon erwähnt, ist TCP für die zuverlässige Übertragung der Nachrichten verantwortlich. Jede Nachricht kann,

z.B. durch Abbruch der Verbindung, verloren gehen, sich verspäten, dupliziert werden oder in der falschen Reihenfolge gesendet werden. Nachrichten müssen also eindeutig sein, denn sonst würde das Protokoll Nachrichtenduplikate akzeptieren. Dies wird von TCP durch verschiedene Mechanismen gewährleistet. Durch die Funktion der Neuübertragung (Retransmission) gleicht der Sender einen möglichen Paketverlust aus. Daran nehmen beide Seiten der Kommunikation teil. Bevor Daten versendet werden, startet TCP einen Timer. Ist dieser abgelaufen, bevor die Daten empfangen wurden, werden diese noch einmal versendet. Hat der Empfänger die Daten jedoch erhalten, schickt er dem Sender eine Bestätigung (Acknowledgement, ACK). Diese Bestätigung wird innerhalb von wenigen Millisekunden erwartet. Bevor das Transmission Control Protocol entwickelt wurde, basierte diese Wartezeit bis zu einem erneuten Sendeversuch auf einem festen Wert. Es wurde aber erkannt, dass sich eine feste Zeiteinheit nicht gut zur Übertragung eignet, deshalb wird nun die sogenannte adaptive Neuübertragung verwendet. Dies bedeutet, dass jede, auf einer Verbindung anliegende, Verzögerung überwacht wird und den Timer für eine Neuübertragung dementsprechend anpasst bzw. ändert. Allerdings können diese Verzögerungen nicht zu jeder Zeit von TCP überwacht werden. Die Umlauf- bzw. Roundtrip-Zeiten (RTT) der jeweiligen aktiven Verbindung werden abgeschätzt, indem die Zeit bis zum Erhalt einer Antwort gemessen wird. Diese ist dann die maximale Zeit, die das Protokoll für die Übertragung einer Nachricht benötigt bzw. benötigen darf. Da diese Werte immer wieder von TCP ermittelt und Mittelwerte gebildet werden, kann TCP diese bei Bedarf z.B. im Fall einer hohen Netzauslastung, anpassen.

Der Datenfluss wird mit der Funktion des Fenstermechanismus kontrolliert. Bei jedem Verbindungsaufbau wird dem Ende der Verbindung ein Puffer für die Aufnahme der empfangenen Daten zugeteilt. Die Größe dieses Puffers wird dem anderen Ende mitgeteilt. Wenn der Empfänger die Daten erhalten hat, sendet er eine Bestätigung mit Angaben zur verbleibenden Puffergröße. Der verfügbare Speicherplatz im Puffer wird Fenster (Window) genannt. Die Änderung der Größe dieses Speicherplatzes trägt den Namen Fensteranzeige (Window-Advertisement). Der Empfänger sendet mit jeder Bestätigung eine Fensteranzeige.

Wenn das Zeitverhältnis stimmt und die empfangenen Daten so schnell gelesen werden wie sie ankommen, sendet der Empfänger mit jeder Bestätigung eine positive Fensteranzeige. Ist dies aber nicht so, also wenn schneller gesendet als empfangen wird, füllt sich der Puffer beim Empfänger, was dazu führt, dass er ein Nullfenster (Zero-Window) anzeigt. Wenn so ein Nullfenster gesendet wurde, muss der Sender so lange pausieren, bis der Empfänger wieder eine positive Anzeige sendet.

Damit garantiert ist, dass Verbindungen zwischen Sender und Empfänger zuverlässig aufgebaut und beendet werden, nutzt das Transmission Control Protocol das Drei-Wege-

Handshake (3-way-handshake). Die Entwickler von TCP fanden heraus, dass ein dreifacher Austausch notwendig und dennoch ausreichend ist, um trotz Duplikaten, Verzögerungen und Paketverlusten eine eindeutige Vereinbarung sicherzustellen. Dafür wird auch der Begriff Synchronisationssegment (SYN-Segment) verwendet. Mit dem Ausdruck End-Segment (FIN-Segment) wird das Ende einer Verbindung im Drei-Wege-Handshake bezeichnet. Bei der Beendung einer Verbindung werden beispielsweise in beide Richtungen Bestätigungen, also ACKs gesendet, um zu gewährleisten, dass alle Daten vor Abbau der Verbindung empfangen wurden. Wie auch bei anderen Nachrichten wird die Übertragung verlorener SYN- oder FIN-Segmente von TCP wiederholt. Ein Teil des Handshakeverfahrens wird dazu genutzt eine Verbindung zu erstellen, bei der jedes Ende eine zufällig gewählte 32 Bit lange Sequenznummer erzeugen muss. Wenn eine Anwendung versucht, eine neue TCP-Verbindung aufzubauen nachdem ein Computer gestartet wurde, wählt TCP eine neue Zufallsnummer. Zwei Anwendungsprogramme können also über TCP kommunizieren, eine Verbindung beenden und später neu aufnehmen, ohne dass Duplikate oder verzögerte Pakete nachteilig darauf einwirken, da jede neue Verbindung eine neue Sequenznummer erhält. [6] [7]

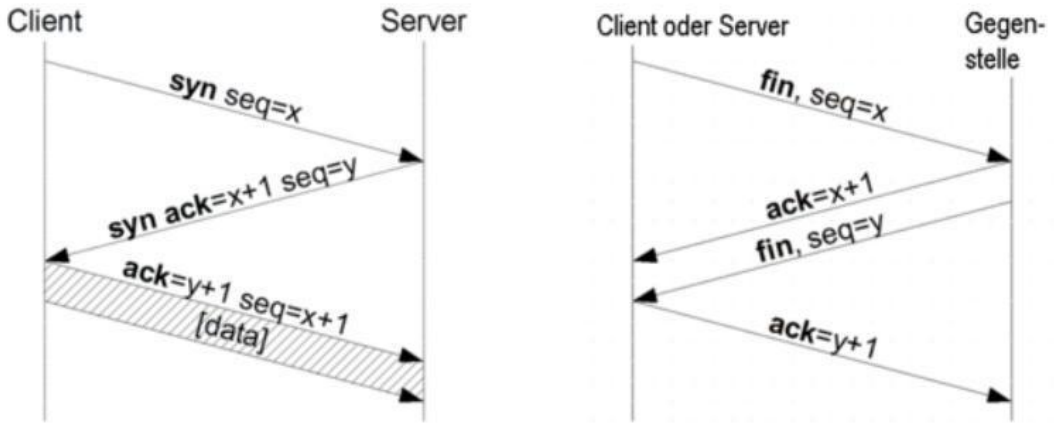

Abbildung 2 3-Wege-Handshake bei Verbindungsauf- und Abbau Quelle: [13]

Ein weiteres Merkmal, welches das Transmission Control Protocol auszeichnet, ist der Mechanismus der Überlastkontrolle (Congestion Control). Oft entstehen Paketverluste oder extrem lange Verzögerungen eher durch Überlast als durch einen Hardwarefehler. Transportprotokolle weiten dieses Problem sogar noch aus, da sie zusätzliche Kopien einer Nachricht einschleusen. Das gesamte System kann einen Kollaps erleiden, wenn durch Überlast ein übermäßig hohes Volumen ausgelöst wird. Dies wird verhindert, indem TCP den Paketverlust immer als Messeinheit für Überlast zugrunde legt und durch das Absenken der Rate, in der es Daten erneut überträgt, reagiert. Da TCP im Gegensatz zu anderen Transportprotokollen keine Neuübertragungen durchführt, wird der Prozess nicht weiter verschärft.

Für alle Nachrichten, auch Bestätigungen und Nachrichten im Rahmen des Drei-Wege-Handshakes, verwendet TCP das gleiche Format. Eine Nachricht wird Segment genannt.

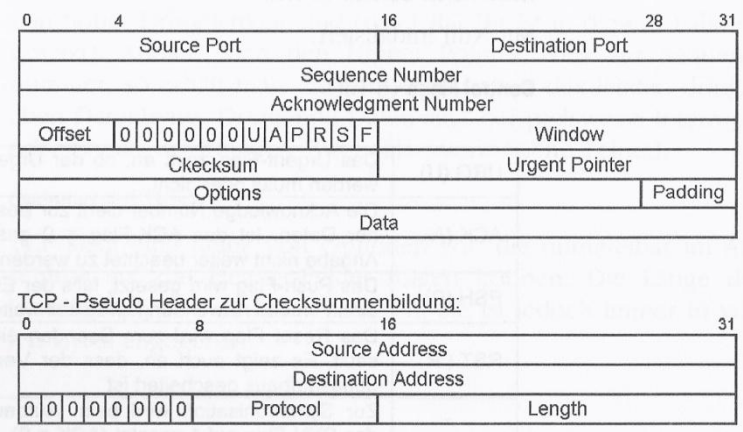

Abbildung 3 TCP Segmentformat Quelle: [1]

Das Segmentformat beinhaltet beide Datenströme, die TCP umfasst. Wenn die Anwendungen an beiden Enden gleichzeitig Daten senden, kann TCP ein einzelnes Segment senden, das die Bestätigung (ACK) für Eingangsdaten, Ausgangsdaten und eine Fensteranzeige mit der verfügbaren Puffergröße beinhaltet. Daher beziehen sich einige Felder des Segments auf die Datenströme, die in Vorwärts- und Rückwärtsrichtung fließen.
Wenn der Rechner ein Segment sendet, beziehen sich die Felder acknowledgment number und window auf die Eingangsdaten. Dabei definiert acknowledgment number die Sequenznummer der zu empfangenden Daten und window die für weitere Daten noch verfügbare Puffergröße. Das Feld sequence number bezieht sich auf die Ausgangsdaten und gibt die Sequenznummer der im Segment enthaltenen Daten an. Der Empfänger ordnet die Segmente dann anhand ihrer Nummern und nutzt sie zur Berechnung von Bestätigungsnummern. Welches Anwendungsprogramm auf dem empfangenen PC die Daten erhalten soll, wird im Feld destination port definiert. Das Feld source port bezeichnet das sendende Anwendungsprogramm. Eine Prüfsumme für den Header und die Nutzdaten des TCP-Segments sind im Feld checksum enthalten. [2] [3] [7]

2.3 User Datagram Protocol

Wie das TCP-Protokoll auch, ist das User Datagram Protocol (UDP) ein Transportprotokoll, das im OSI Referenzmodell der Transportschicht zugeordnet ist. Es ist als Erweiterungsmodul des Betriebssystems (z.B. WINSOCK.dll) implementiert und definiert einen verbindungslosen Transportmechanismus zum Versenden von Datagrammen innerhalb eines Netzwerkes. UDP verpackt die zu sendenden Daten als IP-Pakete. Damit gibt das Protokoll keine Garantie in Bezug auf eine gesicherte Zustellung und Fälschungen, arbeitet dafür aber unkompliziert und schnell. Außerdem ist UDP nicht streamorientiert, das heißt, dass Pakete am Stück verschickt werden und nicht erst darauf gewartet wird, ob Teile des Paketes angekommen sind, bevor mit dem weiteren Senden begonnen wird. Bei Verwendung von UDP ist also nicht gewährleistet, dass die Daten überhaupt oder in der richtigen Reihenfolge ankommen, da keine Antwort auf das Senden erfolgt. [3] [7]

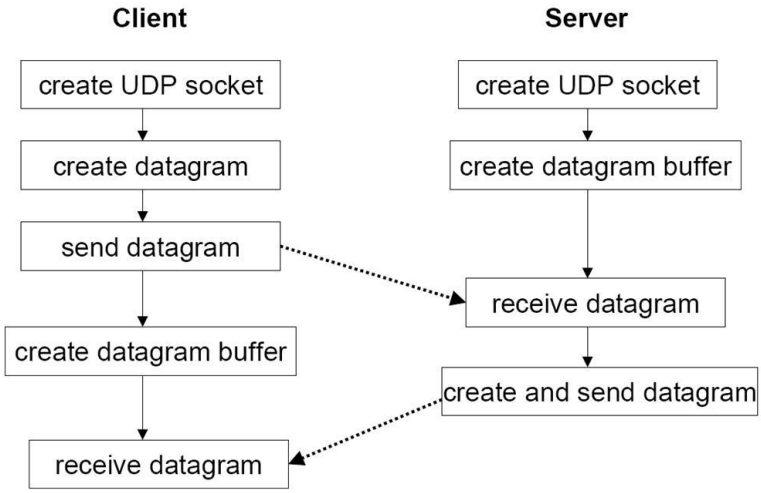

Abbildung 4 Ablauf einer UDP-Kommunikation Quelle [14]

Flusskontrolle, Verbindungsmanagement, Fehlerbehandlung und Zeitüberwachung sind nicht enthalten. Auf einem Computer laufen meist gleichzeitig mehrere verschiedene Prozesse, Dienste und Anwendungen wie z.B. ein Internet-Browser, ein Mail-Programm, diverse Anwendungsprogramme usw. Über die IP-Adresse leistet IP nur die Verbindung zwischen den Rechnern, aber nicht die Zuordnung zu den jeweiligen darauf laufenden Diensten, wofür die Daten aber bestimmt sind.

Aus diesem Grund wurden mit dem User Datagramm Protocol sogenannte Portnummern eingeführt. Jede Anwendung und jeder Dienst, die UDP-Daten senden und empfangen können, bekommen eine eigene eindeutige Portnummer zugewiesen. Manche dieser Portnummern sind reserviert, sodass bei der Wahl der Nummern Vorsicht geboten bist. Die Portnummern von 0 bis 1023 (Well known port numbers) sind für allgemein gültige Zwecke fest reserviert. Dieser Bereich wird von der IANA (Internet Assigned Numbers Authority)

gepflegt und verwaltet. Portnummern aus diesem Bereich für eigene Zwecke zu verwenden, ist somit zu vermeiden, da dies zu Problemen und Kollisionen führt. Für den Eigengebrauch stehen die Portnummern 1024 bis 65535 zur Verfügung. [1]

Auf dem lokal verwendeten Rechner existiert eine ASCII-Datei namens Services, in der die Zuordnung aller auf dem Rechner verwendeten Portnummern zu den jeweiligen Protokollen und Diensten hinterlegt ist. Mit einem gängigen Texteditor kann die Datei editiert werden. Bei Verwendung einer neuen Portnummer ist es also sinnvoll, in dieser Datei zu verifizieren, dass dieser Port noch nicht belegt ist. Auch im Multiplexbetrieb lassen sich aufgrund der IP-Adresse, der IP-Protokollnummer und der Portnummer Pakete bzw. Datagramme eindeutig zustellen. Über eine Leitung gelangen Pakete verschiedener Protokolle an die unterschiedlichen Ports, wobei ein Port gleichzeitig sowohl einem UDP- als auch einem TCP-Dienst zugewiesen sein kann, ohne dass es dadurch zu Kollisionen kommt. Dies funktioniert mit den sogenannten Listenern. Listener sind aktive „Lausch“-Programme, die permanent im Hintergrund laufen und gezielt den am Rechner vorbeiströmenden Datenverkehr nach Datenpaketen, die für die ihnen bekannten Ports bestimmt sind, abhören. Wenn ein relevantes Paket erkannt wurde, wird es herausgefiltert und an die über den Port angebundene Applikationen, Dienste oder Prozesse weitergeleitet. Prinzipiell ist UDP somit nicht viel mehr als ein erweitertes IP-Protokoll. Die IP-Protokollnummern werden durch die Verwendung von Portnummern entlastet. Die Nummern schaffen mehr Raum, sodass nicht für jeden neuen Dienst eine neue Protokollnummer auf IP-Ebene vergeben werden muss.

Eine gesicherte Verbindung ist nicht für alle Dienste oder Anwendungen oberhalb der Netzwerkschicht von Nöten. Das User Datagramm Protocol wird meist bei Diensten wie dem Internet Name Server, dem Trivial File Transfer, RIP sowie zur Telefonie, Videokonferenzen und Radioübertragungen über das Internet eingesetzt. Also dort, wo ein einzelnes Paket nicht über die Aussagefähigkeit der gesamten Information entscheidet und die Geschwindigkeit Vorrang hat.

Der Header von UDP hat eine konstante Größe von acht Byte. Unmittelbar im Anschluss daran folgen die Nutzdaten des Datagramms.

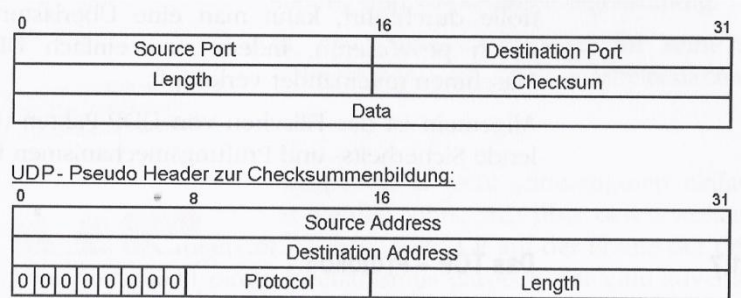

Abbildung 5 UDP Header Quelle: [1]

Die Source-Port Angabe ist optional und bezeichnet den Port des absendenden Prozesses und ist hilfreich, wenn Antworten an diesen Port zurückgesendet werden sollen. In jedem anderen Fall wird eine Null eingetragen. Der Destination Port beschreibt den Zielport, an den das Datagramm gerichtet ist. Im Feld Length wird die Gesamtlänge des Datagramms (Header und Byte) in Byte angegeben. Ohne Daten ist der kleinste zulässige Wert die Acht. Das Datagramm kann höchstens 1400 Byte groß sein. Die Berechnung der Prüfsumme ist optional. Standardmäßig wird keine Prüfsumme generiert, dabei enthält das Feld den Wert Null. Ansonsten wird bei der Checksummenbildung derselbe Algorithmus wie bei IP verwendet. Wenn die errechnete Checksumme ein NULL ergibt, werden zur Unterscheidung vom Standardzustand alle Bits auf Eins gesetzt. Zur Berechnung werden mit einem Pseudo-Header zusätzliche Daten, die selbst nicht im UDP-Datagramm enthalten sind, mit einbezogen. [1] [2] [7]

2.4 Prozess Variablen Interface

Die Steuerungssoftware Automation Studio der Firma B&R stellt in ihrer Software den Kommunikationstreiber PVI (Prozess Variablen Interface) zur Verfügung um Daten zwischen einem PVI Client und einer Steuerung auszutauschen. Der PVI Manager als zentrale Komponente des PVI ist für die Verwaltung von Prozessdaten wie z.B. Prozessvariablen, Listen oder Datenobjekte, zuständig. Der PVI Manager führt sowohl die zeitliche als auch die richtungsorientierte Organisation der Prozessdaten durch, d.h. der Manager koordiniert aus der Anwenderkonfiguration des Automation Studios, wo z.B. die Datenübertragungsrichtung, das Protokoll und das Medium festgelegt werden kann, die Datenübertragungen. Die PVICOM Schnittstelle stellt den Zugang zum PVI auf der untersten Ebene her. Sie wird von allen Windows-basierenden Komponenten mit PVI Zugang verwendet. Der PVI Manager wird als Windows Service oder als Windows Prozess gestartet. Er verwaltet alle Prozesse in einer Objektstruktur, der sogenannten Objekthierarchie. Dabei übernimmt jedes Prozessobjekt eine bestimmte Aufgabe und wird mittels eines Pfadnamens und dazugehörigen Parametern definiert. [9]

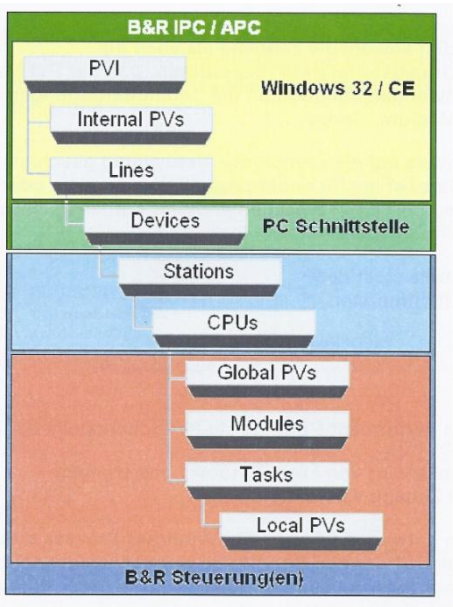

Abbildung 6 PVI Objekthierarchie Quelle: [9]

Ein Prozessobjekt repräsentiert entweder einen bestimmten logischen Wert bzw. physikalischen Teil der Kommunikationsverbindung oder ein Objekt auf der Steuerung. Es wird durch seinen Namen, seinen Typen und die Anschlussbeschreibung definiert. Es kann sowohl statisch als auch temporär eingerichtet werden. Die Arbeitsweise beider Arten ist hierbei aber gleich. Ein statisches Prozessobjekt wird nur einmal eingerichtet und bleibt während der gesamten Laufzeit des PVI Managers erhalten. Temporäre Prozessobjekte werden zusammen mit Verbindungsobjekten, also aktiven Prozessobjekten, eingerichtet. Bei der Freigabe des Verbindungsobjektes oder beim Beenden der PVICOM Anwendung wird auch das temporäre Prozessobjekt wieder freigegeben. Die PVI Client Anwendung wird bei dem Vorhandensein von Ereignisdaten und bei Datenänderung von aktiven Prozessobjekten benachrichtigt.

Die PVI Kommunikation erfolgt asynchron, d.h. dass das Senden und Empfangen von Daten zeitlich versetzt und ohne Blockieren des Prozesses durch Warten auf Empfangs- oder Sendebestätigungen erfolgt.

Die Kommunikation zwischen der Windows PVI Anwendung und der Steuerung erfolgt über die sogenannte PVI Linie. [9]

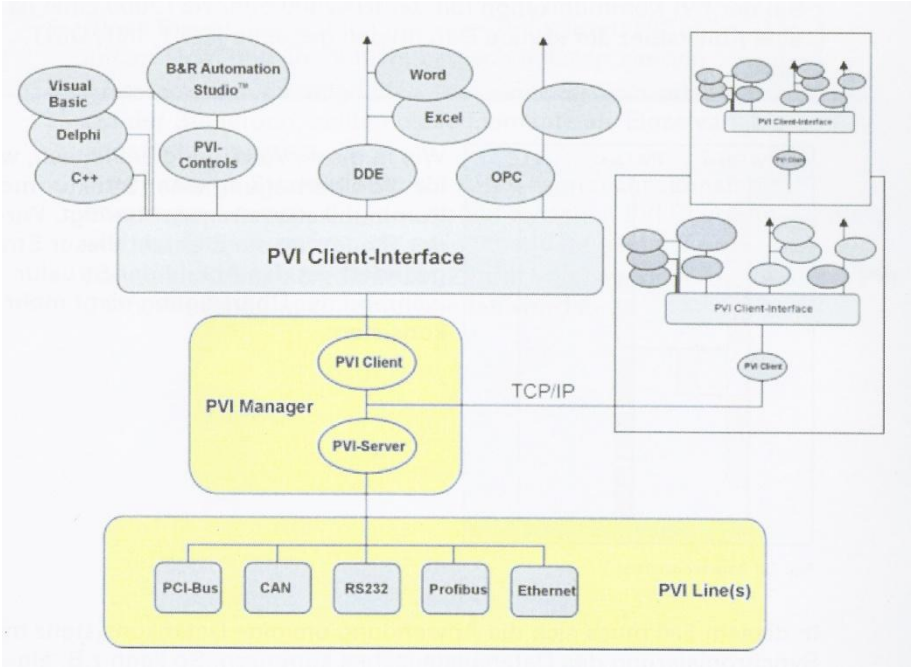

Abbildung 7 PVI Linien Kommunikation Quelle: [9]

Als Basis wird die INA200 Linie verwendet, da diese PVI Linie die am meisten verwendete Kommunikationsart darstellt. Immer wenn ein neues Prozessobjekt mit Linienanschluss neu eingerichtet wird, löst diese einen Vorgang zum Verbindungsaufbau oder zur Objektidentifizierung aus. Wird das Prozessobjekt freigegeben und wieder neu eingerichtet, wird dieser Vorgang wiederholt. Die Unterbrechung der Verbindung zwischen PC und Steuerung z.B. durch das Abstecken des Verbindungskabels, führt nach Ablauf des Response Timeouts, in welchem die Wartezeit auf eine Antwort konfiguriert werden kann, zu einem erneuten Verbindungsaufbau, sofern die Hardwareverbindung wieder hergestellt wurde.

Wird ein Objekt temporär erzeugt, erfolgen nach der Identifizierung des Objektes ein automatischer Leseauftrag und somit auch ein Leseauftrag zur Steuerung. Bei einer Datenänderung wird ein Datenereignis an die PVI Client Anwendung geschickt. Statische Objekte werden nur identifiziert. Erst nach dem Erzeugen eines Verbindungsobjektes wird das Objekt gelesen und an die PVI Client Anwendung geschickt. [9]

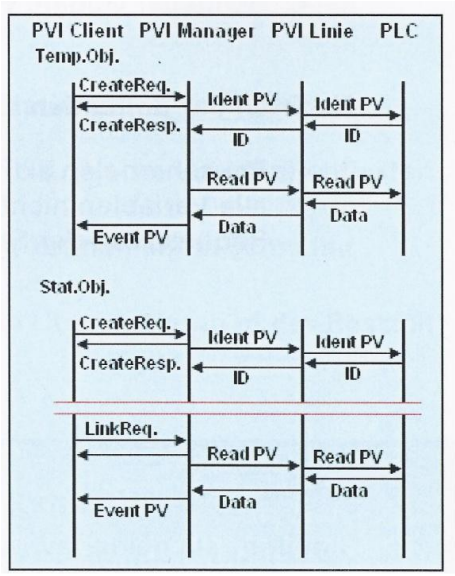

Abbildung 8 PVI Verbindungsaufbau Quelle: [9]

Unterschieden wird in der PVI Kommunikation zwischen pollender und ereignisgesteuerter Kommunikation. Fällt die Entscheidung darauf mit der pollenden Kommunikation zu arbeiten, werden aktive Prozessvariablen zyklisch von der Steuerung ausgelesen. Der PVI Manager vergleicht den Wert des internen Prozessabbildes mit dem gelesenen Wert und schickt bei jeder Datenänderung ein Datenereignis an die PVI Client Anwendung. Wird hingegen die ereignisgesteuerte Kommunikation verwendet, wird mit Eventvariablen gearbeitet. Die Steuerung überwacht die Eventvariablen mit der eingestellten Refreshzeit der Steuerung. Mit dieser Zeit wird die zeitliche Aktualität eines Variablenobjektes vorgegeben. Bei einer Datenänderung auf der Steuerung wird dem PVI mitgeteilt, dass sich Eventvariablen geändert haben. Diese Variablen werden dann von der PVI Linie gelesen. [9]

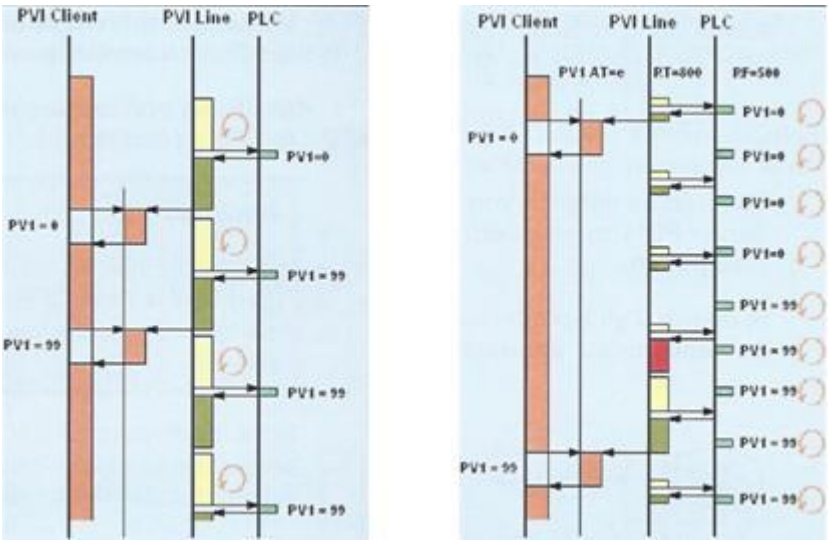

Abbildung 9 PVI - pollende und ereignisgesteuerte Verbindung Quelle: [9]

Mit PVI ist es außerdem möglich eine Fernwartung von Anlagen mittels einer PVI Remote Verbindung über ein Netzwerk durchzuführen. Dafür ist es notwendig, dass zwischen den PCs eine TCP/IP-Verbindung besteht. Dabei ist es möglich von einem PC eine Verbindung auf mehrere PVI Server PCs zu erstellen oder von mehreren PCs auf einen PVI Server PC zuzugreifen. In diesem Fall befindet sich der PVI Manager und die PVI Client Verbindung auf unterschiedlichen PCs. Eine Remote Verbindung ist also sowohl eine einfache Client/Server Architektur als auch eine Multi Client/Multi Server Architektur. [9]

2.5 OPC

Ursprünglich ist OPC die Abkürzung für OLE in Process. OLE wiederum steht für Object Linking and Embedding. Dies bezeichnete bis in die Mitte der 1990er Jahre die gesamte Microsoft-Technologie für die Realisierung objekt-orientierter Systeme. In dieser Zeit begann die Entwicklung von OPC. Deshalb findet sich dieses Akronym in der ursprünglichen Bedeutung, OLE for Process Control, in den drei Buchstaben „OPC“ wieder. OLE wird heute nur noch für das dynamische Verknüpfen von Objekten in unterschiedlichen Office-Anwendungen verwendet.

Da mit der Ausbreitung der Automatisierungsprodukte auch die Zahl an Kommunikationsprotokollen und Bussystemen stieg, wuchs der Druck der Hersteller etliche von Treibern zu entwickeln und zu warten. Im Jahr 1995 taten sich deshalb die Firmen Fisher-Rosemount, Intellution, Intuitive Technology, Opto22, Rockwell und die Siemens AG zusammen, um eine Lösung für dieses Problem zu erarbeiten und bildeten die OPC Task Force. Auch Vertreter von Microsoft waren daran beteiligt und lieferten technische Unterstützung. Basierend auf Microsofts DCOM-Technologie sollte ein Standard für den Zugriff auf Echtzeitdaten unter Windows-Betriebssystemen gefunden werden, da 90 Prozent (Stand 2002) aller Industrierechner mit diesem Betriebssystem laufen. Bereits 1996 wurde die OPC Specification Version 1.0 veröffentlicht und die OPC Foundation gegründet, die seitdem alle Spezifikations- und Marketingarbeiten koordiniert und heute mehr als 400 Mitgliedsfirmen (Stand 2012) hat. [4]

OPC basiert auf einer Client/Server-Struktur. Ein Client (master) hat Zugang zu einem oder mehreren Servern (slaves). Mehrere Clients können dementsprechend nur einen Server bedienen. Obwohl ein Client Daten zum Server übertragen kann, ist die hauptsächlich verwendete Datenflussrichtung vom Server zum Client. OPC ist im ISO/OSI Modell in der Anwendungsschicht angesiedelt und definiert auch die Präsentations-, Sitzungs- und Transportschicht. [5]

7	Application Layer	OPC DA, AE, HD etc.	OPC DA XML
6	Presentation Layer	COM/DCOM (Distributed) Component Object Model	SOAP, XML
5	Session Layer	RPC Remote Procedure Calls	HTTP
4	Transport Layer	TCP/IP	
3	Network Layer		
2	Link Layer		
1	Physical Layer		

Abbildung 10 OPC im ISO/OSI Model Quelle: [15]

OPC beinhaltet eine Anzahl von Schnittstellen, die verschiedenen Zwecken dienen.

OPC Overview and OPC Common Definitions and Interfaces

Diese Spezifikationen enthalten informative Angaben zu den Themen Einsatzgebiet, Basistechnologie und verfügbare Spezifikationen. Außerdem sind allgemeine Definitionen und Interface-Beschreibungen enthalten. [17]

OPC Data Access Specification

Dies ist die älteste aller OPC-Spezifikationen. Hier wird eine Schnittstelle zwischen Client- und Server-Programmen zur Prozessdatenkommunikation definiert. Dabei ermöglichen Server vom Typ Data Access einem oder mehreren Data Access Clients den transparenten Zugriff auf verschiedenste Datenquellen (z.B. Temperatursensoren) und Datensenken (z.B. Regler). Diese Quellen und Senken können sich auf direkt im Computer gesteckten I/O Karten befinden, aber auch auf beliebigen Geräten wie Reglern, Ein-/Ausgabemodulen u.a. liegen, die über serielle Verbindungen oder über Feldbusse angeschlossen sind. Ein Client vom Data Access kann auf mehrere Data Access Server zugreifen.

Data Access Server können einfache Programme sein, aber auch komplexere Programme, die den Zugriff auf eine Vielzahl von Variablen in einer großen Anzahl von Geräten über umfangreiche Kommunikationsmechanismen ermöglichen, sind realisierbar. In der Spezifikation sind keine Vorgaben oder Einschränkungen hinsichtlich der Implementierungsaspekte enthalten.

In der Data Access Specification werden zwei unterschiedliche Konzepte definiert, die ein Data Access Server implementieren und ein Data Access Client nutzen kann – den Namensraum (namespace) und die OPC-Objekthierarchie. [17] [18]

OPC Alarms and Events Specifications

Hier wird eine Schnittstelle definiert, welche zwischen Client- und Serverprogrammen zum Strukturieren, Übertragen und Quittieren von Ereignissen (events) und Alarmen (alarms), angesiedelt ist. Der Alarms and Events Server kann aus verschiedenen Datenquellen Werte erfassen, auswerten und entscheiden, ob ein Ereignis aufgetreten ist. Die Datenquellen können sich, wie auch die Datenquellen- und senken, auf direkt im Computer gesteckten I/O Karten befinden, aber auch auf beliebigen Geräten wie Reglern, Ein-/Ausgabemodulen u.a. liegen, die über serielle Verbindungen oder über Feldbusse angeschlossen sind. Ein Alarms and Events Server schickt keine Werte an den Client, sondern eine Information darüber, dass sich etwas ereignet hat (z.B. dass eine Temperatur einen Grenzwert überschritten hat). Die Spezifikation legt aber nicht fest, wie die Entscheidung durchzuführen ist oder die Kriterien anzugeben sind. Es ist daher für den Benutzer möglich, Entscheidungskriterien für beliebige Größen (z.B. analog oder binär) und Verknüpfungen von Größen (z.B. Temperatur und Uhrzeit) zu definieren. Ein Alarms and Events Client kann Bestandteil von Auszeichnungskomponenten, Bedienerstationen oder Managementsystemen sein. Möglich sind einfache Komponenten wie z.B. ein Excel-Sheet oder umfangreiche Programme wie z.B. Prozessleitsysteme. [18]

OPC Historical Data Access Specification

Servern vom Typ Historical Data Access ermöglichen Clients den Zugriff auf historische Daten. Dabei wird zwischen Rohdaten, also den aufgezeichneten historischen Daten, und den Daten, die aus diesen Rohdaten gewonnen werden (Aggregate), unterschieden. Die aggregierten Daten werden erst erstellt, wenn der Client die Anweisung dafür gibt. Auf die Daten kann lesend, schreibend oder verändernd zugegriffen werden. In der Historical Data Access Specification werden zwei unterschiedliche Strukturen definiert. Zum einen eine Struktur, die ein Historical Data Access Server implementieren kann – den Adressraum, wo alle historischen Daten gespeichert sind, und eine Struktur, die ein Historical Data Access Client nutzen kann – die Objekthierarchie. [4]

OPC Batch Specification

Im Gegensatz zu den Spezifikationen Data Access, Alarms and Events und Historical Data Access beschreibt die Batch Specification keine vollständig neue Schnittstelle zwischen einem Client und einem Server. Sie definiert lediglich Ergänzungen zur Data Access Specification für den Einzelfall Batchprocessing. Ein Batch besteht aus verschiedenen Rezepturen, die die Herstellung eines Produktes beschreiben. Wird dieser Batch abgearbeitet, findet ein Datenaustausch mit Geräten statt. Dabei werden Auftragsdaten

gesendet und Reportdaten empfangen. Die OPC Batch Specification ermöglicht Interoperabilität zwischen Produkten der Batch-Verarbeitung und beschreibt die Möglichkeit, dass Lösungen verschiedener Hersteller in einer heterogenen Umgebung operieren können. [4]

OPC Security

Die bisher beschriebenen Spezifikationen definieren Schnittstellen zwischen OPC-Clients und OPC-Servern. Diese müssen nicht auf einem PC laufen, sondern können auf mehrere Computer verteilt sein. Da aber nicht immer jeder Nutzer mit seinem Client auf einen anderen Server zugreifen soll, müssen hierfür Sicherheitsfestlegungen definiert werden. OPC Security definiert drei Sicherheitsebenen:

Disabled Security, keinerlei Security wird unterstützt. Auf den OPC-Server kann von allen Nutzern zugegriffen werden und der Zugriff auf herstellerspezifische Security-Objekte wird nicht kontrolliert.

DCOM Security, Start- und Zugriffsrechte sind auf ausgewählte Clients, also Nutzer, beschränkt. Wie bei der Disabled Security wird auch hier der Zugriff auf hersteller-spezifische Security-Objekte nicht überwacht. Diese Ebene ist die Standardeinstellung für DCOM.

OPC Security, herstellerspezifische Security-Objekte im Server werden vom OPC Server überwacht, der als Referenzmonitor fungiert. Ein OPC-Server kann OPC.Security allein implementieren oder aber auch gemeinsam mit der DCOM Security laufen. [4]

OPC Compliance Test

Mit dem Compliance Test wird eine Lösung zur Verfügung gestellt, die einen Test von Data Access, Alarms and Events und Historical Data Access Servern ermöglicht. Es handelt sich dabei um einen Testframework, verschiedene Testfälle und den OPC Compliance Client. Die Tests überprüfen, ob das Produkt des Herstellers den OPC-Spezifikationen entspricht. Außerdem kann die Performance und die Robustheit des Produktes bewertet werden, allerdings nur dann, wenn es eine Vergleichsgröße z.B. einen Mess-Server für einen Performancetest gibt. [4] [17]

OPC and XML

Die eXtensible Markup Language (XML) ist eine flexible Datenbeschreibungssprache und gilt als Nachfolger von HTML (Hypertext Markup Language). XML wird also verwendet, wenn es um eine Integration von Web-Anwendungen mit anderen Bereichen geht. In der Automatisierungsindustrie wird XML besonders häufig zur Dokumentenerstellung verwendet. Produkte, die auf der OPC XML Spezifikation basieren, sollen den Austausch von OPC

Daten innerhalb und zwischen verschiedenen Hierarchien (z.B. Feldebene - Unternehmensleitebene) der automatisierten Produktion vereinfachen. Diese Produkte stehen durch die Verwendung von XML aber auch für einen größeren Umfang an Plattformen (z.B. SPS oder Embedded Devices) zur Verfügung. Die Spezifikation beschreibt also XML-Schnittstellen für den Austausch von Data Access Daten. [4]

OPC Data Access Specification 3.0

Diese Spezifikation ist eine Erweiterung der ursprünglichen Data Access Specification. Im Folgenden werden nur kurz die bearbeiteten Punkte aufgelistet:

- Kodieren und Dekodieren von Strukturinformationen über Prozessgrößen
- Lesen und Schreiben von Werten von und zu peripheren Geräten
- Verwendung von XML zum Kodieren und Dekodieren der Strukturinformation
- Definition eines Interfaces für die Kommandobearbeitung
- Definition eines Verfahrens für das Lesen und Schreiben von Werten
- Senden von Kommandos und Überwachen des Abarbeitungsstandes

OPC Data eXchange (DX)

OPC DX-Server ermöglichen den direkten Austausch von Daten ohne Einbeziehung eines Clients. Dies erfolgt dabei direkt von einer Quelle (source) in einem Server zu einer Senke bzw. einem Ziel (target) in einem anderen Server. Dabei befinden sich die Datenquellen in Data Access- und DX-Servern, Datenziele hingegen nur in DX-Servern. Die Funktionalität eines Data Access Clients muss im DX-Server implementiert sein. Besteht für den Austausch eines Wertes eine Verbindung zwischen Servern, wird dies mit DX Connection betitelt. Eine DX Configuration besteht aus allen Connections eines Servers, was allen Verbindungen zwischen Datenquelle und –ziel entspricht. Festgelegt werden diese durch einen Konfigurationsclient. [4]

OPC UA

Mit der Einführung von OPC UA (Unified Architecture) wurde 2006 auch die Bedeutung der Buchstaben von OPC geändert. Was ursprünglich für OLE in Process stand, steht heute für Openess, Produductivity, Collaboration. Diese OPC Spezifikation unterscheidet sich von ihren Vorgängern insoweit, dass es nicht mehr nur möglich ist Maschinendaten zu transportieren, sondern diese auch maschinenlesbar semantisch zu beschreiben. Vor Veröffentlichung von OPC UA war OPC an DCOM und dementsprechend auch an Windows gebunden. Nun wurde ein eigener Kommunikationsstack entwickelt, der DCOM ersetzt. Die einzelnen Spezifikationen, aus denen OPC UA besteht, sollen hier nur aufgelistet, aber nicht

näher erläutert werden. Sie sind nur von großer Wichtigkeit, wenn der Stack portiert oder ein eigener implementiert werden soll, da sie die UA-Interna beschreiben, die vom Kommunikationsstack gehandelt werden.

Teile der OPC UA Spezifikation:

- Concepts
- Address Space Model
- Information Model
- Profiles
- Alarms and Conditions
- Historical Access
- Aggregates
- Security Model
- Services
- Mappings
- Data Access
- Programs
- Discovery

[18]

2.6 Dual Ported Memory

Als Dual Ported Memory oder auch Shared Memory wird eine Kommunikation von Prozessen über einen gemeinsamen Speicher bezeichnet. Dies wird auch speicherbasierte Kommunikation genannt. Shared Memory Segmente sind Speicherbereiche, auf die mehrere Prozesse direkt zugreifen können. Damit ein Speicherbereich zu einem Shared Memory Segment werden kann, muss eine bestimmte Funktion des Betriebssystems aufgerufen werden. Der Speicherbereich wird dann als Shared Memory Segment registriert. Sehr wichtig ist, dass die Prozesse, die auf ein Shared Memory Segment zugreifen, ihre Lese- und Schreibzugriffe selbst sicherstellen und koordinieren, sodass sich ihre Speicherzugriffe gegenseitig ausschließen. Der Empfänger, also der lesende Prozess, darf erst dann lesen, wenn der schreibende Prozess (Sender) etwas fertig in den gemeinsamen Speicher geschrieben hat. Passiert diese Koordination nicht stetig, kommt es zu Inkonsistenzen.

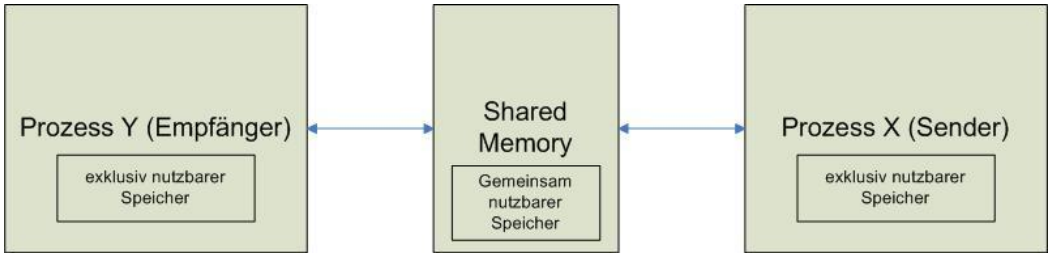

Abbildung 11 Zugriff auf Shared Memory Quelle: eigene Darstellung

Das verwendete Betriebssystem speichert eine Shared Memory Tabelle, die Informationen über die aktuell existierenden Shared Memory Segmente beinhaltet. Diese Informationen bestehen aus der Anfangsadresse im Speicher, der Größe, dem Besitzer (Username und

Gruppe) und den Zugriffsrechten. Angesprochen wird ein Shared Memory Segment immer über die dazugehörige Indexnummer in der Shared Memory Tabelle. Diese Nummer entspricht einem Abschnitt im Speicher.
Das Prinzip des gemeinsamen Speichers wird auch in der Prozess- und Automatisierungstechnik verwendet, um eine Datenübertragung zwischen einer Steuerung und einer Anwendungssoftware zu ermöglichen. Hier teilen sich Computer und Steuerung den Speicher. Eine sogenannte Slot-SPS in Form einer Einsteckkarte wird hierfür in den PC integriert. Die SPS ist unabhängig vom Betriebssystem des PCs und kann mit externer Spannung und Pufferbatterie versorgt werden, um zu gewährleisten, dass die SPS nicht ausfällt, wenn der PC ausfällt. Die Slot-SPS besitzt Anschlüsse für Feldbussysteme, sodass auch der Zugriff auf dezentrale Aktoren und Sensoren realisiert werden kann. [19] [20]

3 Analyse

Die in Kapitel 2 beschriebenen Schnittstellen bzw. Transportprotokolle müssen auf ihre Eignung als Übertragungssystem und Kompatibilität mit der Automation Studio Software von B&R überprüft werden. Die Vor- und Nachteile jedes Übertragungssystems sind gegenüber zu stellen und zu bewerten. Außerdem muss die Tauglichkeit als Schnittstelle sowohl für das Projekt Asterix als auch als Standardschnittstelle in Anlehnung an das Lastenheft bewertet werden.

3.1 Lastenheft

Um eine geeignete Schnittstelle für das Projekt Asterix zu finden, ist es notwendig die Anforderungen an diese genau zu spezifizieren. Hierfür ist es nötig, alle zur Diskussion stehenden Transportprotokolle auf diese Anforderungen hin zu untersuchen. Die Protokolle, die zurzeit in der YXLON International GmbH in verschiedenen Anlagentypen realisiert sind (siehe Anhang), wurden auf sechs verschiedene Eigenschaften untersucht und miteinander verglichen. Hierfür wurde eine Tabelle erstellt (siehe Anhang), die übersichtlich zusammenfasst, welche Eigenschaften die einzelnen Protokolle aufweisen. Die relevantesten Kriterien für die Auswahl der Schnittstelle für das Projekt Asterix sind eine sehr hohe Dynamik, eine sichere Kommunikation, geringe Kosten und eine gute Kompatibilität zur B&R Software Automation Studio, sowie zur B&R Hardware. Das wichtigste Kriterium ist aber die hohe Dynamik, also eine sehr schnelle Kommunikationsgeschwindigkeit. Diese soll bei höchstens 10 ms liegen. Für das Projekt Asterix ist diese enorm hohe Geschwindigkeit nicht allzu relevant, da es sich um eine Anlage handelt, die immer von Hand bedient wird.

Das bedeutet, dass die Reaktionszeit des bedienenden Menschen in jeden Fall sehr viel höher (>100 ms) ist als die Kommunikationszeit zwischen PC und Steuerung. [4b]
Bei den Anlagen, die an einen vollautomatisierten Betrieb angebunden sind und automatisch laufen, sind diese Performancekriterien schon von sehr viel höherer Bedeutung. Hier ist es von großer Relevanz, dass wichtige Daten, wie zum Beispiel die Position, schnell an den PC und somit die Bildverarbeitungssoftware weitergegeben werden können. Denn in einem vollautomatisierten Betrieb, wie zum Beispiel einem Reifenhersteller, ergeben niedrige Durchlaufzeiten des Produktes innerhalb der Produktionskette eine höhere Effizienz und somit mehr Umsatz in kürzerer Zeit. Außerdem ist eine zuverlässige Kommunikation enorm wichtig, die Daten müssen verlässlich ohne Datenverlust vom Sender zum Empfänger gesendet werden.

3.1.1 Automation Studio

Im Rahmen des Projekts Asterix wurde definiert, dass das Steuerungskonzept vollständig mit B&R Hard- und Software realisiert werden soll. B&R bietet als Programmierumgebung das „Automation Studio“ an, welches die Steuerung, Antriebstechnik und Visualisierung umfasst. Im Rahmen dieser Arbeit wurde die Software ausschließlich hinsichtlich der Steuerung bzw. als Programmierumgebung zum Kommunikationsmedium genutzt.
Das Automation Studio stellt verschiedene Programmiersprachen zur Verfügung. Die grafischen Sprachen Kontaktplan (LAD), Funktionsplan (FBD) und Kontinuierlicher Funktionsplan (CFC) sowie die textuellen Sprachen Anweisungsliste (IL), Automation Basic (AB), Ansi C (C) und Strukturierter Text (ST). Außerdem die Ablaufsprache (SFC), die sowohl grafisch als auch textuell arbeitet. Es ist möglich in einem Projekt mehrere Programmiersprachen zu verwenden und zu mischen. Das Automation Studio stellt für alle Sprachen dieselben Diagnosewerkzeuge zur Verfügung, sodass die Entscheidung, welche Sprache im Projekt verwendet werden soll, nicht von den Diagnosefunktionen abhängig ist.
Im Rahmen der Standardisierung soll bei dem Projekt Asterix ausschließlich in Strukturiertem Text erstellte Software verwendet werden. Dieser Beschluss galt somit auch als Anforderung für jegliche im Rahmen dieser Arbeit erstellte Software.
Das Automation Studio stellt Bibliotheken mit Beispielprogrammen und Funktionen zur Verfügung. Auch diese sind weitestgehend unabhängig von der verwendeten Programmiersprache.
Die Oberfläche des Automation Studios ist in drei Teile aufgeteilt: den Projektexplorer, den Workspace und das Meldungsfenster. Im Workspace erfolgt die Programmierung durch den Anwender. Im Meldungsfenster werden zum Beispiel Compiler- und Debugger-Meldungen

ausgegeben. Außerdem werden dort die Suchergebnisse der Funktion „Find in Files“ ausgegeben. Der Projektexplorer ist wiederum ebenfalls in drei Ansichten untergliedert: die Logical View, die Configuration View und die Physical View. In der Logical View wird die Projektstruktur dargestellt. Hier befinden sich alle im Projekt erstellten Programme und die dazugehörigen Variablentabellen, außerdem dem Projekt zugefügte Bibliotheken und Beispielordner. In der Configuration View werden die erstellten Hardwarekonfigurationen angezeigt. Es ist möglich mehrere Konfigurationen zu erstellten und innerhalb eines Projektes zu wechseln. In der Physical View können die Programme den einzelnen Zeitscheiben zugeordnet und deren Zykluszeiten geändert werden. [24]

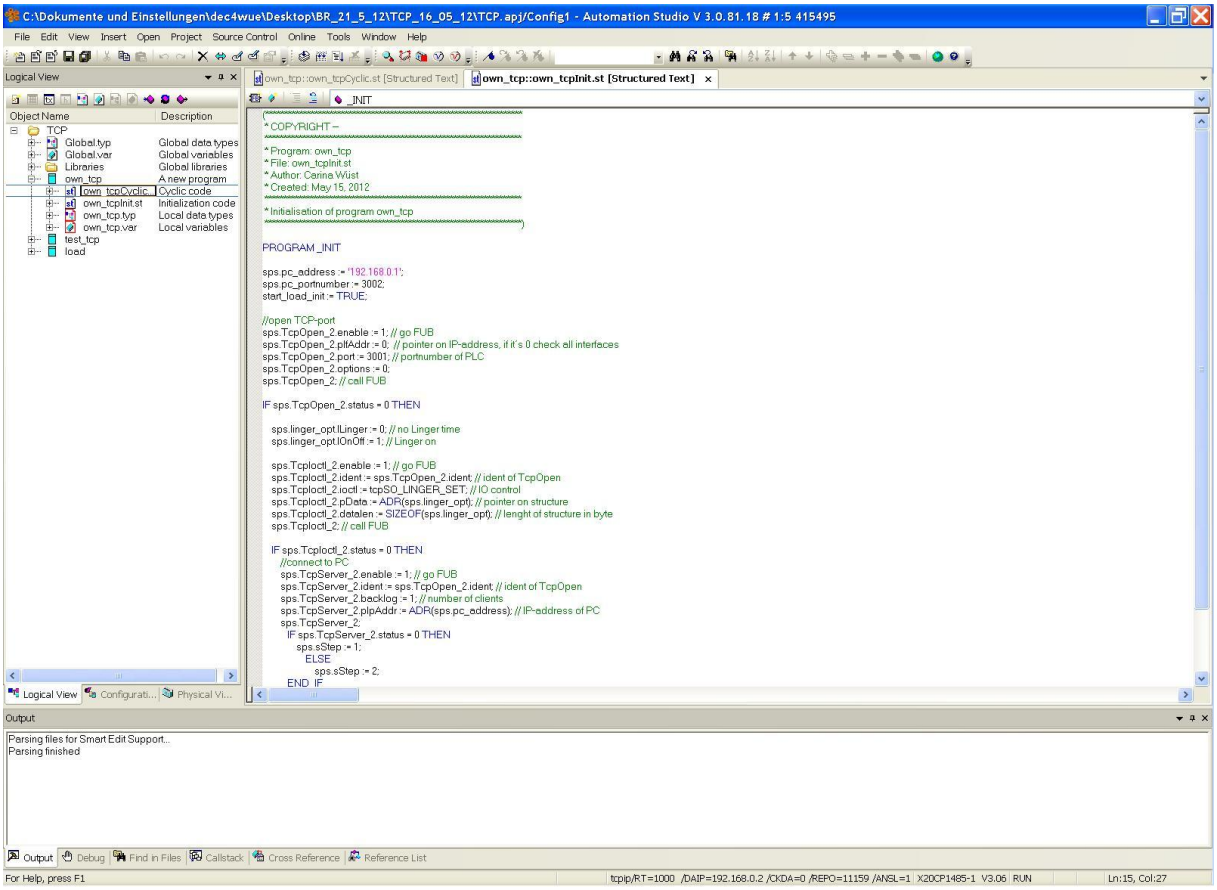

Abbildung 12 Screenshot Automation Studio

Dieses Zeitscheibensystem unterscheidet Automation Studio von vielen anderen Entwicklungsumgebungen anderer Steuerungshersteller. Dieses System funktioniert so, dass die einzelnen Programme des Projektes jeweils einer Zeitscheibe zugeordnet werden. Jeder Zeitscheibe kann nur ein Programm zugeteilt werden. Die Zykluszeiten der Zeitscheiben dürfen nicht gleich sein. Es werden acht Zeitscheiben vom Automation Studio bereitgestellt, also können auch acht Programme quasi parallel laufen. Die Zeitscheiben werden mit Cycle #1 bis Cycle #8 betitelt. Cycle #1 wird mit höchster Priorität abgearbeitet,

Cycle #2 mit zweithöchster usw. Die Bearbeitung der Programme funktioniert nur deshalb nicht wirklich parallel, da das Programm des Cycle #2 nur dann abgearbeitet wird, wenn Cycle #1 nicht arbeitet. Die Abarbeitung der darunterliegenden Cycles erfolgt nach demselben Prinzip.

Das Automation Studio stellt viele Funktionen zur Diagnose zur Verfügung. Vieler dieser Funktionen arbeiten auch, wenn keine Hardware konfiguriert und online ist. Für diesen Fall stellt das Automation Studio eine Simulation mit dem Namen „AR000“ bereit, mit der es möglich ist, erstellte Programme auf Funktionstüchtigkeit und Verhalten zu testen.

Das einfachste und zugleich effektivste Diagnosewerkzeug ist die Watch-Funktion. Hiermit lassen sich die Werte der Variablen und ihre Veränderungen, sowie die Zustände der verwendeten Funktionen beobachten. Außerdem ist es hier möglich die Variablen auf einen definierten Wert zu setzen. Diese Funktion nennt sich „FORCE“. Damit kann also analysiert werden, wie sich das Programm verhält, wenn die Variablen bzw. Ein- oder Ausgänge ihren Wert ändern.

Das Betriebssystem protokolliert alle schwerwiegenden Fehler, wichtige Warnungen und Hinweise, die im Rahmen einer Anwendung auftreten. Dieses Protokoll wird in einem Systemlogbuch im Speicher der Steuerung abgelegt. Diese Logdatei kann über das Hauptmenü geöffnet und eingesehen werden. [11]

Wie es aus der Hochsprachenprogrammierung bekannt ist, ist es auch im Automation Studio möglich, die in einer textuellen Programmiersprache erstellten Programme zu debuggen. Hiermit können die Programme auf etwaige Programmierfehler untersucht werden. Durch Setzen von Haltepunkten im Sourcecode wird das Programm beim Abarbeiten an der Codezeile angehalten, an der der Haltepunkt gesetzt wurde. Es ist aber auch möglich, das Programm ohne Haltepunkt Schritt für Schritt zu durchlaufen. Hiermit kann geprüft werden, ob das Programm genauso abläuft, wie es beabsichtigt ist.

Zum Betrachten von veränderlichen oder statischen Variablen ist außer der Watchfunktion auch die TRACE-Funktion sehr hilfreich. Der Tracer zeichnet den zeitlichen Verlauf von Werten auf und stellt diesen in Form eines Diagramms dar. Somit ist es möglich, Werte aus verschiedenen Taskklassen in zeitlich definiertem Zusammenhang aufzuzeichnen.

Mit dem Profiler ist es möglich, Systemdaten zu messen und anzeigen zu lassen. Diese Funktion kann nur verwendet werden, wenn eine SPS online ist. Es können die Tasklaufzeiten, die Stackauslastung sowie die Systemauslastung angezeigt werden. Diese Funktion ermöglicht also eine genaue Analyse des Verhaltens der SPS. Diese Daten können zum Beispiel zur Optimierung der Systemauslastung verwendet werden. [11] [11b]

3.1.2 X600

Die Steuerungssoftware muss mit der YXLON internen PC-Software X600 zusammen arbeiten. Sie ist eine von mehreren Softwareplattformen, die in den verschiedenen Anlagentypen der YXLON International GmbH verwendet werden. Programmiert wird Software in der Hochsprache C# (sprich: C sharp), diese ist die aktuelle Software von Microsoft und wird auch mit „managed code“ bezeichnet. Das X600 besteht aus vier Komponenten:

- GUI (auch im folgenden mit „Runtime“ betitelt)
- Control.6000
- Alarm and Event
- User Management

Das Graphical User Interface ermöglicht dem Endkunden Anzeige- und Eingabemöglichkeiten, z.B. Röntgenbilder und Achspositionen. Der Kunde hat nur auf diese Komponente des X600 Zugriffsrechte. Dem Service Mitarbeiter ist es jedoch möglich, auch direkt beim Kunden vor Ort durch erweiterte Benutzerrechte mit den Diagnosefunktionen zu arbeiten.

Control.6000 gewährleistet die Kommunikation mit externen Geräten wie der SPS, der Blendensteuerung oder dem Röntgensteuerteil und sorgt für die Verwaltung von Prüfprogrammen oder Rezepturen. Es ist im sogenannten Plugin-Design aufgebaut. Der Kern beinhaltet einen Datenpunktbrowser, mit dem es möglich ist Variablen zu forcen oder zu disablen und auf diese Weise Diagnosemöglichkeiten bereitstellt. Des Weiteren bietet er auch die Möglichkeit des hierarchischen Loggings bzw. Tracings, was zur Fehlerdiagnose ebenso hilfreich ist wie der Datenpunktbrowser. Jedes Plugin kann ein eigenes Debug-GUI erhalten, welches nur für die Softwareentwicklung vorgesehen ist und vom Endkunden nicht aufgerufen werden kann.

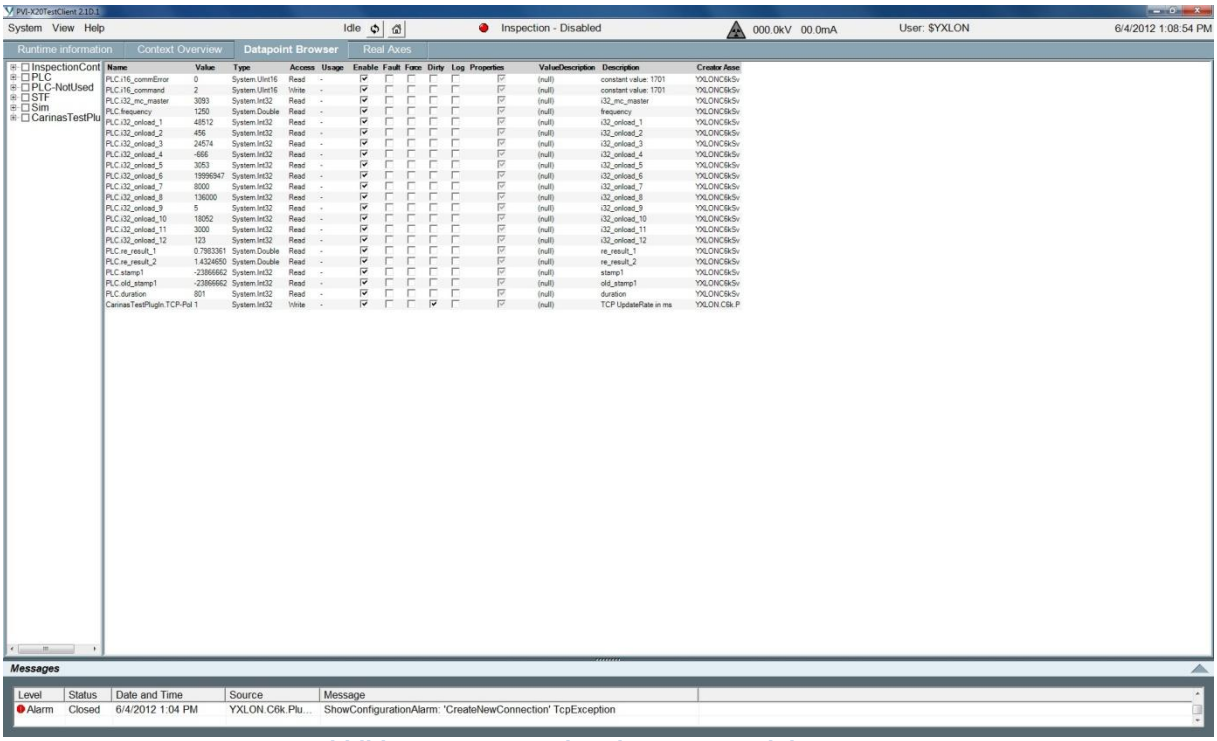

Abbildung 13 Screenshot des Datenpunktbrowsers

Das User Management ist für die Zugriffsrechte verantwortlich. Wie bereits erwähnt, ist es z.B. dem Endkunden nicht gestattet ohne besondere Schulung auf den Datenpunktbrowser zuzugreifen. Die Zugriffsregelung wird durch das User Management sichergestellt.

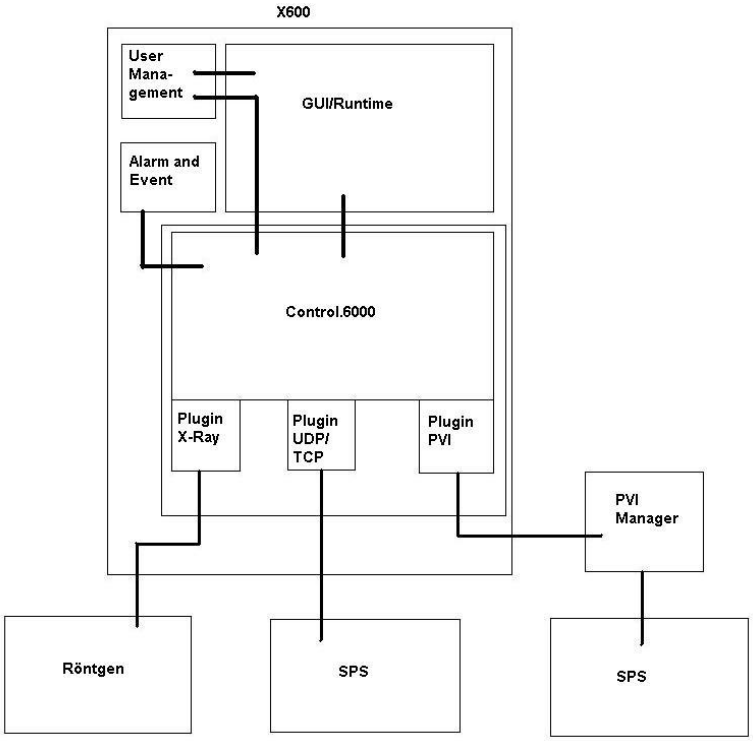

Abbildung 14 Aufbau der X600 Software Quelle: eigene Darstellung

Bei der Auswahl der Kommunikationsschnittstelle für das Projekt Asterix, bzw. zur Standardschnittstelle ist auch die Seite des Rechners mit einzubeziehen. Diese Arbeit fokussiert zwar die Steuerungsseite, dennoch soll hier ein Einblick gewährt werden, wie die PC-Seite die verschiedenen Transportprotokolle verarbeitet.
Für die verschiedenen Protokolle werden jeweils neue Plugins geschrieben. Im Folgenden werden einige der zur Auswahl stehenden Transportdienste erläutert.

PVI

Um eine Verbindung zwischen SPS und PC herzustellen, muss auf dem Rechner der PVI Manager installiert werden. Um das Plugin zu programmieren, stellt B&R Unterlagen und Beispielprogramme, wie der Quellcode für die Kommunikation aufgebaut ist, zur Verfügung. Der C# Quellcode nutzt die von B&R mitgelieferte „BR.AN.PVIServices.dll“. Ein Vorteil der Verwendung von PVI ist, dass sich mehrere PVI-Applikationen über einen gemeinsamen PVI-Manager mit der angeschlossenen SPS verbinden lassen.

OPC

Ebenso wie PVI benötigt ein OPC-Client Plugin Fremdsoftware auf dem Rechner, in diesem Fall ist es die Software „OPC Server“. Das Plugin kommuniziert mit dieser Software. Dazu ist es von Nöten, dass der Rechner physisch mit der SPS verbunden ist. Diese OPC Server, die die eigentliche Kommunikation mit der Steuerung abwickeln, werden von verschiedenen Herstellern angeboten. Zurzeit wird in der YXLON International GmbH die OPC Data Access Schnittstelle verwendet, wozu die Dateien „opcrcw.comm.dll“ und „opcrcw.da.dll“ in den C#-Quellcode eingebunden werden müssen. Auch bei Verwendung von OPC ist es möglich, dass sich mehrere OPC-Clients über einen gemeinsamen OPC-Server mit einer SPS verbinden. Außerdem hat es den Vorteil, dass das Transportprotokoll durch seine Allgemeingültigkeit sehr schnell zu programmieren und implementieren ist.

UDP/TCP

Für die Verwendung eines UDP- bzw. TCP-Plugins, welches eine Punkt-zu-Punkt-Verbindung herstellt, ist es nicht nötig eine Fremdsoftware auf dem PC zu installieren. Allerdings muss der Softwareentwickler mehr Zeit in das Design des Plugins investieren, damit Verbindungsunterbrechungen (bei UDP außerdem Fehlerhandling) zur Steuerung entsprechend sinnvoll behoben werden. Außerdem muss festgelegt werden, wie die einzelnen Variablen in der Datenstruktur platziert bzw. kodiert werden. Dies muss weder bei der Verwendung von OPC noch bei PVI beachtet werden, da dort jede Variable über ihren

Namen adressiert wird. Ein weiterer Nachteil ist, dass mehrere Plugins nur mit viel Aufwand gleichzeitig auf eine SPS zugreifen können, diese benötigen jeweils einen separaten Port.
Um TCP zu realisieren benötigt es sehr viel mehr Aufwand als UDP, da es Sicherheitsmechanismen, wie z.B. automatische Korrekturen bei Übertragungsproblemen, beinhaltet.
Alle genannten Transportprotokolle basieren physisch auf einem Ethernet-Netzwerk. Einzig der PVI Manager kann außerdem eine serielle Verbindung (COM-Port) benutzen.

3.2 SWOT-Analyse

Das Ziel der SWOT-Analyse (strenghts, weaknesses, opportunities, threats) ist, eine oder mehrere geeignete Transportprotokolle aus den bereits bestehenden und beschriebenen auszuwählen. Deshalb werden die einzelnen Protokolle nacheinander auf ihre Vor- und Nachteile, in Anlehnung an die spezifizierten Anforderungen, analysiert.

3.2.1 TCP

Das Transmission Control Protocol realisiert die Transportschicht und arbeitet synchron und verbindungsorientiert. TCP fügt jeder Nachricht einen Header bei, der die vollständige und zuverlässige Übertragung der Nachrichten sicherstellt. Es wird also eine zuverlässige Kommunikation zugesichert, obwohl die TCP-Segmente, bestehend aus Header und Nutzdaten, letztlich als IP-Paket in einem Netzwerkprotokoll verpackt, im Netz weiterbefördert werden. Ob die Pakete, die losgeschickt wurden, auch tatsächlich alle richtig und vollständig ankommen, wird durch TCP kontrolliert. Anwendungen, die per TCP Daten versenden und empfangen, scheint es somit, als ob es dazu feste Verbindungsleitungen von einem Ende zum anderen gibt, über die die Daten in beide Richtungen ohne Unterbrechung durchströmen können. Es wird eine Zeitüberwachung durchgeführt, die beim Absenden jedes Pakets eine Kopie davon erstellt und einen Timer startet. Wenn die Übertragung nach Ablauf der Zeit nicht vollständig absolviert wurde, also keine Empfangsbestätigung vom Empfänger zurückkam, werden die Daten erneut gesendet. Solange die Bestätigungen zu den versendeten Pakten jedoch immer rechtzeitig beim Sender eingehen, erfolgt die Datenübermittlung kontinuierlich nach dem Prinzip des Three-Way-Handshakes.
Die Vor- und Nachteile von TCP lassen sich wie folgt zusammenfassen:

Vorteile:

- Keine Kosten durch Lizenzgebühren
- Weitestgehend plattformunabhängig
- Absicherung durch Sequenznummern und Prüfsummen
- Jedes fehlerfrei empfangene Paket wird vom Empfänger bestätigt
- Fehlerbehaftete Pakete werden verworfen
- Sender überwacht die Quittierung von Datensegmenten durch einen Timer
- Sicherung der Reihenfolge
- Gedoppelte Pakte werden verworfen
- Schnelle Kommunikationszeit

Nachteile:

- Hoher Implementierungsaufwand da „Neuentwicklung“

Das Transmission Control Protocol zeichnet sich durch viele Vorteile aus. Die Anforderungen der Plattformunabhängigkeit, zuverlässige und schnelle Übertragung (< 10 ms) sowie Vermeidung von Kostenentstehungen durch den Kauf von Lizenzen, werden allesamt erfüllt. Das Protokoll wird bereits bei Anlagen bei YXLON International GmbH verwendet, sodass die Vorteile bereits bekannt sind und verifiziert wurden. Allerdings wurde das Protokoll noch nicht auf B&R-Systemen verwendet, sodass die Eigenschaften darunter getestet werden sollten. Da das Automation Studio Funktionen zur Realisierung eines TC Protokolls bereitstellt, ist der Implementierungsaufwand als nicht besonders hoch einzuschätzen. Da dies aber nur eine Einschätzung ist, muss das Protokoll realisiert werden, um Gewissheit zu erlangen.

3.2.2 UDP

Das Datagram User Protocol ist ein einfaches Transportprotokoll, welches ohne Verbindungsaufbau und ohne Fehlerkorrektur arbeitet. Im Vergleich zu dem TCP-Header ist der Header eines UDP-Pakets sehr einfach aufgebaut. Er besteht aus insgesamt acht Byte. Ebenso wie beim Transmission Control Protocol gibt es eine Source bzw. Destination Port Number, welche die gleiche Bedeutung haben. Diese Nummern kennzeichnen zusammen mit der IP-Adresse den Service Access Point der UDP-Schicht. In einem Endsystem kann damit zwischen mehreren Applikationen unterschieden werden. Außer den Portnummern gibt es noch eine Längenangabe (UDP length) und eine Prüfsumme (UDP checksum). Wie bei TCP wird die Prüfsumme über das gesamte Paket und Teile des IP-Headers berechnet.

Somit ist UDP für kurze Nachrichten, die nur aus einem Paket bestehen, wie z.B. eine Abfrage bei einem DNS-Server, gut geeignet. Da die Wiederholung verlorener Pakete wegen der damit verbundenen großen Verzögerung wenig Sinn ergibt, wie z.B. bei der Sprachübertragung, wird UDP auch bei Echtzeitdiensten verwendet. Bei diesen Diensten kann in der Regel auch keine Flusssteuerung wie bei TCP erfolgen, da die Senderate durch den Dienst vorgegeben ist und nicht durch das Netz beeinflusst werden kann. [3]
Zusammenfassend ergeben sich also folgende Vor- und Nachtteile für die Verwendung von UDP:

Vorteile:

- Keine Kosten durch Lizenzgebühren
- Weitestgehend plattformunabhängig
- Hohe Übertragungsgeschwindigkeit

Nachteile:

- Keine Fehlermeldungen
- Verlorene Daten werden nicht erneut gesendet
- Mehrfachzustellungen möglich
- Ankunftsreihenfolge kann nicht vorhergesagt werden (Pakete können sich überholen)
- Keine Bestätigung
- Keine Synchronisation
- Pakete können verloren gehen
- Hoher Implementierungsaufwand da „Neuentwicklung“

Bei der Gegenüberstellung von Vor- und Nachteilen fällt auf, dass die Quantität der Nachteile überwiegt, qualitativ sind die Vorteile jedoch so überzeugend, dass UDP als ernsthafte Möglichkeit als Schnittstelle in Betracht gezogen werden muss. Das Automation Studio bietet fertige Funktionen zur Implementierung eines User Datagram Protocols an, sodass der Aufwand, der betrieben werden muss, um UDP umzusetzen, nicht besonders hoch ist. Außerdem zeichnet sich UDP durch seine Plattformunabhängigkeit aus und ist somit nicht nur für das Projekt Asterix interessant, sondern auch zur Verwendung als Standardschnittstelle geeignet. UDP wird auch schon in den CT-Anlagen der Firma YXLON verwendet, sodass eine hohe Übertragungsgeschwindigkeit von unter 10 ms bereits realisiert und belegt wurde. Da UDP ein freies Protokoll ist und alle bei YXLON eingesetzten Automatisierungssoftwarehersteller Funktionen für die Implementierung bereitstellen,

müssen keine finanziellen Mittel durch etwaige anfallende Lizenzkosten bereit gestellt werden. Ein großer Nachteil, nämlich die fehlende Sicherheitsschicht, kann durch eigens programmierte Sicherheitsmechanismen weitestgehend ausgeglichen werden. Wie groß der Aufwand hierfür ist, soll getestet werden. Außerdem muss geprüft werden, ob UDP auch unter B&R-Soft- und Hardware die Kommunikationszeiten von unter 10 ms einhalten kann.

3.2.3 OPC

OLE for Process Control ist eine standardisierte Softwareschnittstelle, die den Datenaustausch zwischen Anwendungen unterschiedlicher Hersteller ermöglicht. Dieser Austausch wird durch die Verwendung von Microsofts DCOM-Technologie realisiert. Über eine TCP/IP-Verbindung greift der OPC-Client auf die vom OPC-Server bereitgestellten Daten zu, der Client holt sich die Prozessdaten über den verwendeten Feldbus der SPS und stellt sie dem Server als OPC-Objekte zur Verfügung. OPC stellt nicht nur ein Transportprotokoll, sondern stellt noch andere Spezifikationen für die Arbeit in der Prozessautomatisierung zur Verfügung. OPC kann somit für Datenarchivierung, Alarm-Meldungen, Steuerung und Überwachung verwendet werden.
Die Verwendung von OPC bringt folgende Vor- und Nachteile mit sich:

Vorteile:

- Geringer Implementierungsaufwand
- Automatische Fehlererkennung
- Verschlüsselung der Daten möglich
- Daten während Verbindungsunterbrechungen zwischenspeicherbar

Nachteile:

- Hohe Kosten durch Lizenzgebühren und Mitgliedschaft in der OPC Foundation
- Langsame Kommunikation (>50 ms) [4]
- Keine konfigurierbaren Timeouts
- Bindung an das Betriebssystem Windows
- Keinen Einblick in Quellcode

OPC erfüllt die Anforderung an die Schnittstelle, plattformunabhängig zu sein und würde sich demnach sehr gut als Standardinterface eignen. Allerdings zeichnet sich OPC nicht durch eine hohe Performance aus. Eine Kommunikationsdauer von über 50 ms ist weit von der

Anforderung, kleiner als 10 ms zu kommunizieren, entfernt. Somit wird das wichtigste Kriterium nicht erfüllt. Außerdem wird OPC zum aktuellen Zeitpunkt im Hause YXLON nicht verwendet, sodass eine Einführung einer komplett neuen und unbekannten Schnittstelle als Standardinterface nicht wirtschaftlich wäre. Alle Angestellten, die mit der Schnittstelle arbeiten, müssten geschult werden, was mit hohen Kosten verbunden ist. Weitere Kosten entstehen durch Lizenzgebühren und den jährlich zu zahlenden Mitgliedsbeitrag an die OPC Foundation.

3.2.4 Dual Ported Memory

Als Dual Ported Memory wird der Zugriff zweier Prozesse auf einen gemeinsamen Speicher beschrieben. Der gemeinsame Speicher wird realisiert, indem eine sogenannte Slot-SPS in Form einer Einsteckkarte in den PC integriert wird. Die SPS kann dennoch unabhängig vom PC arbeiten, da sie eine eigene CPU besitzt. Da somit keine Datenübertragung, sondern lediglich ein Datenzugriff erfolgt, zeichnet sich die speicherbasierte Kommunikation durch ihre hohe Performance aus.
Folgende Vor- und Nachteile ergeben sich bei der Verwendung des Prozess Variablen Interfaces:

Vorteile:

- Sehr schnelle Kommunikation (< 5 ms)

Nachteile:

- Sehr hohe Kosten durch teure Hardware
- Prozesse müssen gegenseitige Speicherzugriffe selbst ausschließen
- Keine Allgemeingültigkeit da hardwareabhängig
- Mit B&R-Software nicht kompatibel

Das Prinzip des gemeinsam genutzten Speichers wird bei einigen Anlagen der YXLON International GmbH schon verwendet und hat sich dort durch eine äußerst geringe Kommunikationszeit bewährt, dennoch kommt dieses Prinzip weder für das Projekt Asterix noch als Standardinterface in Frage. Die hohen Kosten sind für beide Projekte inakzeptabel. Bisher wurde in der Firma YXLON die Slot-SPS der Firma Siemens für die Realisierung des Prinzips eingesetzt, allerdings hat Siemens das Produkt abgekündigt, sodass ein Standard auf diese Schnittstelle nicht nachhaltig wäre. B&R bietet Produkte, die Dual Ported Memory realisieren, nicht an. Außerdem steht die Wahl der Hardwarekomponenten, also die SPS,

schon fest, sodass eine Schnittstelle gefunden werden muss, die mit der gegebenen Hardware arbeiten kann.

3.2.5 PVI

Das Prozess Variablen Interface des Automatisierungsherstellers B&R ist ein Kommunikationstreiber zum Datenaustausch zwischen PC und Steuerung. Der genaue Aufbau des Protokolls kann nicht beschrieben werden, da der Quellcode dazu nicht öffentlich zugänglich ist. PVI arbeitet aber ähnlich wie TCP mit Sicherungsmechanismen um eine zuverlässige Kommunikation sicherstellen zu können. Der PVI Manager ist die zentrale Komponente des PVI. Er ist für die Verwaltung von Prozessdaten, Listen und Programm- oder Datenobjekten zuständig und erledigt sowohl die zeitliche als auch die richtungsorientierte Organisation der Prozessdaten.
Zusammenfassend kristallisieren sich folgende Vor- und Nachteile heraus:

Vorteile:

- geringer Implementierungsaufwand bei Verwendung von B&R-Steuerungen

Nachteile:

- keine Einsicht in Quellcode
- nur einsetzbar auf B&R-Steuerungen, somit plattformabhängig
- keine Kenntnisse über Übertragungsgeschwindigkeit
- nur unter Windows einsetzbar

PVI schneidet bei Gegenüberstellung der Vor- und Nachteile als Schnittstelle zwar nicht sehr gut ab, dennoch soll PVI in den nachfolgenden Test mit einbezogen werden. Da die Nutzung der B&R Hardware sowie die dazugehörige Software, das Automation Studio, fest gesetzt ist, ist es sinnvoll, sich auch mit dem Übertragungssystem auseinanderzusetzen. Durch den erstmaligen Einsatz von B&R-Produkten in der Firma YXLON International GmbH sowie den nicht öffentlichen Quellcode und die geringe Dokumentation über das Prozess Variablen Interface herrschen wenig Kenntnisse über das reale Verhalten der Komponenten. Da die schnelle Übertragungszeit das wichtigste Kriterium bei der Wahl der Schnittstelle ist und über diese keine Richtwerte von B&R existieren, sollen diese überprüft werden. Zwar kommt PVI als Standardschnittstelle aufgrund der Inkompatibilität mit Produkten anderer Automatisierungshersteller nicht in Frage; sollte sich PVI aber durch eine extrem hohe

Übertragungsgeschwindigkeit und Zuverlässigkeit auszeichnen, muss überlegt werden, ob das Interface nicht dennoch als Schnittstelle für das Projekt Asterix genutzt werden kann.

3.3 Ergebnis der Analyse

Als Fazit aus den Punkten 3.2.1 bis 3.2.5 folgt, dass sich OPC und Dual Ported Memory weder als Schnittstelle für das Projekt Asterix noch als Standardinterface eignen. Sie müssen also nicht weiter untersucht werden und werden im Folgenden nicht mehr behandelt. Das größte Augenmerk muss auf TCP und UDP gelegt werden, da sich die beiden durch ihre Eigenschaften besonders gut eignen würden. PVI muss kritisch betrachtet werden, da es als Standardschnittstelle zwar nicht in Betracht kommt. Es soll aber trotzdem näher untersucht werden, da es durch seinen engen Bezug zu den verwendeten B&R-Produkten wahrscheinlich ist, dass diese untereinander gut zusammenwirken und so eine gute Performance erreicht werden kann.

4 Realisierung und Tests

Um das Verhalten und die Eigenschaften der Transportprotokolle ermitteln und bewerten zu können, wurden diese unter gleichen Bedingungen und mit denselben Verfahren getestet. Hierzu wurde ein Testaufbau mit einem PC und einer Steuerung aufgebaut. Bei der Steuerung handelt es sich um eine X20CP1485-1 der Firma B&R (Datenblatt im Anhang), die in der Anlage Asterix verwendet werden soll. Außerdem wurden ein DELL Latitude D810 Laptop (ein Prozessor mit 2.26GHz) und ein DELL Precision M65 (zwei Prozessoren mit 2.33GHz) verwendet. Die Entscheidung die Tests mit einem zweiten PC durchzuführen, fiel auf die Fragestellung zurück inwieweit die Kommunikationszeit zwischen Rechner und Steuerung hardwareabhängig ist.

Die verwendete Steuerung verfügt über zwei Schnittstellen. Eine POWERLINK V1/V2 sowie eine Ethernet 10/100 Base-T Schnittstelle. Letztere wurde für die Realisierung des Testaufbaus gewählt. Um eine Onlineverbindung zwischen PC und Steuerung herzustellen, muss der Steuerung bei der Hardwarekonfiguration im Automation Studio eine IP-Adresse sowie eine Subdomain zugewiesen werden. Es wurden die Adressen 192.169.0.2 und 255.255.255.0 gewählt. Außerdem muss eingestellt werden, mit welcher IP-Adresse eine Verbindung hergestellt werden soll. Hierfür wurde dem Ethernetport des Rechners ebenfalls eine feste IP-Adresse und Subdomain zugewiesen. Diese wurden auf die Adressen 192.168.0.1 und 255.255.255.0 festgelegt. Sobald die Verbindung mittels Ethernet

hardwaretechnisch realisiert wurde, erfolgt die Verbindung zwischen den zwei Komponenten mittels der verwendeten B&R Steuerungssoftware Automation Studio automatisch. Verifiziert werden kann der Onlinestatus entweder durch einen Blick auf die Symbolleiste im Automation Studio, die den Verbindungsstatus permanent anzeigt und bei einer erfolgreichen Verbindung ihren Status von OFFLINE nach RUN wechselt oder die Steuerung wird vom Rechner aus „angepingt“. Mittels des Windows Kommandozeileninterpreters cmd.exe kann hierbei die IP-Adresse, mit der eine Verbindung hergestellt werden soll, eingegeben werden. Der Kommandozeileninterpreter ermittelt dann den aktuellen Verbindungsstatus und gibt bei Erfolg sowie bei Misserfolg eine Meldung über das Fenster zurück.

Um die Kommunikationszeit zwischen PC und Steuerung zu ermitteln, wurde ein zyklisches Programm mit dem Namen „test_pvi“ bzw. „test_udp“ und „test_tcp“ erstellt, welches in einer einstellbaren Zeit die Zyklusdurchläufe ermittelt. Hierfür wird eine Variable mit dem Namen „i32_read_by_pc“ an den PC gesendet, der diese ausliest und in eine Variable mit dem Namen „i32_write_to_plc“ schreibt und an die Steuerung zurückschickt. i32_read_by_pc wird in jedem Zyklusdurchlauf von der Steuerung so lange hochgezählt bis der Timer abgelaufen ist; ihr Wert wird dann in die Hilfsvariable „i32_readout“ geschrieben. Dieser Wert dividiert durch den Wert des Timers ergibt dann die Kommunikationszeit in Sekunden.

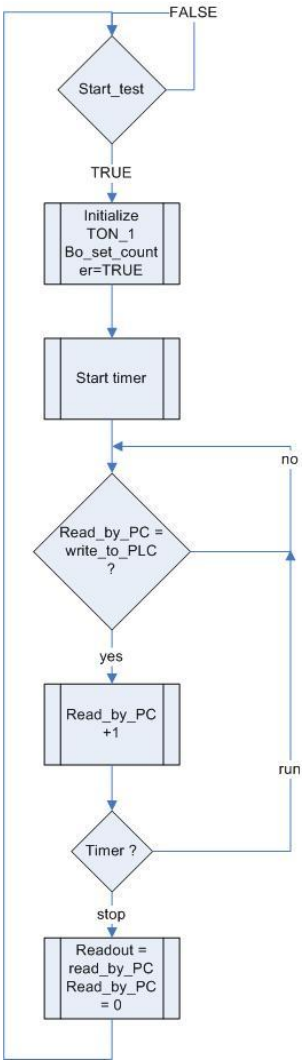

Abbildung 15 Flussdiagramm Testprogramm Quelle: eigene Darstellung

Um das Verhalten der Kommunikationszeit auch unter Belastung zu testen wurde ein zusätzliches Programm „load“ erstellt, welches sowohl die CPU des Rechners, als auch die CPU der Steuerung künstlich belasten soll. In diesem Programm werden Berechnungen mit Variablen durchgeführt. Einfache Rechenoperationen wie Addition, Subtraktion, Division und Multiplikation werden ebenso durchgeführt wie anspruchsvollere Operationen wie die Ermittlung von Sinuswerten oder Betragsberechnungen.

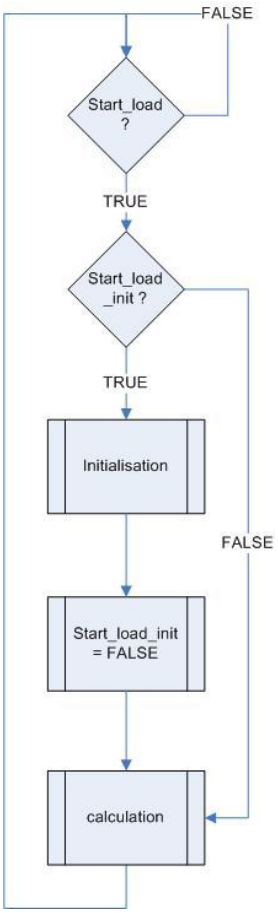

Abbildung 16 Flussdiagramm Lastprogramm Quelle: eigene Darstellung

Die Tests wurden mit einer Timerlänge von 1 s, 10 s und 30 s durchgeführt. Hieran lässt sich feststellen wie konstant sich die Kommunikationsdauer in Abhängigkeit des Timers verhält.

Die Programme wurden den höchstpriorisierten Zeitscheiben zugeordnet. Die Standardeinstellung des Automation Studios gibt im Cycle #1 eine Laufzeit von minimal 1 ms vor. Durch Ändern des Systemticks von 1 ms auf 400 µs ist es aber möglich, den ersten Zyklus mit einer Laufzeit von 400 µs einzustellen. Alle anderen Zyklen müssen einen Wert, der ein Vielfaches des Systemticks beträgt, annehmen.

4.1 PVI Test auf DELL Latitude D810

Für den Test des Prozess Variablen Interfaces ist es nicht nötig, dieses in ein eigenes Programm zu implementieren. Es ist bereits in das Automation Studio integriert. Allerdings muss für die Gegenseite der Steuerung, also die Rechnersoftware ein passendes PlugIn geschrieben werden, welches gewährleistet, dass die Daten auch tatsächlich übertragen werden. Die Übertragung der Daten erfolgt im Automation Studio selbständig über PVI.

Bei der Durchführung des ersten Tests mit PVI auf einem Dell Latitude D810 waren das Runtime GUI und der Datapoint Browser der C6k Software auf dem PC geöffnet, ebenso das Automation Studio um mittels der Watch-Funktion die Zyklusdurchläufe beobachten zu können und die relevanten Werte zu dokumentieren. Dabei war zu beobachten, dass die CPU des Rechners zu 100 Prozent ausgelastet war, sowohl bei dem Test mit Belastung als auch ohne. Da die Kommunikationszeit bei Zuschalten der Last um durchschnittlich 6 ms anstieg, wurden alle weiteren Tests ohne die Runtime GUI durchgeführt, um ihren Einfluss auf die CPU-Auslastung zu verringern. Außerdem wurden im Datapoint Browser der C6k Software die Lastvariablen ausgeblendet. Dies führte zu einer erheblich geringeren CPU-Auslastung des Rechners.

Das Programm „test_pvi“ wurde dem Cycle #1 mit einer Dauer von 400 µs zugeordnet. Das Programm „load“ wurde dem Cycle #2 mit einer Dauer von 1,2 ms zugeordnet. Die Last war zwar während allen Tests einer Zeitscheibe zugewiesen, wurde aber nur ausgeführt sobald eine Aktivierungsvariable mit dem Namen „start_load“ im Programm „test_pvi“ auf TRUE gesetzt wurde.

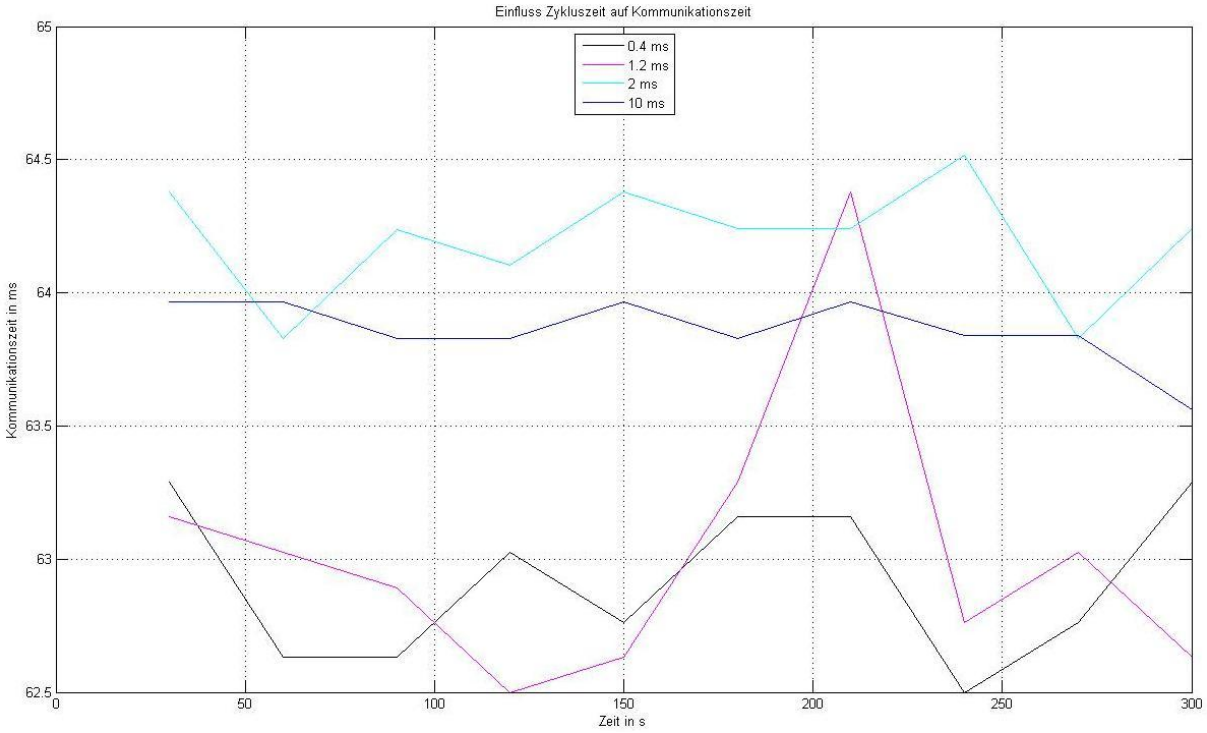

Abbildung 17 Einfluss der Zykluszeit auf Kommunikationszeit mit PVI auf DELL Latitude ohne Last

Wie auf dem obigen Diagramm zu sehen ist, steigt die Kommunikationszeit nicht gleichmäßig mit Erhöhung der Zyklusabarbeitungszeit an. Die Messreihe, die mit einer Dauer von 2 ms abgearbeitet wurde, braucht für die Übertragung der Datenpakete länger, als wenn

10 ms eingestellt wurden. Dennoch benötigt der Zyklus mit 0.4 ms am wenigsten Zeit für den Datentransport. Die Werte der Messreihe mit 10 ms zeigen sich am stabilsten, sie schwanken lediglich um 0.1361 ms, wobei die anderen Messreihen bis zu 1.877 ms schwanken. Ein genereller regelmäßiger Zusammenhang zwischen Zykluszeit und Kommunikationszeit scheint nicht zu bestehen.

Werden die verwendeten Timer im Hinblick auf die Stabilität der aufgenommenen Werte verglichen, würde theoretisch erwartet werden, dass ein Timer, je länger er ist, genauere und somit auch stabilere Werte liefern kann. In der Praxis wird dies bestätigt. In der unten dargestellten Grafik wurden die Timer anhand der Kommunikationszeit dargestellt. Für den Timer mit einer Dauer von 30 s ergibt sich tatsächlich der stabilste Verlauf. Dies liegt daran, dass mehr Zeit zur Verfügung steht um die Datentransporte zu zählen. Je höher diese Zahl wird, desto genauer wird auch die Kommunikationszeit berechnet.
Die anderen beiden Timer verhalten sich ebenso wie in der Theorie erwartet. Beide zeigen große Instabilität auf, wobei der Timer mit einer Länge von 10 s geringere Schwankungen (bis zu 8.422 ms, das entspricht 10.78 Prozent Abweichung) aufzeigt als der Timer mit einer Dauer von 1 s (8.9286 ms, das entspricht 12.5 Prozent).

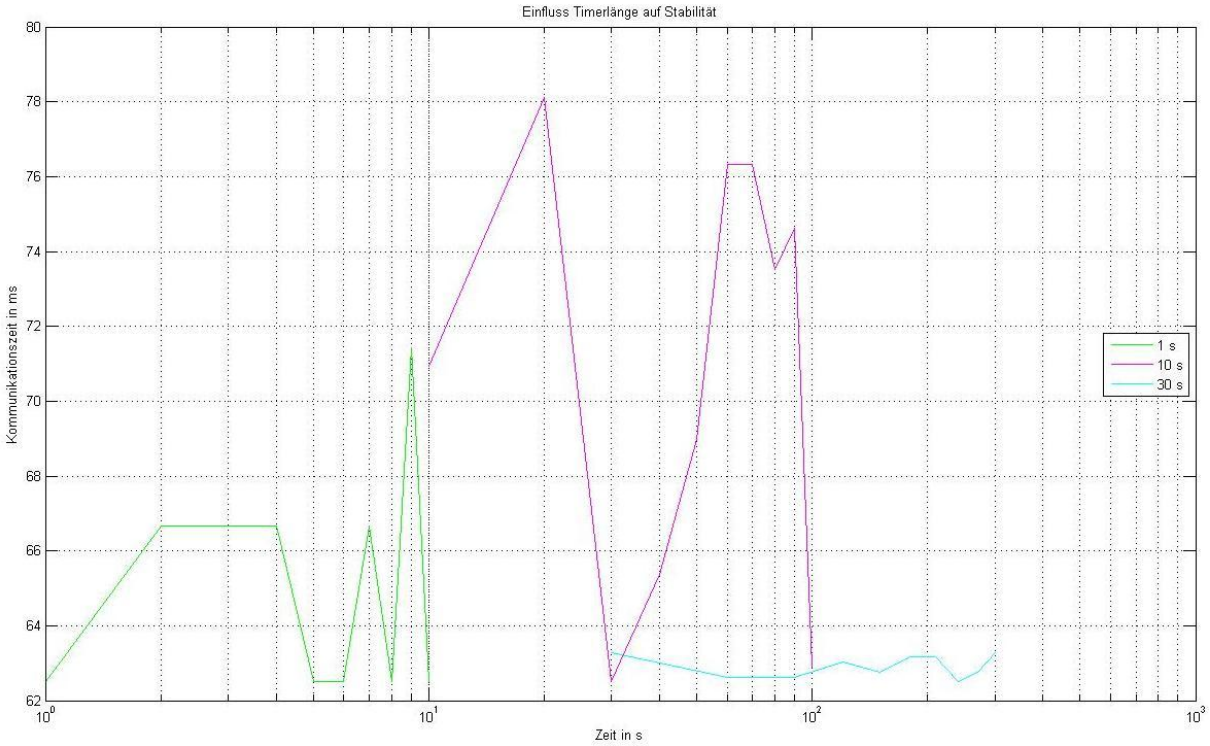

Abbildung 18 Einfluss der Timerlänge auf Stabilität mit PVI auf DELL Latitude ohne Last

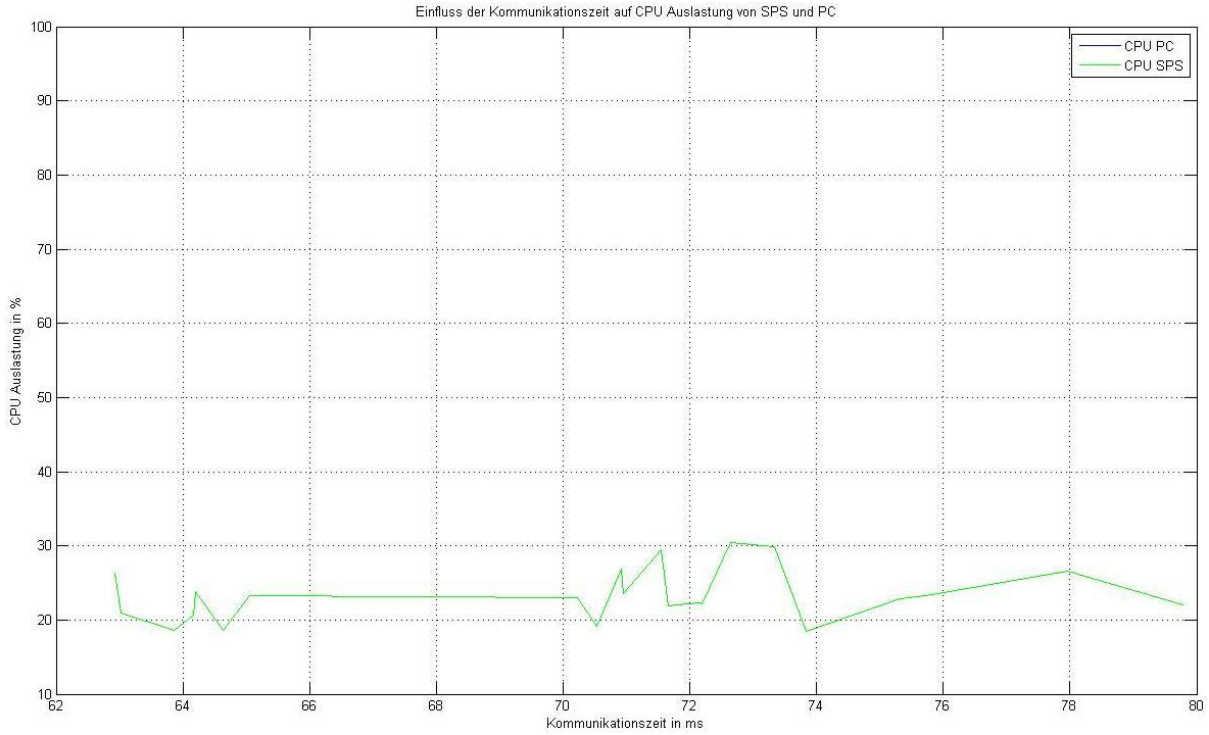

Abbildung 19 Einfluss der Kommunikationszeit auf CPU Auslastung mit PVI auf DELL Latitude

Bei Auswertung der CPU-Auslastung von Steuerung und PC ist kein regelmäßiges Verhalten zu erkennen. Die Prozessoren des DELL Latitude D810 sind voll unter Last, denn der Taskmanager des Rechners gibt eine permanente CPU-Auslastung von 100 Prozent an. Da nicht bekannt ist, ob die CPU des Rechners die Datenübertragungsgeschwindigkeit hemmt, wurden alle Tests, die in diesem Kapitel beschrieben werden, noch einmal mit einem anderen Rechner mit leistungsfähigeren Prozessoren durchgeführt.

Der Celeron 400 Prozessor der SPS ist nicht voll ausgelastet, die gesamte CPU-Auslastung beträgt nur 30 Prozent. Dennoch verläuft der Graf auf der obigen Darstellung sehr unregelmäßig. Ein umgekehrt rampenförmiges Verhalten, wie es in der Grafik zwischen 74 ms und 78 ms zu sehen ist, wurde theoretisch erwartet. Der sprunghafte Verlauf des Grafen zeigt keine Regelmäßigkeit auf. Ein Zusammenhang zwischen Auslastung und Kommunikationszeit lässt sich demnach nicht feststellen.

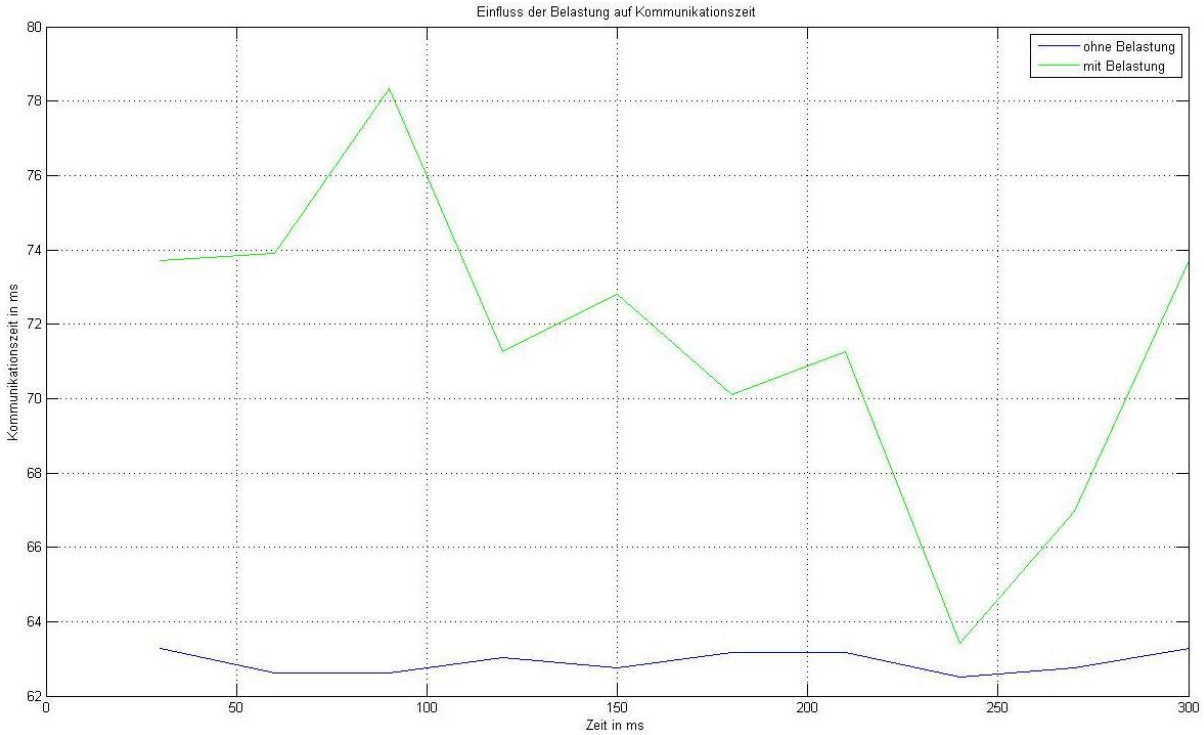

Abbildung 20 Einfluss der Last auf Kommunikationszeit mit PVI mit DELL Latitude

Die Last hat einen sehr großen Einfluss auf die Kommunikationszeit. Im obigen Diagramm ist zu erkennen, dass bei Zuschalten der Last keine Stabilität mehr gegeben ist. Die Werte der Übertragungsdauer schwanken um durchschnittlich 16.2432 ms, was 20.46 Prozent entsprechen. Es lässt sich kein gleichmäßiges Verhalten der Kommunikationszeit feststellen. Ohne Last schwanken die Werte lediglich um 0.7911 ms, was ca. 1.245 Prozent entsprechen. Das Prozess Variablen Interface zeichnet sich demnach weder durch eine gute Performance aus (die durchschnittliche Datenübertragungszeit liegt deutlich über den geforderten 10 ms) noch durch eine hohe Stabilität.

4.2 PVI Test auf DELL Precision M65

Da beim PVI Test auf dem DELL Latitude D810 aufgefallen ist, dass die CPU-Belastung des Rechners permanent zu 100 Prozent ausgelastet war, fiel die Frage auf, ob die Kommunikationszeit davon negativ beeinflusst wird. Der verwendete Rechner hat lediglich einen Intel(R) Pentium(R) M Prozessor mit 2.26 GHz. Um diese Fragestellung beantworten zu können wurde der Test erneut auf einem anderen Rechner mit leistungsstärkeren Prozessoren durchgeführt (Intel(R) Core(TM) 2 CPU T760 mit 2.33GHz).

Das Vorgehen war dabei dasselbe, d.h. die Tests wurden mit den gleichen Einstellungen realisiert.

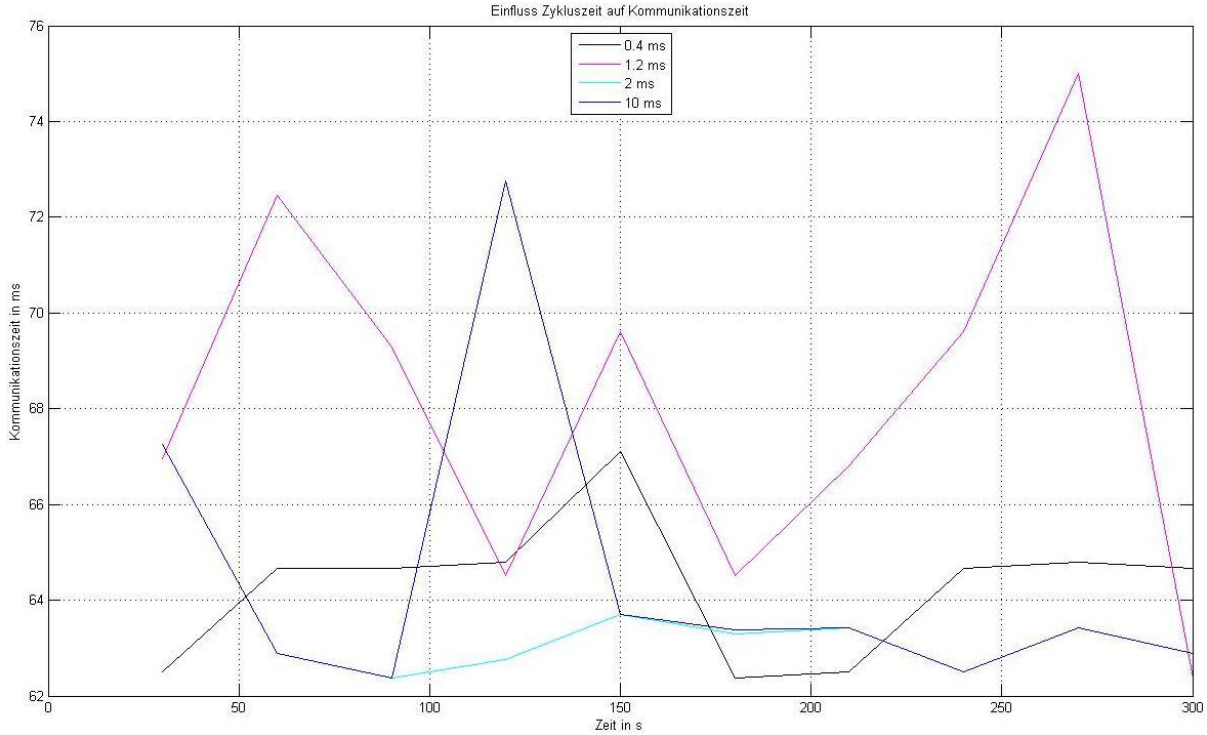

Abbildung 21 Einfluss der Zykluszeit auf Kommunikationszeit bei PVI mit DELL Precision ohne Last

Die Länge der Zyklen scheint überhaupt keinen Einfluss auf die Kommunikationszeit zu haben. Aus der obigen Abbildung ist zu erkennen, dass die Graphen keinen konstanten Verlauf annehmen. Zu erwarten wäre eine Zunahme der Kommunikationszeit je länger der Zyklus gewählt wurde. Doch die Werte springen ohne Regelmäßigkeit von Punkt zu Punkt. Auch die großen Schwankungen innerhalb der einzelnen Messreihen von bis zu 12.6 ms sind ein sehr untypisches Verhalten für ein Transportprotokoll.

Auch bei Bewertung des Einflusses des Timers auf die Übertragungszeit fällt auf, dass die Werte sehr unstetig sind. Der Timer mit einer Länge von 1 s ist der stabilste, er springt lediglich am Anfang um 4.17 ms, bleibt dann aber konstant. Der Timer mit einer Dauer von 30 s schwankt um 5.29 ms, damit bleibt er aber deutlich unter den 9.78ms Differenz, die der Timer von 10 s Länge mit sich bringt. Dort ist auch die mittlere Kommunikationszeit mit 71.7610 ms deutlich am höchsten.

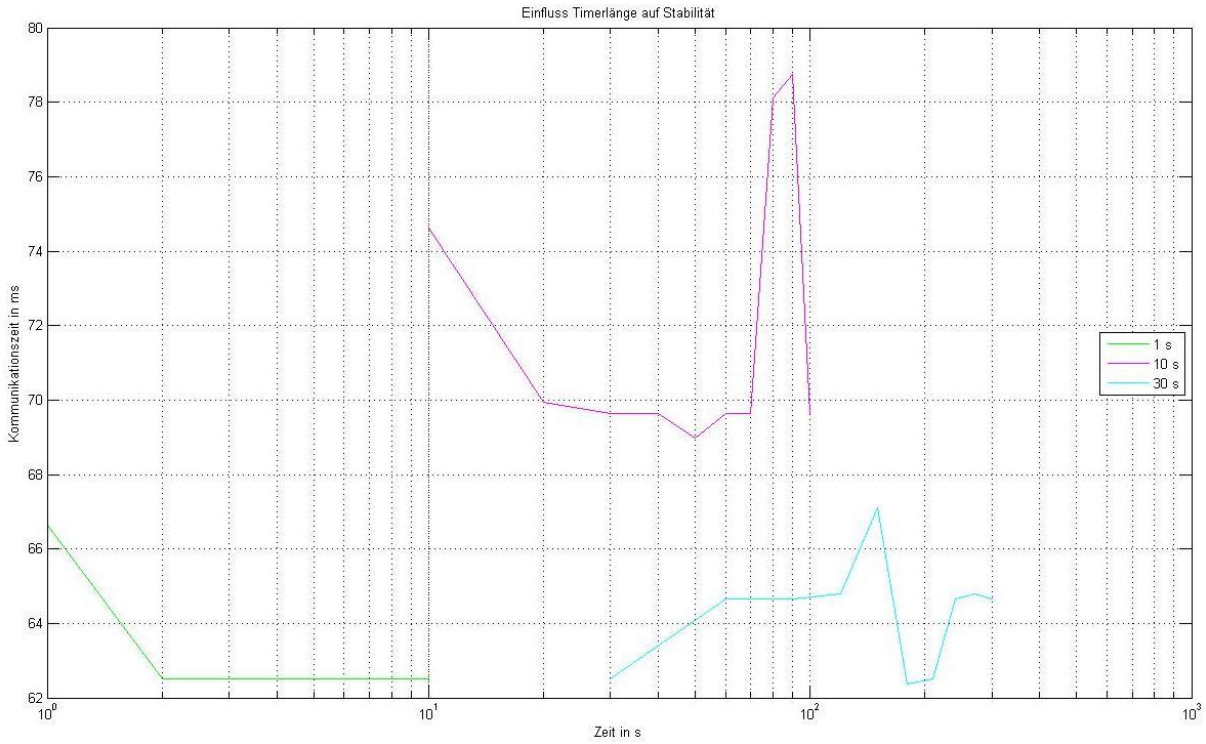

Abbildung 22 Einfluss des Timers auf Stabilität bei PVI mit DELL Precision ohne Last

Bei der Auswertung der CPU-Auslastung von Rechner und Steuerung fällt auf, dass die Prozessoren des PCs kaum ausgelastet sind. War die CPU beim Test mit dem DELL Latitude noch zu 100 Prozent ausgelastet, liegt sie hier bei durchschnittlich 5.5 Prozent. Die Auslastung der SPS hat sich im Vergleich zum vorhergegangenen Test nicht verändert. Auch in diesem schwankt sie zwischen 20 Prozent und 30 Prozent. Die Kurvenverläufe der unten dargestellten Abbildung zeigen dennoch, dass auch hier wider Erwarten die Kommunikationszeit nicht rampenförmig mit der CPU-Auslastung steigt. Da die maximal benötigte Übertragungsdauer hier bei nur 71.761 ms liegt, ist jedoch davon auszugehen, dass die Rechner-CPU einen nicht geringen Einfluss auf die Kommunikationszeit hat. So lag die maximale Übertragungsdauer bei einer Rechner-CPU-Auslastung von 100 Prozent auf dem DELL Latitude bei 79.7869 ms, was einer Differenz von 8.0259 ms entspricht. Die minimal erreichte Kommunikationszeit liegt bei beiden Tests auf dem gleichen Wert, nämlich 62.5 ms.

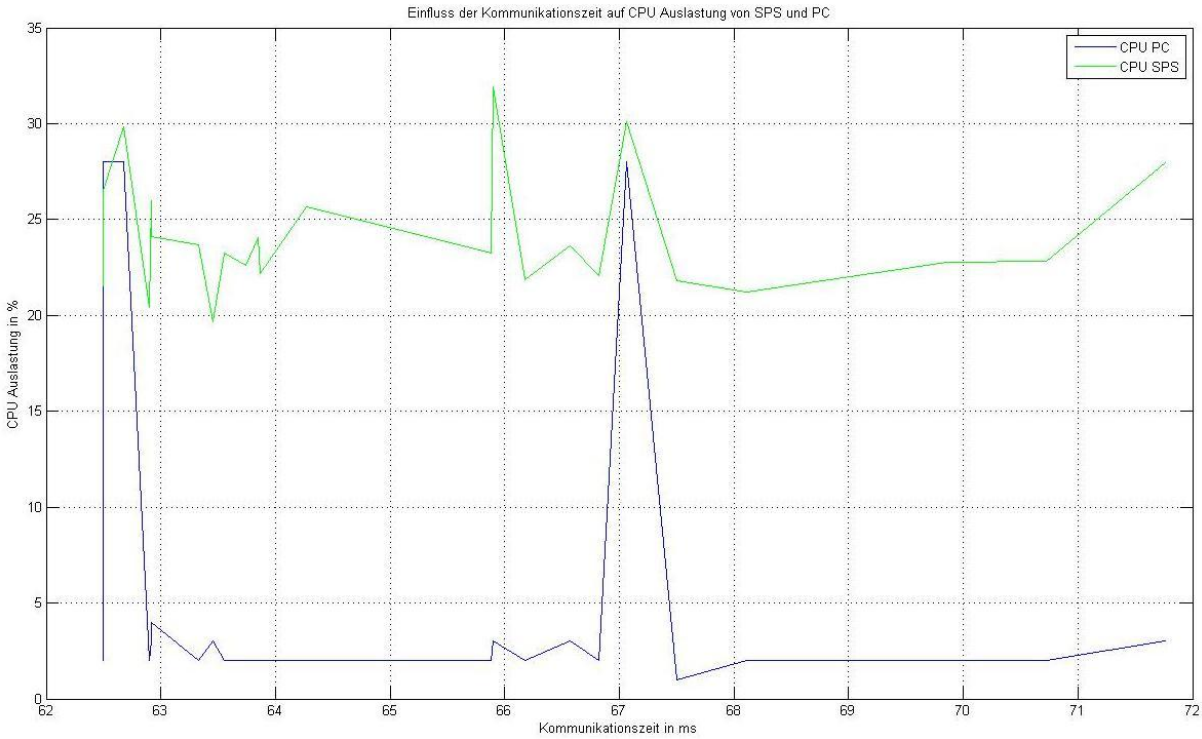

Abbildung 23 Einfluss der Kommunikationszeit auf CPU-Auslastung bei PVI mit DELL Precision

Die unten dargestellte Grafik zeigt den Einfluss der Last auf die Kommunikationszeit auf. Die Zeit, die ein Datenpaket benötigt, um von der SPS zum Rechner und wieder zurück übertragen zu werden, ändert sich gegensätzlich als theoretisch erwartet. Vergleicht man die Mittelwerte der Messreihen, braucht die Kommunikation unter Last 1.5919 ms weniger als wenn sie nicht belastet wird. Dieses Verhalten ist nur mit der Instabilität des Prozess Variablen Interfaces zu erklären.

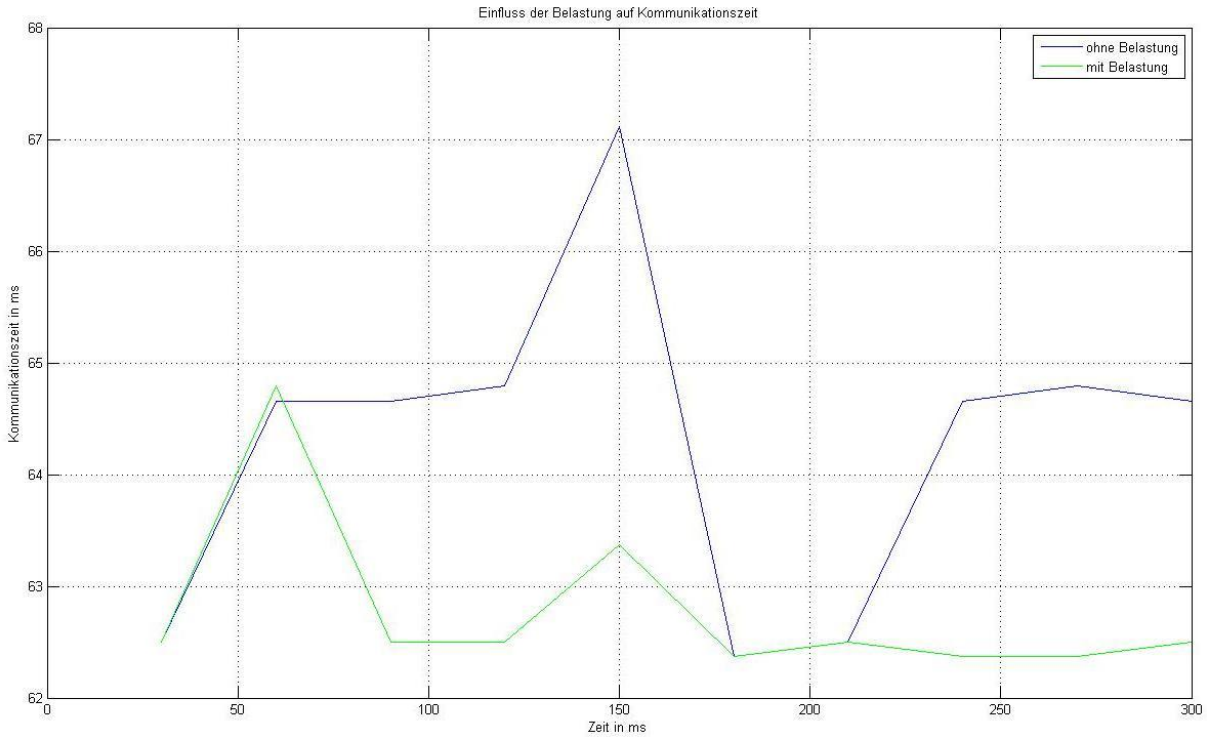

Abbildung 24 Einfluss der Last auf Kommunikationszeit mit PVI mit DELL Precision

4.3 UDP Test auf DELL Latitude D810

Da es sich, wie bereits erwähnt, bei dem User Datagram Protocol um ein freies Protokoll handelt, ist zwar das Übertragungsprinzip immer das gleiche, die tatsächliche Umsetzung erfolgt aber je nach verwendeter Software zu Software unterschiedlich. In diesem Fall wurde das Prinzip an die Entwicklungsumgebung des Automation Studios angepasst. B&R stellt Funktionen bereit um einen UDP-Port zu schließen, Daten zu empfangen und zu senden und den Port wieder zu schließen. Doch bevor die eigentliche Übertragung mittels Programmiercode realisiert wird, ist es nötig zu definieren, was übertragen werden soll. Hierfür muss noch einmal daran erinnert werden, dass UDP über keine eigenen Sicherheitsmechanismen für eine zuverlässige Übertragung verfügt. Es muss also ein eigener Header kreiert werden, der diese Aufgabe übernimmt.

Das UDP Transportprotokoll kann höchstens 1400 Byte übertragen, deshalb ist es für einen Performancetest sinnvoll auch Pakete in dieser Größe zu verschicken, also den worst-case anzunehmen.

Um den Test unter möglichst realen Anforderungen zu realisieren wurde als Datagramm eine Struktur gewählt. Es ist ebenfalls denkbar ein Array zu verschicken. Da dies in der realen Kommunikation aber nicht praktiziert wird, weil es nicht möglich ist ein Array aus verschiedenen Datentypen zu erstellen und somit sehr unflexibel ist, wurde eine Struktur verwendet. Dieser Struktur musste nun ein einfacher Sicherheitsmechanismus zugefügt

werden. Dieser soll gewährleisten, dass Rechner und Steuerung „über die gleichen Daten reden“. Hierfür wurde eine ID zugefügt, die von der Steuerung lediglich vom Empfangs- ins Sendedatagramm kopiert wird. Der Rechner jedoch schreibt und liest diese Identifikationsnummer. Zunächst schreibt er sie ins zu sendende Datenpaket, liest sie aus, wenn er Daten von der SPS empfängt und vergleicht sie mit der ursprünglich Versendeten. Immer wenn er ein Datenpaket versendet, zählt er die ID um Eins hoch. So wird gewährleistet, dass Sender und Empfänger immer über die gleichen Daten „sprechen“ und Daten nicht unbemerkt verloren gehen. Zwar wird damit nicht garantiert, dass Daten verloren gehen, dass dies nicht verborgen geschieht jedoch schon. Um zu verhindern, dass ein Datenpaket nicht vollständig übertragen wird, werden der Struktur ein Start- und ein Endwert mit festen Werten beigefügt. Diese werden sowohl von der SPS als auch vom Rechner bei jedem Datenerhalt auf Richtigkeit überprüft, ist dies nicht der Fall wird das Datenpaket mit einer Fehlernummer zurückgesendet, ohne dass die weiteren Daten bearbeitet werden. Zwar kann mit diesem Verfahren nicht vollständig ausgeschlossen werden, dass Daten verloren gehen, es gibt aber einen Anhaltspunkt dafür, wenn Start- oder Endwert nicht korrekt übertragen wurden.

Diese zwei implementierten Sicherheitsmechanismen schließen zwar noch nicht alle Sicherheitslücken des User Datagram Protocols, für die Testzwecke reichen sie jedoch aus. Außerdem wurde eine Variable vom Typ Integer zur Kommandoauswertung in die Struktur integriert. In diesem Testfall gibt es nur zwei Kommandos: eines um den Test ohne Belastung durchzuführen, in diesem Fall muss eine Eins in der Variable stehen und eines für den Test mit Belastung, dann muss eine Zwei empfangen werden. Steht in der Variable eine andere Zahl als Eins oder Zwei, wird eine entsprechende Fehlernummer an den PC gesendet, über den die Eingabe des Kommandos erfolgt. Der Header besteht also aus einem Start- und Endwert, wobei der Endwert konsequenterweise am Ende der Struktur steht, einer Identifikationsnummer, einer Fehlernummer und einem Kommando. Die Variablen „i32_read_by_PC“ und „i32_wirte_to_PLC“ werden wie schon im Test mit PVI ausgewertet und entsprechend behandelt. Da die Struktur mit den relevanten Werten keine Länge von 1400 Byte erreicht, ist eine künstliche Länge von 1304 Byte in Form einer Stringvariablen vom Typ Boolean nötig. Alle weiteren Variablen dienen nur der eigentlichen Übertragung und werden nicht weiter ausgewertet, sondern nur an den PC gesendet und dort angezeigt.

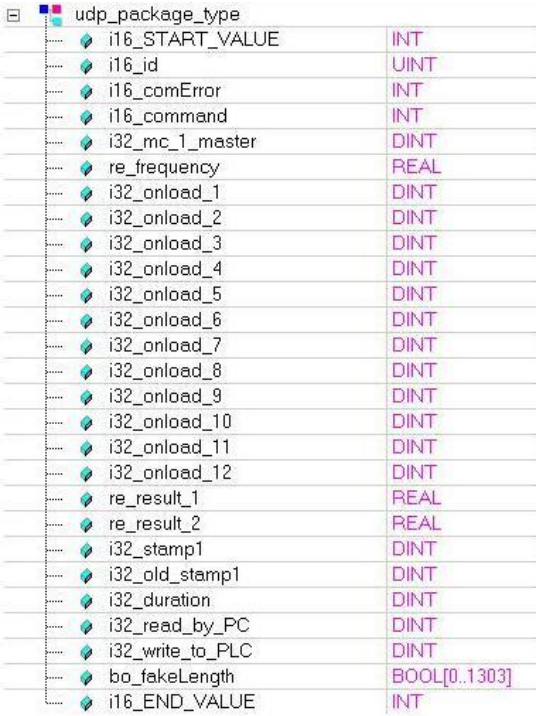

Abbildung 25 Aufbau der UDP-Struktur

Beim Erstellen der UDP-Struktur muss das Speichermanagement des Automation Studios berücksichtigt werden, da es sich durch Besonderheiten von herkömmlichen Speicherverfahren abhebt. Der kleinste zu verwendende Datentyp ist ein Boolean, abweichend von anderen Programmierumgebungen, benötigt dieser aber nicht nur ein Bit Speicherplatz, sondern ein ganzes Byte. Alle anderen Datentypen haben den gleichen Speicherplatzbedarf wie es aus anderer Software bekannt ist.

Datentyp	Speicherbedarf [Byte]	Wertebereich
BOOL	1	TRUE (=1), FALSE (=0) (z. B. Digitale Ein- und Ausgänge)
SINT	1	-128 ... +127
INT	2	-32 768 ... +32 767 (z. B. Analoge Ein- und Ausgänge)
DINT	4	-2 147 483 648 ... +2 147 483 647
USINT	1	0 ... 255
UINT	2	0 ... 65 535
UDINT	4	0 ... 4 294 967 295
REAL	4	-3.4E38 ... +3.4E38
TIME	4	T#-24d_20h_31m_23s_648ms ...T#24d_20h_31m_23s_647ms
DATE_AND_TIME	4	DT#1970-01-01-00:00:00 ... DT#2106-02-07-06:28:15
STRING	Variabel, aber mind. 2 Bytes	Darstellung von Zeichenketten

Abbildung 26 Speicherbedarf der Datentypen im Automation Studio

Die Besonderheit, die das Automation Studio mit sich bringt, ist die Tatsache, dass Datentypen nur auf einer Startadresse liegen dürfen, die durch ihre eigene Größe teilbar ist und die Summe der insgesamt verwendeten Bytes durch den größten verwendeten Speicherbedarf dividierbar ist. Wenn dies durch eine Anordnung in der Struktur nicht vom Programmierer selbst berücksichtigt wird, fügt das Automation Studio automatisch Leerbytes ein, so dass die Anforderung wieder erfüllt ist. Ist dem Programmierer dieses Speichermanagement nicht bewusst oder ignoriert er es, kann es sein, dass mehr Speicher benötigt wird, als ursprünglich dafür vorgesehen war.
In diesem Testfall ist es besonders wichtig auf die Speicherplatzanordnung zu achten, da die Software auf dem verwendeten PC auf die Daten zugreifen und diese auslesen bzw. schreiben muss. Sollten Rechner und Steuerung also nicht dieselbe Variablenanordnung verwenden, würde dies zu falschen Werten in den Variablen führen und das ganze Datenpaket wäre somit wertlos. Eine genaue Absprache zwischen PC- und SPS-Programmierer ist für die erfolgreiche Realisierung des UDP Protokolls also unausweichlich.

Nachdem definiert ist, welche Daten versendet werden sollen und welche Seite, nämlich PC oder SPS, diese liest oder schreibt, ist das eigentliche Übertragungskonzept zu erstellen. Wie bereits erwähnt, stellt das Automation Studio einige Funktionen zur Erstellung eines eigenen UDP-Transportmechanismus zur Verfügung. Die einzelnen Funktionen sollen hier nicht weiter erläutert werden, ihre Funktionsweise und Parametrierung ist dem Quellcode im Anhang zu entnehmen. Anzubringen sind jedoch die Funktionen UdpConnect() und UdpDisconnect(). Mit diesen Funktionsbausteinen kann ein UDP-Port mit einer Gegenstelle fest verbunden werden. Der UDP-Port kann dann nun mit dieser Gegenstelle kommunizieren, also senden und empfangen. Es ist daher nur sinnvoll einen Port zu verbinden, wenn dieser mit nur einer Gegenstelle Daten austauscht. Das Verbinden eines UDP-Ports bedeutet nicht, dass Handshakes zwischen beiden stattfinden und auch der Datenaustausch bleibt unkontrolliert. Nach B&R führt diese feste Verbindung zwischen den Ports allerdings zu einer Performancesteigerung von 30 Prozent. Da sich die neue Schnittstelle durch eine hohe Übertragungsgeschwindigkeit auszeichnen soll, wird die Übertragung sowohl mit fester Verbindung als auch ohne getestet um diese Versicherung seitens B&R zu bestätigen oder zu widerlegen.
Die drei Testprogramme “load”, “test_udp” und “own_udp_Connect” bzw. “own_udp_without_con” müssen den Zeitscheiben zugeordnet werden. Im Gegensatz zum Test mit PVI ist eine weitere Zeitscheibe von Nöten, da die UDP-Kommunikation ein eigenes Programm ist und nicht, wie PVI, in die Programmierumgebung integriert ist. Das UDP-Programm erhält die höchste Priorität, wird also der ersten Zeitscheibe zugeordnet. Diese ist

mit der geringsten einstellbaren zyklischen Abarbeitung von 0.4 ms eingestellt. Das eigentliche Testprogramm “test_udp“ wird in der Standardeinstellung mit 0.8 ms und die Last mit 1.2 ms abgearbeitet. Das Testprogramm wird auch hier mit verschiedenen Zykluszeiten getestet, um den Einfluss dieser auf die Kommunikationszeit bewerten zu können.
Da sich bei den Tests mit dem Prozess Variablen Interface ergeben hat, dass der DELL Latitude D810 mitunter zu 100 Prozent ausgelastet war, wurde das User Datagram Protocol lediglich auf dem DELL Precision M65 realisiert und getestet. Es ist somit zu erwarten, dass die Kommunikationszeit nicht von der Auslastung der Rechner CPU beeinflusst wird.

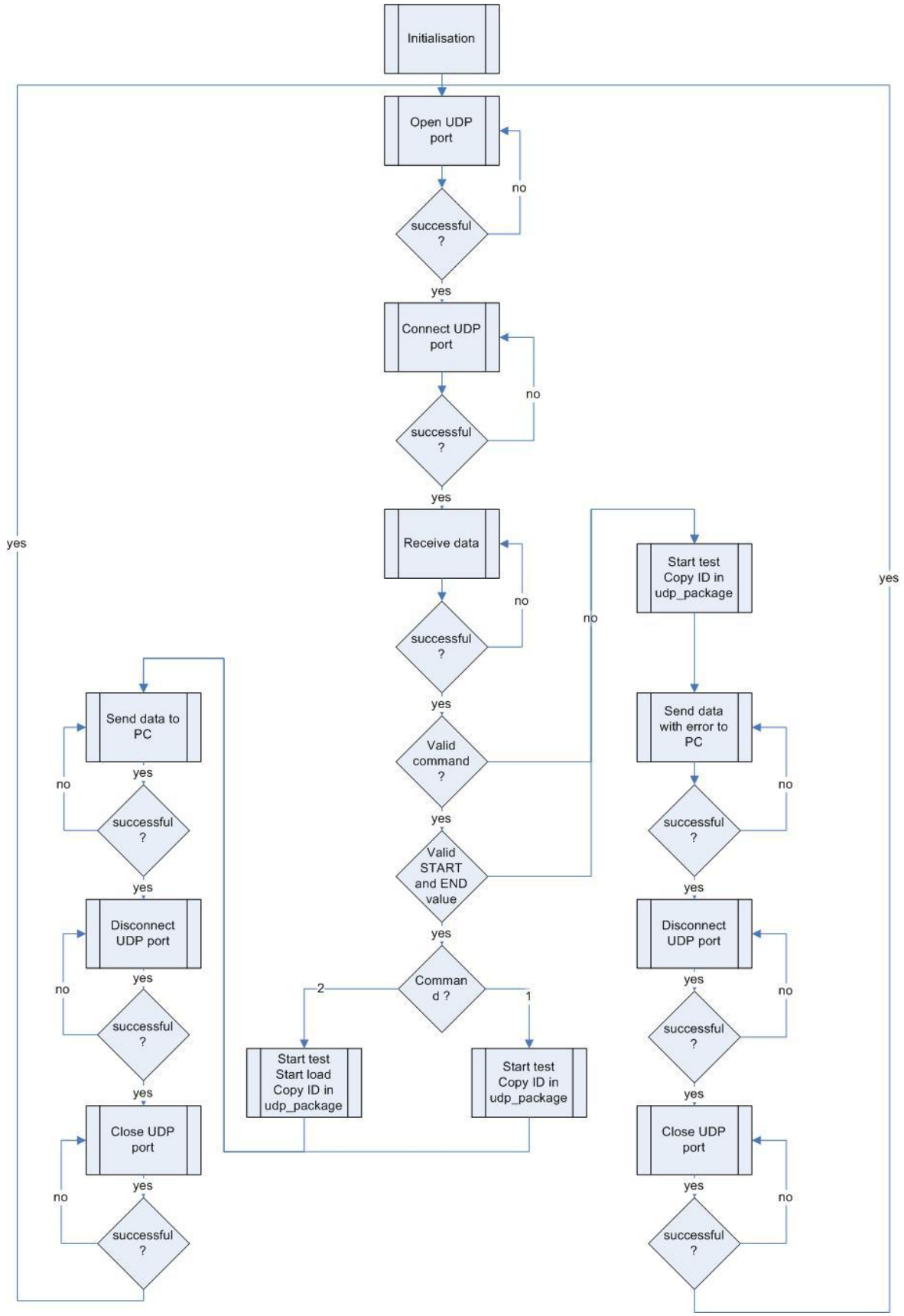

Abbildung 27 Flussdiagramm UDP mit fester Verbindung Quelle: eigene Darstellung

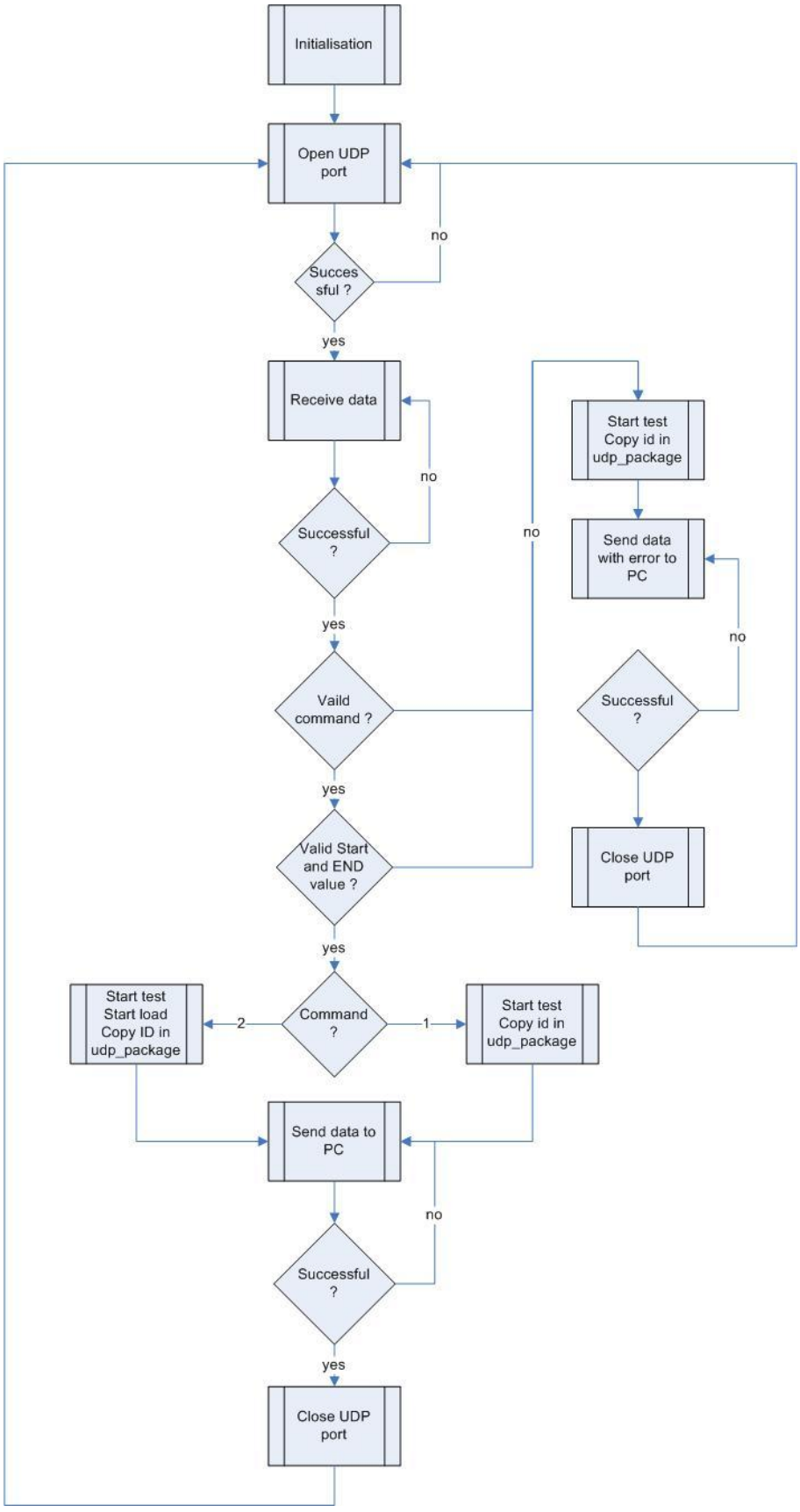

Abbildung 28 Flussdiagramm UDP ohne feste Verbindung Quelle: eigene Darstellung

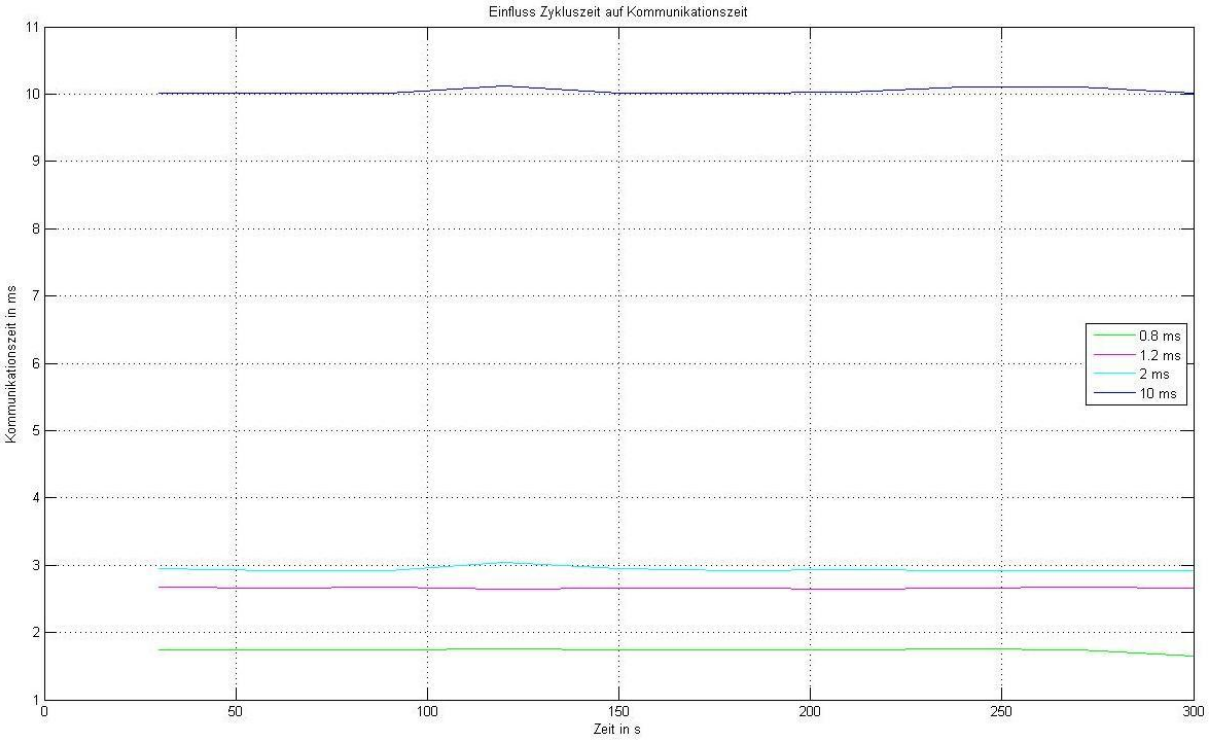

Abbildung 29 Einfluss Zykluszeit auf Kommunikationszeit bei fester UDP Verbindung auf DELL Precision ohne Last

Wie aus der Abbildung oben zu erkennen, ist der Einfluss der Zykluszeit auf die Kommunikationszeit bei UDP relativ groß. Relativ nur deshalb, weil die Kommunikation bei Zykluszeiten ≤ 3 ms dennoch bei 3 ms und damit deutlich unter den geforderten 10 ms liegt. Dass die Übertragungszeit bei einer Zyklusabarbeitungszeit von 10 ms deutlich ansteigt, hängt mit dem Abtasttheorem von Shannon zusammen, welches besagt, dass die Abtastfrequenz zweimal so hoch sein muss, wie die maximal auftretende Frequenz:

$$f_{Abtast} > 2 f_{max} \qquad [3b]$$

Da dies in der Messung nicht berücksichtigt wurde, ist das Ergebnis nicht zu werten.

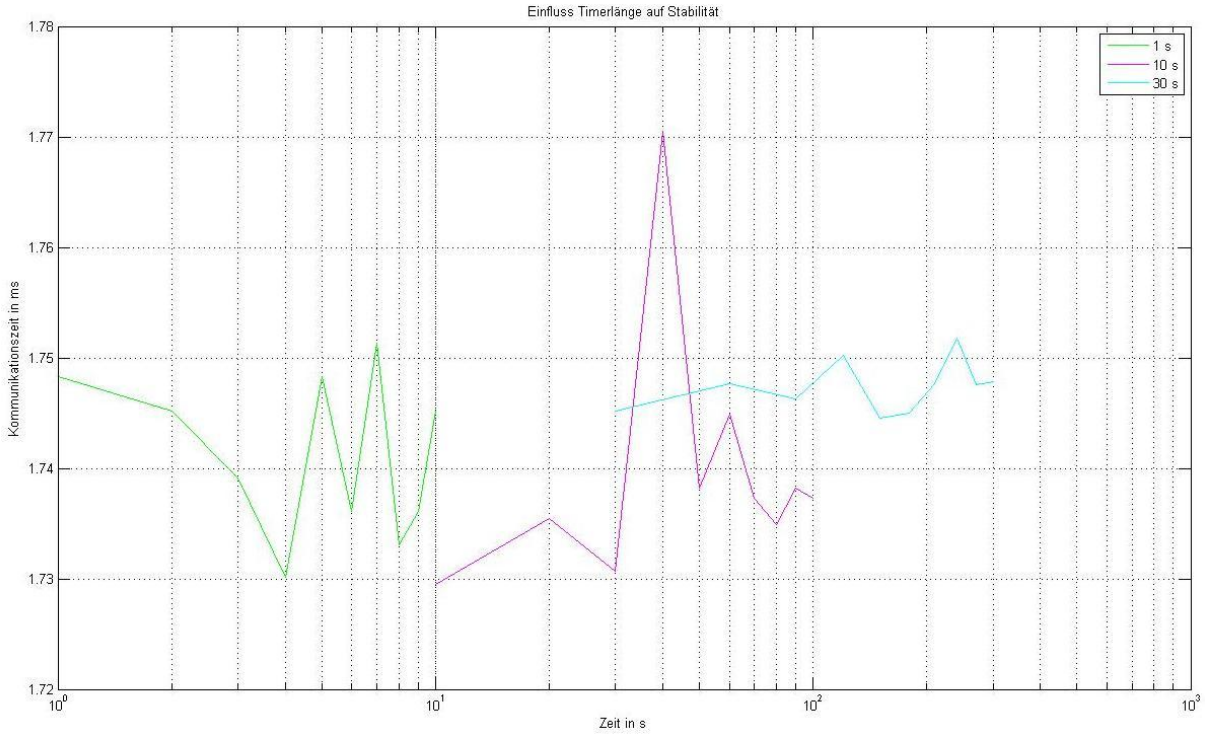

Abbildung 30 Einfluss Timerlänge auf Stabilität bei fester UDP Verbindung auf DELL Precision ohne Last

Wertet man die UDP-Kommunikation im Hinblick auf die Stabilität der Werte aus, fällt auf, dass die Messergebnisse lediglich im Mikrosekundenbereich schwanken. Der Timer mit einer Länge von 30 s zeigt sich allerdings am stabilsten. Dies liegt daran, dass er länger Zeit hat die Variable „i32_read_by_PC“ hochzuzählen und die Messung somit genauer wird. Dennoch zeichnet sich das User Datagram Protocol insgesamt durch sehr hohe Beständigkeit aus. Für die weiteren Testauswertungen wird im Folgenden meist der Timer mit einer Länge von 30 s verwendet, da sich dieser, wie bereits erläutert, als der Stabilste herausgestellt hat.

Würde man die CPU-Auslastung von Rechner und Steuerung in Abhängigkeit der Kommunikationszeit grafisch darstellen, wäre zu erwarten, dass sich eine Rampe ergeben würde. Theoretisch denkbar und sinnergebend würde diese abschüssig sein, denn es ist zu erwarten, dass eine schnelle Übertragung sowohl von PC als auch von SPS eine hohe Leistung fordert. Für den PC trifft dieses theoretisch überlegte Verhalten in keinster Weise zu. Die Auslastung bewegt sich zwischen 51 und 56 Prozent, allerdings ist überhaupt keine Regelmäßigkeit in der Verhaltensweise festzustellen. Dies kann an der ungenauen Messmethode liegen; die Werte wurden manuell vom Task Manager abgelesen, welche ohne Nachkommastellen dargestellt wurden.
Bei den Messergebnissen der CPU-Auslastung der Steuerung hingegen, ergibt sich für Kommunikationszeiten zwischen 3 ms und 9.5 ms ein rampenförmiges Verhalten, welches

mit den theoretisch erwarteten Werten übereinstimmt. Auffällig ist außerdem, dass die CPU der Steuerung zwischen 28 Prozent und 44 Prozent schwankt, was eine Differenz von 16 Prozent ergibt. Dies ist eine relativ große Schwankungsbreite, wobei beachtet werden muss, dass hierbei sowohl die Tests mit als auch ohne Belastung beinhaltet sind.

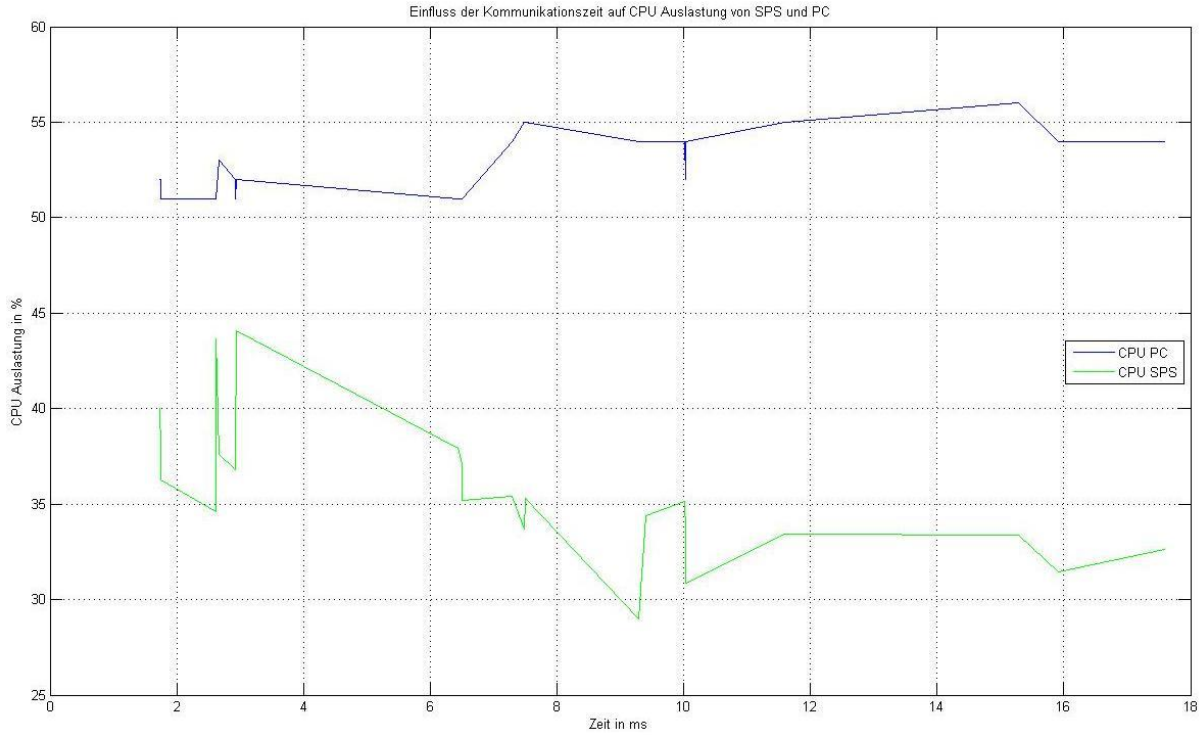

Abbildung 31 Einfluss CPU-Auslastung auf Kommunikationszeit bei fester UDP Verbindung auf DELL Precision

Werden die Messergebnisse der Übertragungszeit von Tests mit und ohne Belastung verglichen fällt auf, dass das User Datagram Protocol sehr anfällig für Belastungen ist. Zwar steigt die CPU-Auslastung der Steuerung unter Belastung im Schnitt nur um 5 Prozent an, dennoch steigt die Kommunikationszeit gegenüber dem Test ohne Belastung um bis zu 4.5 ms an. Positiv festzuhalten ist jedoch, dass die Übertragungszeit trotz Belastung deutlich unter 10 ms bleibt und sehr konstant verläuft.

Als Testreihe wurden hierfür eine Timerlänge von 30 s und eine Zykluszeit des Testprogramms von 0.8 ms gewählt, da sich diese als die Stabilste auszeichnet.

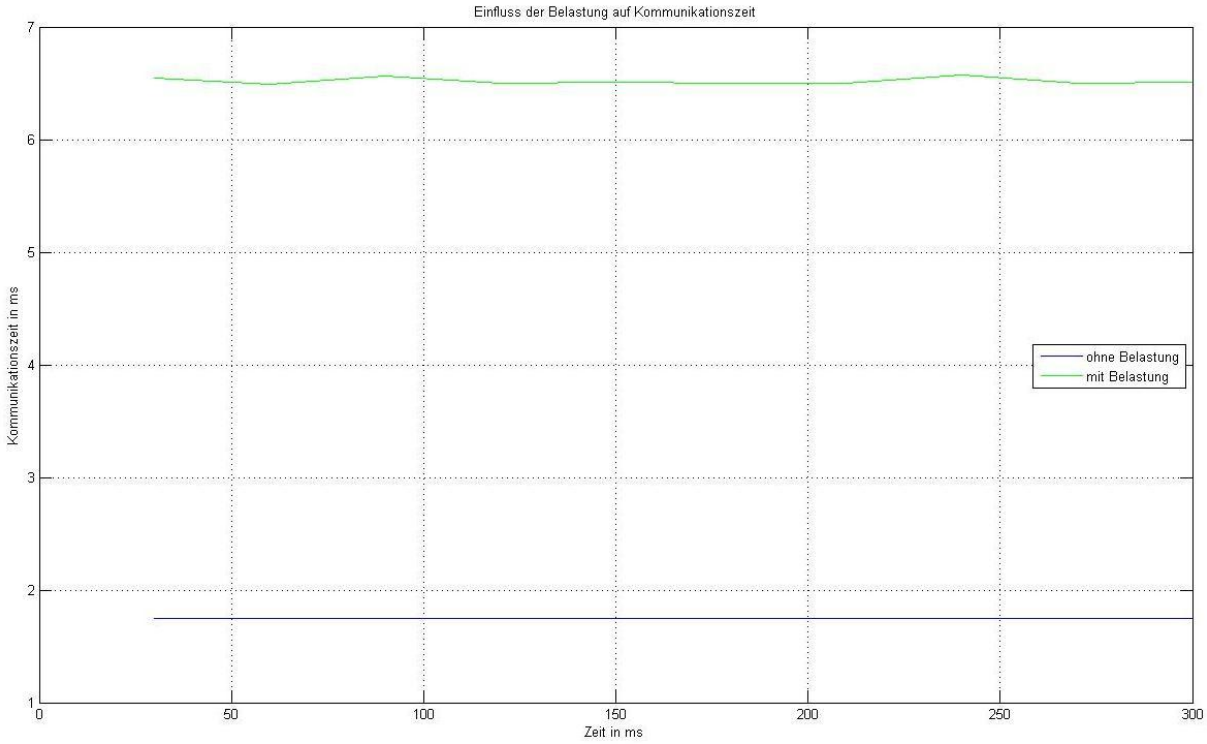

Abbildung 32 Einfluss Belastung auf Kommunikationszeit bei fester UDP Verbindung auf DELL Precision

Der Vergleich der Kommunikationszeiten zwischen fester und ohne Verbindung wurde bei einem Timer von 30 s und ohne Belastung vollzogen, da sich diese Kombination als die stabilste herauskristallisierte. B&R verspricht eine Performancesteigerung von 30 Prozent bei einer festen Verbindung gegenüber einer Verbindung, die immer wieder neu aufgebaut wird. Wie in der Abbildung unten deutlich zu sehen, trifft zwar zu, dass die permanente Verbindung schneller kommunizieren kann, allerdings nicht um 30 Prozent. Im Durchschnitt hat die lose Verbindung 1.8377 ms für die Übertragung gebraucht. Demnach ist für die feste Verbindung ein theoretischer Wert von 1.2864 ms zu erwarten, da dies 30 Prozent entspricht. Es ergibt sich allerdings nur ein Mittelwert von 1.7474 ms, was einer prozentualen Performancesteigerung von 4.91 Prozent entspricht.

Außerdem erweist sich die feste Verbindung stabiler als die Lose. Die Differenz der aufgenommen Werte in der Messreihe der losen Verbindung beträgt 23.4 µs, wobei die der festen Verbindung lediglich 7.3 µs ergibt. In Prozent ausgedrückt, beträgt die Stabilitätsabweichung also 31.2 Prozent.

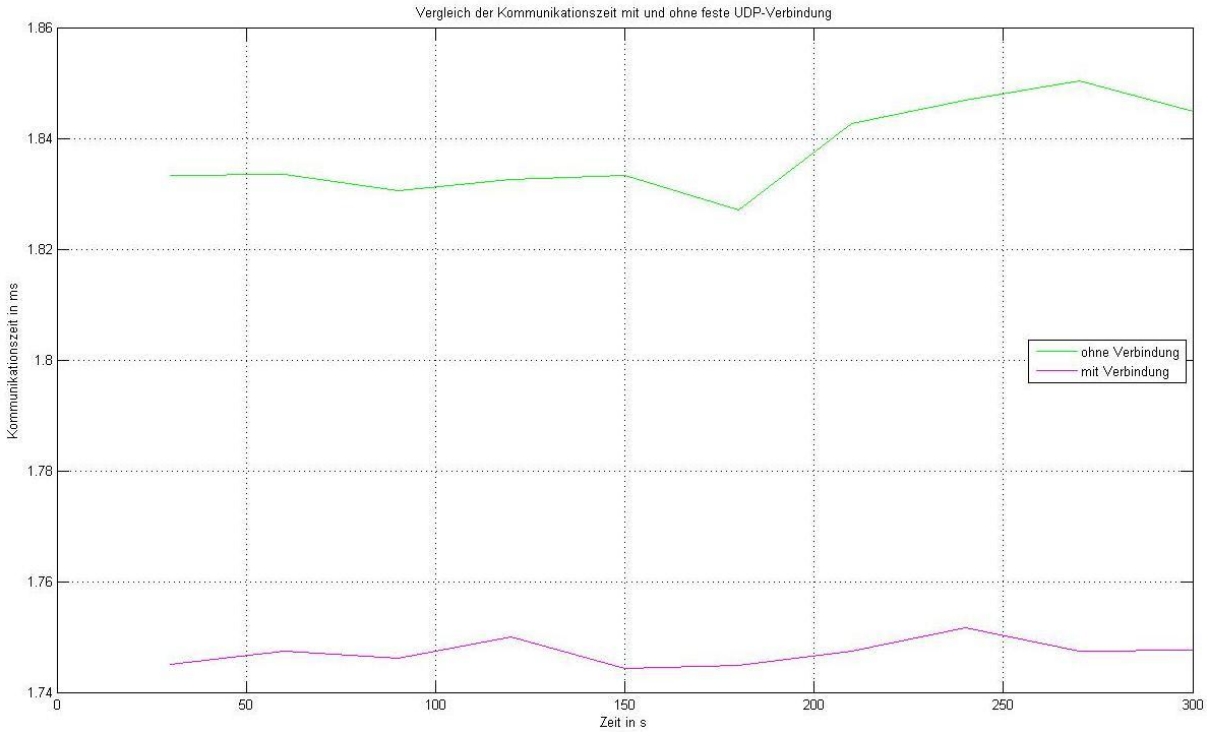

Abbildung 33 Vergleich der Kommunikationszeit mit und ohne feste Verbindung auf DELL Precision ohne Last

4.5 TCP Test auf DELL Precision M65

Auch das Transport Control Protocol ist ein freies Protokoll, welches an die jeweilige Benutzeroberfläche angepasst werden muss. Im Automation Studio werden hierfür Funktionsbausteine bereitgestellt. Auch diese werden an dieser Stelle nicht weiter erläutert, ihre Funktionsweise und Parametrierung ist dem Quellcode im Anhang zu entnehmen. Um das TCP möglichst eins zu eins mit UDP vergleichen zu können, ist es sinnvoll, dieselbe Struktur zu verwenden. Zwar kann TCP größere Datenmengen versenden und muss demnach nicht auf 1400 Byte reduziert werden, dennoch wurde diese Größe gewählt, um dieselbe Struktur verwenden zu können. Auch verfügt TCP über eigene Sicherheitsmechanismen, sodass die extra für UDP implementierten Funktionen nicht von Relevanz sind. Dennoch wurden sie ebenfalls in das Protokoll integriert, da so eventuell auftretende Fehler sichtbar gemacht und entweder im Automation Studio oder im Control.6000 Logger dargestellt werden.

Wie schon bei dem vorangegangenen Test mit UDP wurde das Lastprogramm der dritten Zeitscheibe mit 1.2 ms zugeordnet, das eigentliche Testprogramm der Zweiten mit 0.8 ms, 1.2 ms, 2 ms und 10 ms und die Realisierung von TCP wird mit höchster Priorität mit 0.4 ms abgearbeitet.

Da sich aus den vorher aufgenommenen Messreihen mit PVI ergeben hat, dass der DELL Precision bessere Ergebnisse liefert als der DELL Latitude, wurde die TCP-Kommunikation nur noch auf dem Precision Rechner realisiert und getestet.

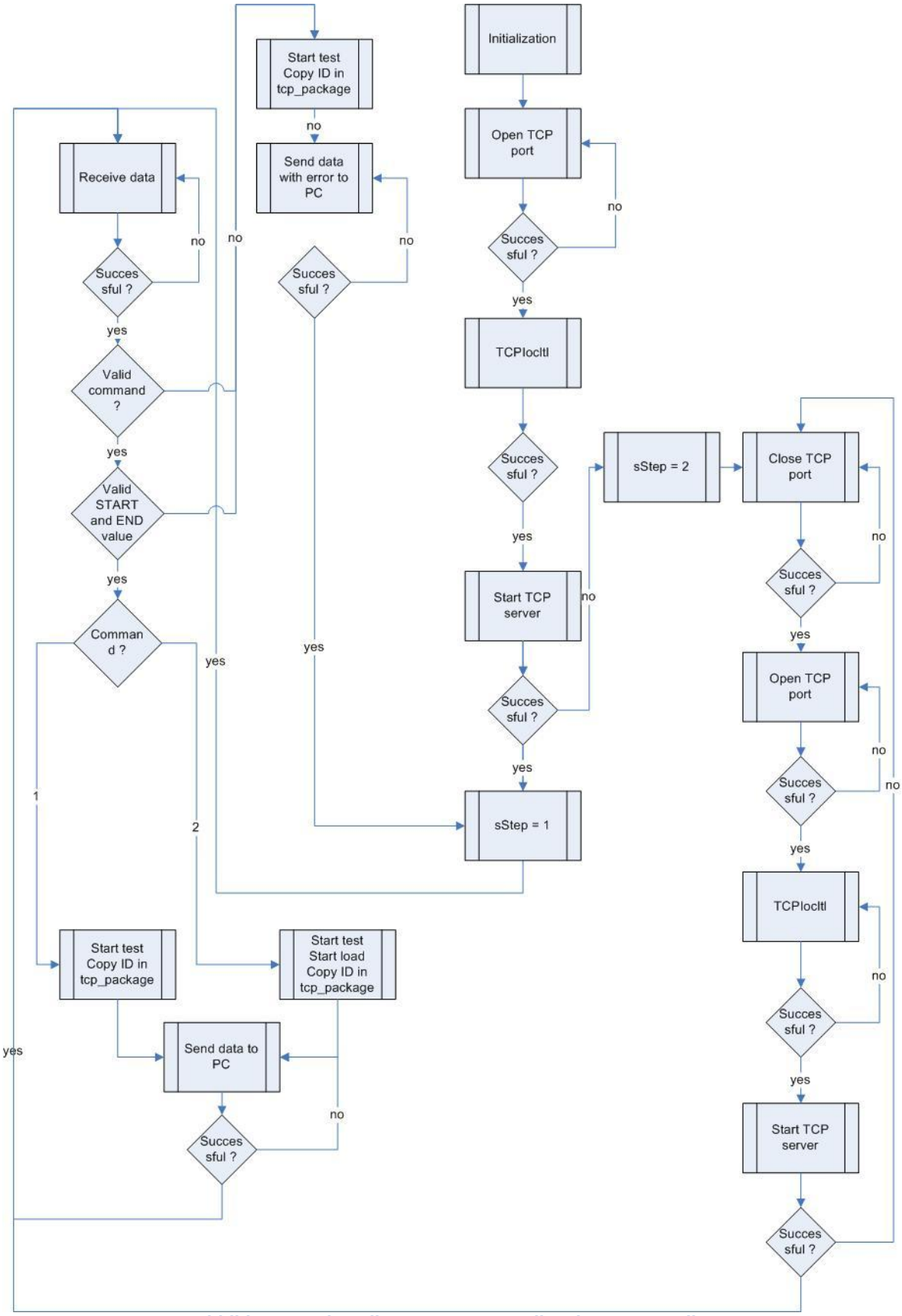

Abbildung 34 Flussdiagramm TCP Quelle: eigene Darstellung

Die Zykluszeit hat keinen Einfluss auf die Kommunikationszeit bei einem Transport Control Protocol. Die Abweichungen sind in jedem Zyklus gleich groß und betragen 57.4 µs, allerdings schwanken die Werte nur zwischen zwei Werten, was wieder einen Eindruck von Stabilität vermittelt.

Herausstechend ist die hohe Kommunikationszeit, die TCP benötigt um einen Datenaustausch zu vollziehen. Da die Werte, wie oben beschrieben, um zwei Werte schwanken, ist anzunehmen, dass eine interne Begrenzung stattfindet. Um auszuschließen, dass dies an einem Programmierfehler liegt, wurde eine TCP-Kommunikation zwischen zwei Rechnern realisiert. Diese konnte einen Datenaustausch innerhalb von 15.6 ms realisieren, wobei der zugehörige Quellcode noch eine „Bremse“ implementiert hatte, welche eine maximale Übertragungsgeschwindigkeit von 12.6 ms sicherstellte. Dies hat zur Folge, dass die niedrige Performance auf die Steuerungssoftware zurückzuführen ist, aus diesem Grund wurde das TCP-Plugin nicht geändert. B&R konnte für das Verhalten keine Begründung geben, eine interne Begrenzung ist nicht bekannt. Auch konnte die Firma keine Richtwerte für eine TCP-Kommunikation liefern. Aus TCP-Realisierungen mit anderen Steuerungsherstellern innerhalb der YXLON International GmbH konnte aber auf Erfahrungswerte zurückgegriffen werden, die eine niedrige Performance mit ca. 40 ms bestätigen.

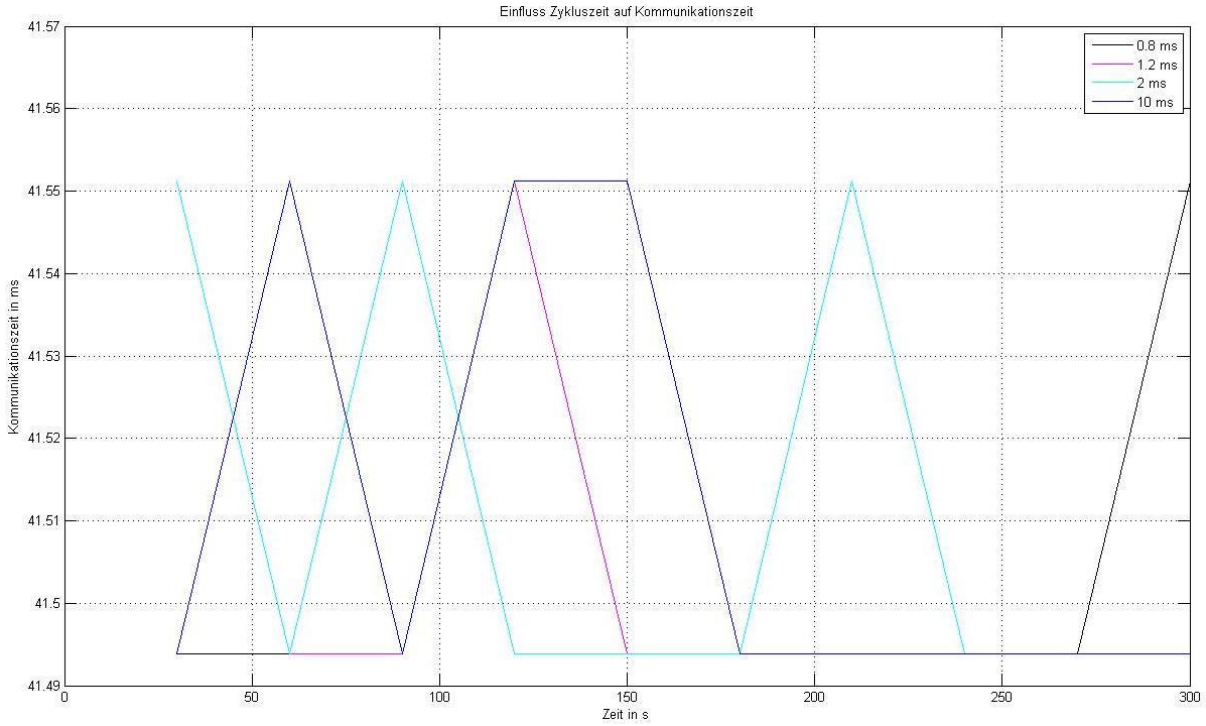

Abbildung 35 Einfluss der Zykluszeit auf Kommunikationszeit bei TCP ohne Last

Die Timerlänge hat einen sehr geringen Einfluss auf die Stabilität der Werte. Wird ein Timer mit einer Länge von 1 s oder 10 s verwendet, werden konstant dieselben Werte gemessen, in beiden Messreihen tritt lediglich ein Ausreißer auf. Diese Ausreißer sind jedoch verhältnismäßig hoch. Bei einem Timer von 1 s ergibt sich ein Ausreißer von 1.6667 ms, was 4 Prozent entsprechen. Bei einem Timer von 10 s wirkt sich die Abweichung nur mit 0.1729 ms aus. Am geringsten sind die auftretenden Abweichungen aber bei einem Timer mit einer Dauer von 30 s, hierbei ergeben sich zwar dreimal häufiger Abweichungen, jedoch betragen diese lediglich 57.4 µs, so dass sich dieser Timer dennoch als der stabilste herausstellt.

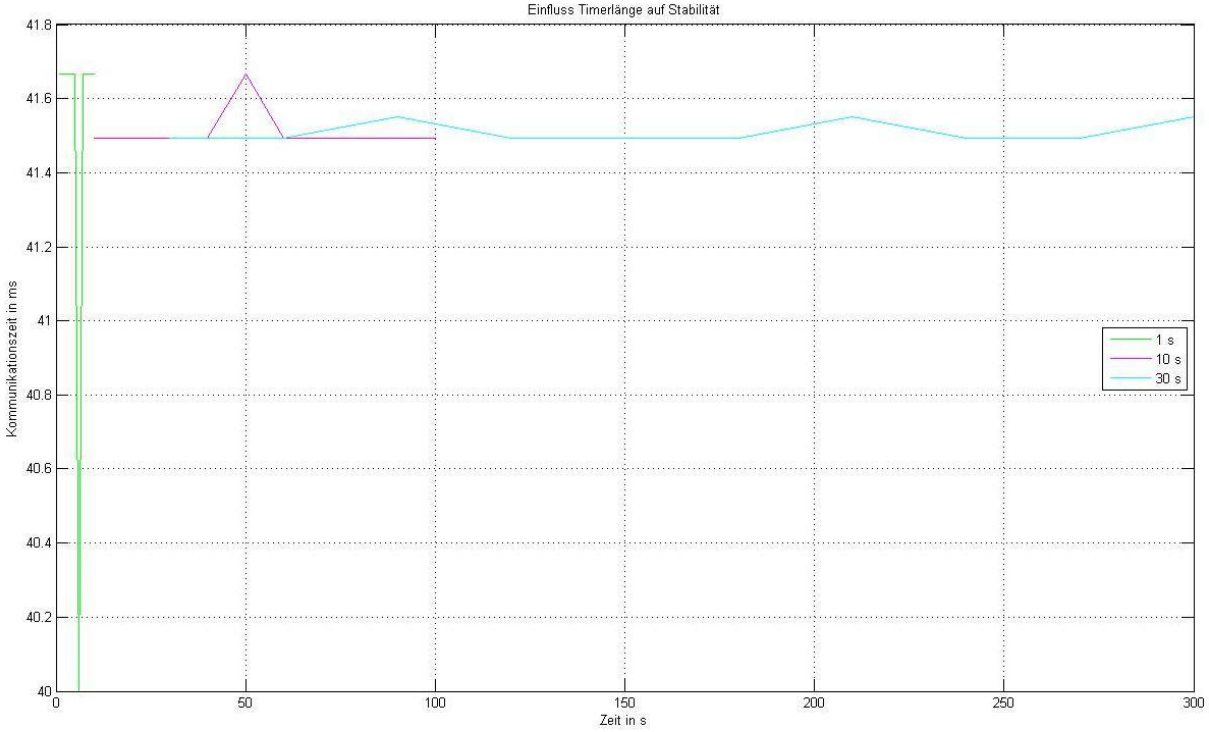

Abbildung 36 Einfluss Timerlänge auf Stabilität bei TCP ohne Last

Wie schon bei beim User Datagram Protocol ist bei TCP zu erwarten, dass die CPU-Auslastung von Rechner und Steuerung direkt von der Kommunikationszeit abhängt. Wird die unten dargestellte Grafik betrachtet, scheint dies aber nicht der Fall zu sein. Bei der Auslastung der SPS ist entgegen der erwarteten abschüssigen Rampe, eher eine steigende Rampe zu erkennen. Also verursacht eine hohe Übertragungszeit eine hohe Auslastung der Steuerungs-CPU.

Beim Verlauf der Rechnerauslastung ist kein sinnergebender Verlauf zu erkennen. Wie schon bei der Messreihe mit UDP wurden die Werte sehr ungenau aufgenommen, indem sie manuell vom Taskmanager des PCs abgelesen wurden. Dennoch sind große Ausreißer um bis zu 40 Prozent sehr ungewöhnlich. Der Einfluss der Kommunikationszeit auf die Rechnerauslastung kann mit diesen Werten nicht bewertet werden.

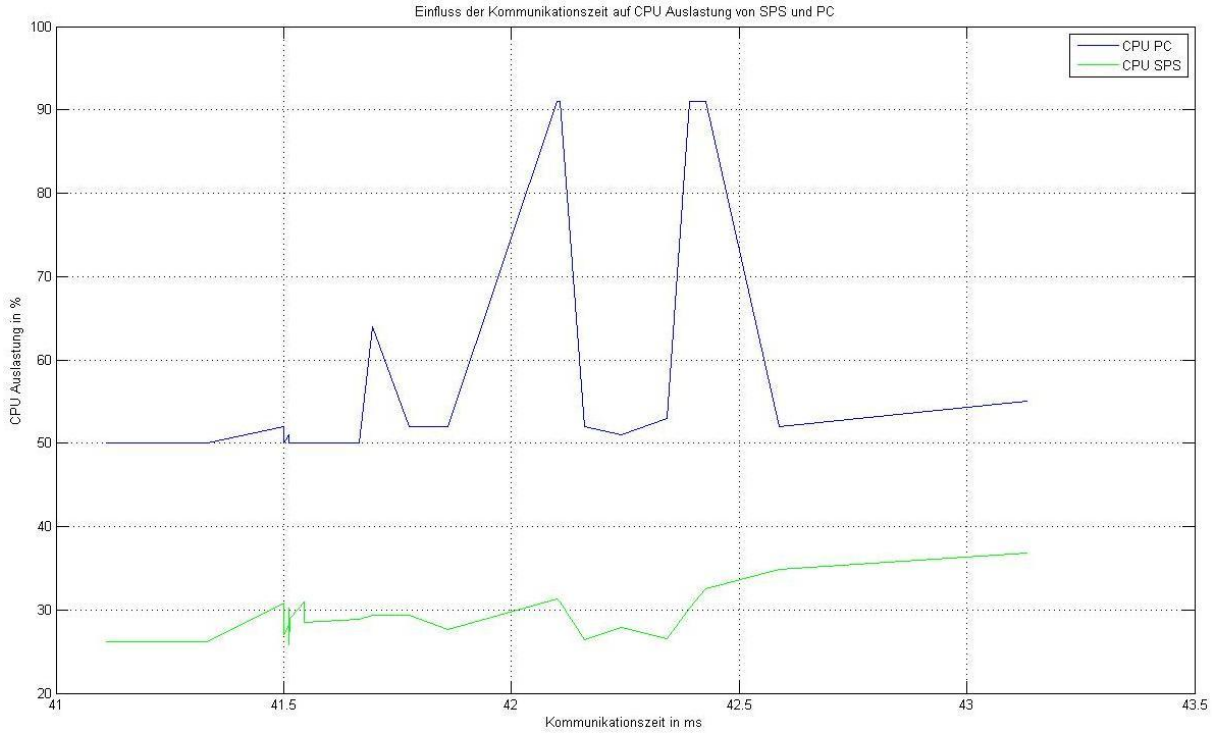

Abbildung 37 Einfluss der Kommunikationszeit auf CPU-Auslastung bei TCP

Beim Zuschalten des Testprogramms ergibt sich eine deutlich höhere Übertragungszeit. Das Transmission Control Protocol benötigt bis zu 0.92 ms länger um einen erfolgreichen Datenaustausch zu realisieren. Das entspricht einer durchschnittlichen Performanceverminderung von 2.16 Prozent. Zwar ist diese Minimierung nicht ausschlaggebend hoch, dennoch ist ebenso negativ zu erwähnen, dass auch die Stabilität der Werte bei Belastung abnimmt. Sie schwankt um 0.5379 ms, was bei Betrachtung des Wertes beim Test ohne Last mit 0.0574 ms groß ins Gewicht fällt. TCP zeichnet sich unter Belastung durch eine noch höhere Übertragungsdauer sowie eine schlechte Stabilität aus.

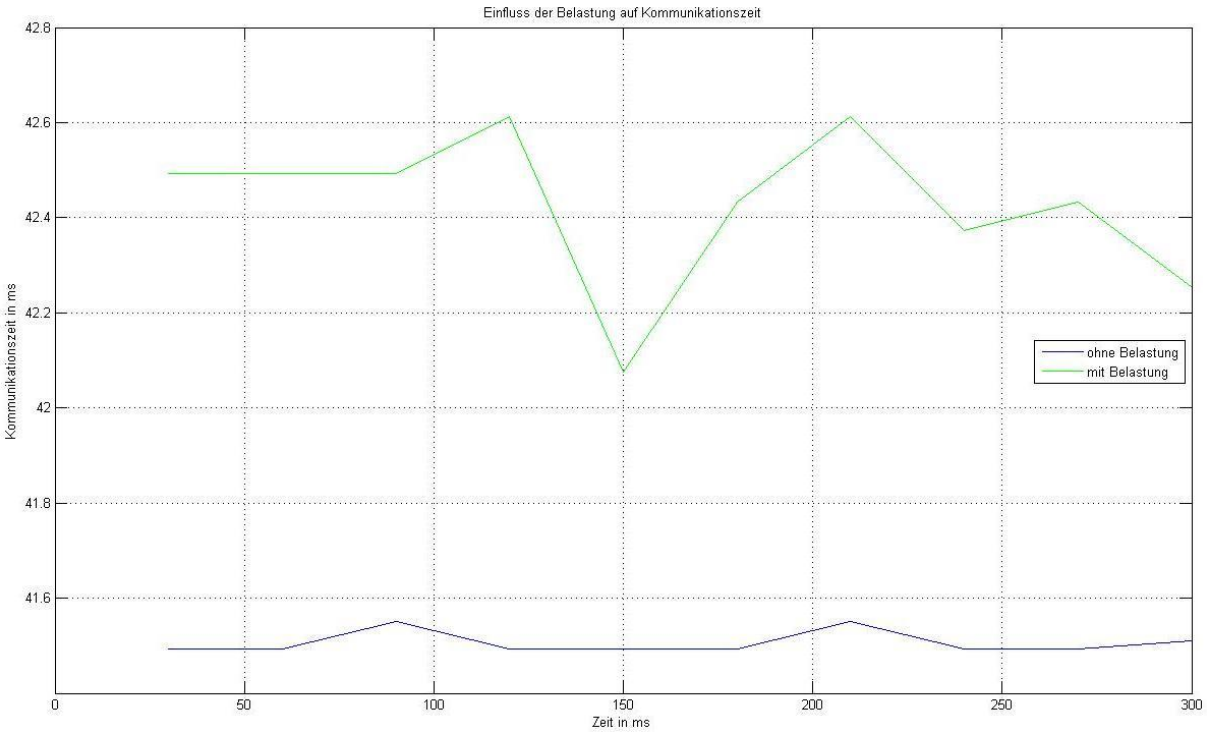

Abbildung 38 Einfluss der Belastung auf Kommunikationszeit bei TCP

5 Auswertung

Es gilt nun, die getesteten Transportprotokolle zu bewerten, indem die Ergebnisse der Tests miteinander verglichen werden. Ziel der Auswertung ist es, ein Protokoll zu benennen, welches sowohl für das Projekt „Asterix“ als auch als Standardinterface geeignet ist.
Herausstechend ist, dass weder das Prozess Variablen Interface noch das Transport Control Protocol die geforderten 10 ms für eine erfolgreiche Datenübertragung zwischen Steuerung und Computer erreichen. In der unten dargestellten Abbildung wurden die Ergebnisse der Messreihe mit einem Timer von 30 s Länge visualisiert. PVI überträgt die Daten auf dem DELL Latitude als auch auf dem DELL Precision sehr langsam. Obwohl der DELL Precision die leitungsstärkeren Prozessoren besitzt, erfolgt die Datenübertragung hier noch langsamer (durchschnittliche Differenz: 1.3487 ms). Da bei leistungsstärkeren Prozessoren eine schnellere Kommunikation zu erwarten ist, wird dieser Vorgang als untypisch bewertet und die Annahme getroffen, dass die Instabilität vom Prozess Variablen Interface hierfür verantwortlich ist. Die Instabilität ist auf dem DELL Precision sogar noch größer als auf dem DELL Latitude, auf dem sich die Werte stabiler verhalten. Dies ist mit der Tatsache zu begründen, dass die CPU des Latitude-PCs zu 100 Prozent ausgelastet war und somit in die Begrenzung kam. Schlussfolgernd hat es nur den Anschein, als ob sich die Werte stabiler verhalten, denn tatsächlich werden sie von dem Prozessor des Rechners beschränkt.

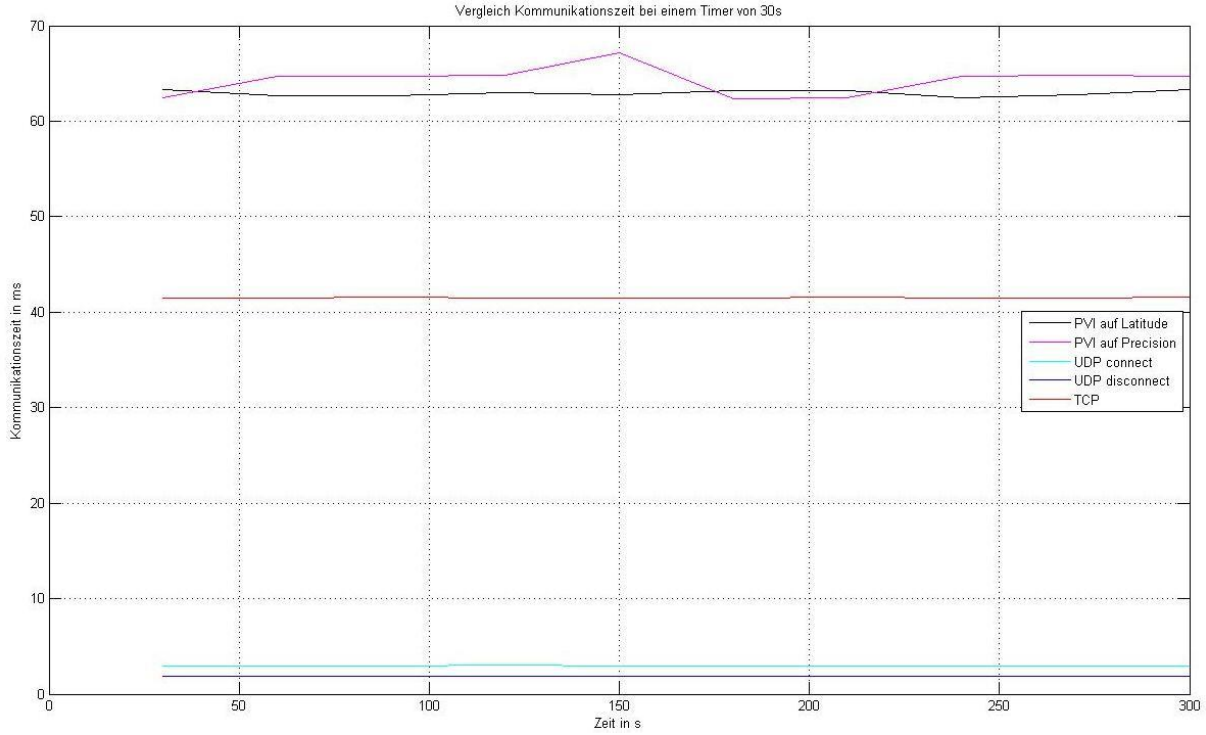

Abbildung 39 Vergleich der Kommunikationszeiten unter PVI, UDP und TCP ohne Last

TCP hingegen zeichnet sich durch eine sehr hohe Stabilität aus, die Werte schwanken in der unten dargestellten Messreihe lediglich um zwei Werte, die nur 0.0574 ms auseinander liegen. Allerdings liegt die Kommunikationszeit mit durchschnittlich 41.5510 ms deutlich über 10 ms und erfüllt somit die wichtigste Anforderung, nämlich eine Performance unter 10 ms, nicht. Die lange Kommunikationszeit kommt durch die vielen Sicherheitsmechanismen zu Stande, die im Transport Control Protocol integriert sind. Der eigentliche Transport der Daten erfolgt sehr schnell, aber die Verarbeitung der Sicherheitsfunktionen benötigt durch ihre Komplexität viel Zeit.

Das User Datagram Protocol kann die Performanceanforderungen im Gegensatz zu den anderen Transportprotokollen erfüllen. Wird der SPS-Port während der Kommunikation dauerhaft mit dem Port des PCs verbunden, wird eine Datenübertragungs- und Verarbeitungsdauer von 1.7447 ms erreicht. Die Werte sind sehr stabil und schwanken in der oben dargestellten Messreihe lediglich um 0.0073 ms. Auch wenn die Ports nicht fest miteinander verbunden sind, kann UDP den Anforderungen standhalten. Zwar zeigen sich die Werte nicht mehr so stabil, dennoch sind die Schwankungen im Bereich von 0.0234 ms immer noch deutlich kleiner als bei PVI oder TCP.

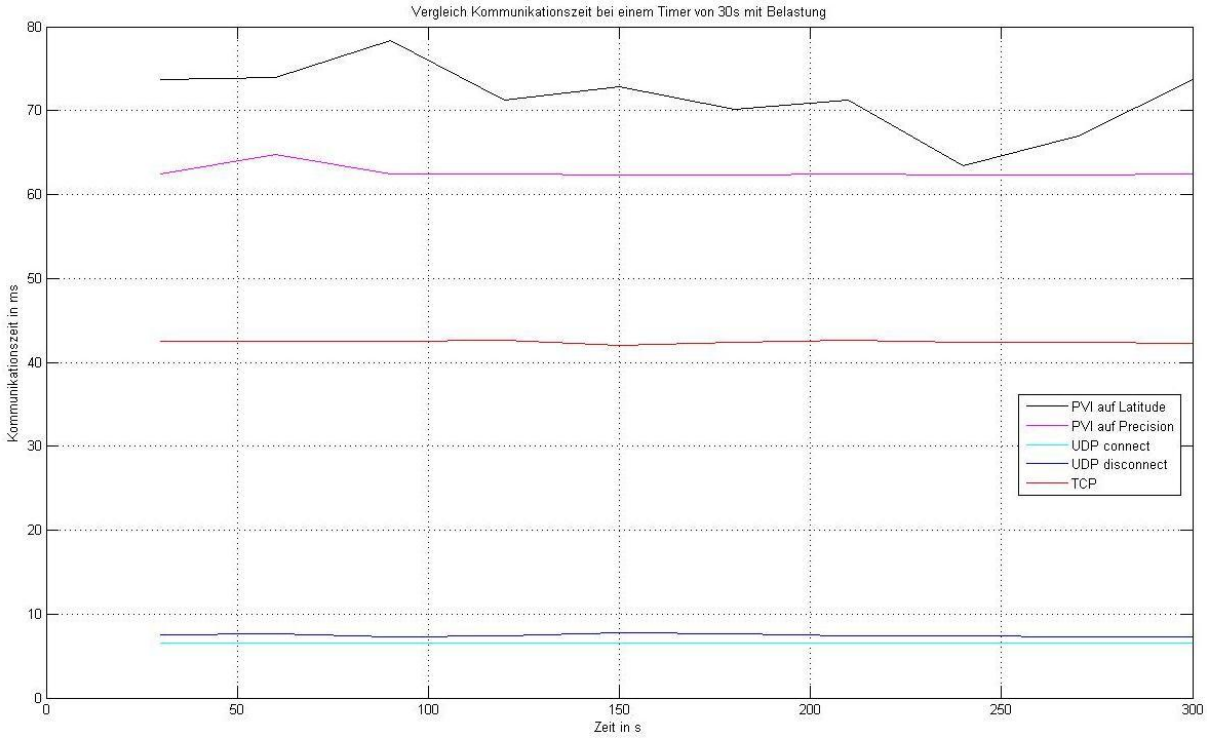

Abbildung 40 Vergleich Kommunikationszeit unter PVI, UDP und TCP mit Belastung

Unter Last verhalten sich alle Transportprotokolle wie erwartet. Die Messreihe, die mit dem Prozess Variablen Interface auf dem DELL Latitude aufgenommen wurde, verläuft beim Test unter Belastung unstetiger als bei der Messung ohne Last. Der pink dargestellte Graph schwankt lediglich um 2.2948 ms, was in den Messungen mit PVI ein sehr gutes Ergebnis bedeutet. Der schwarze Graph hingegen hat sein Maximum bei 78.3290 ms und das Minimum bei 63.4249ms, was einer Differenz von 14.9041 ms entspricht. Dieses Verhalten bestätigt die Annahme, dass PVI eine große Instabilität besitzt und die Messreihen daher schwer miteinander zu vergleichen sind. Ebenso gegensätzlich zum vorangegangenen Test ist auch die Kommunikationszeit auf dem Latitude-PC. Und zwar im Mittel um 8.8682 ms höher als auf dem DELL Precision. Dies bestätigt die These, dass leistungsstarke Prozessoren eine geringe Datenübertragungsdauer gewährleisten können.

Bei Verwendung des Transmission Control Protocols hat die Belastung im Gegensatz zu PVI einen geringen Einfluss auf die Kommunikationszeit. Das Protokoll benötigt - in der Messreihe mit Belastung und einem verwendeten Timer von 30 s - durchschnittlich 0.9164 ms länger, um die Daten erfolgreich zu übertragen als in dem Test ohne Last. TCP zeigt sich ohne Last sehr stabil. Allerdings sind die Schwankungen, die unter Belastung auftreten, mit 0.5379 ms deutlich geringer als bei Anwendung von PVI.

Das User Datagram Protocol zeigt sich unter Last ebenso stabil wie ohne Belastung. Die UDP-Realisierung ohne feste Verbindung zeigt sich wie auch im vorhergegangen Test instabiler (Δ 0.5822 ms) als die UDP-Lösung mit fester Verbindung (Δ 0.074 ms).

Die Kommunikationszeiten werden allerdings sehr stark von der Last beeinflusst. Erreichte UDP mit fester Verbindung und einem Timer von 30 s noch eine mittlere Zeit von 1.7474 ms, werden unter Last durchschnittlich 6.5196 ms benötigt. Dies entspricht einem Anstieg der Zeit von 373.1 Prozent. Dennoch bleibt die benötigte Zeit um ein Datenpaket erfolgreich zu übertragen unter den geforderten 10 ms. Diese Vorgabe wird nur überschritten, wenn eine Zeitscheibenabarbeitungszeit von 10 ms eingestellt wird. Da diese Messung nicht korrekt durchgeführt wurde (siehe Kapitel 4.3), wird dieses Ergebnis nicht berücksichtigt.
Bei den Tests mit UDP sowie TCP konnte mit Hilfe des Loggers der X600 Software und der Identifikationsnummer nachvollzogen werden, ob die Datenprotokolle zuverlässig übertragen wurden. Zwischen den drei Realisierungen (UDP connect, UDP disconnect und TCP) wurde nicht festgestellt, ob ein Protokoll die Daten verlässlicher überträgt als das andere. Der Logger verzeichnete keine nennenswerten Übertragungsprobleme, so dass das Fazit gezogen werden kann, dass alle Transportprotokolle stets zuverlässig arbeiteten. Bei Verwendung von PVI konnte dies nicht geprüft werden, da der zusätzliche Sicherheitsmechanismus nicht integriert wurde.

Nach Auswertung aller Tests ergibt sich für das User Datagram Protocol, dass es sowohl als Schnittstelle als auch für das Projekt Asterix geeignet ist. Als einziges Transportprotokoll kann es eine sehr schnelle und stabile Performance sowie eine zuverlässige Übertragung der Daten aufweisen. Das Transport Control Protocol besitzt zwar ebenso wie UDP die Eigenschaft einer zuverlässigen Übertragung, doch wegen der langsamen Übertragungsgeschwindigkeit von durchschnittlich ca. 41 ms und der Komplexität beim Programmieren des X600-Plugins, eignet es sich weder als Schnittstelle für das Projekt „Asterix“ noch als Standardinterface. Auch das Prozess Variablen Interface erfüllt die Anforderungen nicht. Zwar wurde PVI von vornherein als Standardschnittstelle ausgeschlossen, die Ergebnisse der Performanceuntersuchungen schließen es aber auch als Interface für Asterix aus. Die Erwartung, dass das Transportprotokoll unter B&R Hard- und Software gute Übertragungsgeschwindigkeiten erreicht, wurde nicht erfüllt. Auch die aufgezeigten Schwankungen in den einzelnen Messreihen begründen den Ausschluss von PVI.

6 Bewertung und Ausblick

6.1 Bewertung

Durch die systematische Vorgehensweise der Analyse konnten die zur Verfügung stehenden Transportprotokolle bewertet werden. Sie wurden analysiert und getestet, so dass ein Einblick in den Aufbau und die Arbeitsweisen der einzelnen Protokolle entstand.
In dieser Arbeit wurde mit dem User Datagram Protocol ein Transportprotokoll identifiziert, welches für den zukünftigen Einsatz im Computertomographen, der unter dem Projektnamen „Asterix“ läuft, geeignet ist.
Durch die eigens durchgeführte Realisierung des UDPs kann der notwendige Aufwand, um das Protokoll zukünftig zu implementieren, eingeschätzt werden. Das Protokoll bestätigte nach der Auswertung der durchgeführten Tests alle Vorteile, die aus der Analyse hervorgingen.
Die geringe Zeit, die es braucht, um Datenpakete zuverlässig zwischen Steuerung und SPS zu übertragen, zeichnet es ebenso aus, wie seine Allgemeingültigkeit. Zwar muss UDP um zusätzliche Sicherheitsmechanismen manuell erweitert werden, da es diese nicht selbst integriert hat, was zu höheren Arbeitsstunden bei der Implementierung des Protokolls führt; dennoch steht dieser Nachteil in keiner Relation zu den Vorteilen, über die UDP verfügt.
Die Kosten, die diese Schnittstelle mit sich bringt, entstehen ausschließlich durch die Arbeitsstunden der Programmierer. Zusätzliche Hard- oder Fremdsoftware ist für die Realisierung nicht notwendig, weshalb hier auch keine Investitionen getätigt werden müssen.

6.2 Ausblick

Das User Datagram Protocol wird zukünftig als Transportprotokoll zwischen Rechner und Steuerung für das Projekt „Asterix“ in der YXLON International GmbH verwendet. Das Grundgerüst für eine Übertragung der Daten mittels UDP wurde im Zuge dieser Bachelorthesis erstellt. Es wird ein Konzept erarbeitet, welches die Umsetzung des Protokolls, auch im Zuge der Standardisierung, festhält. Es müssen Sicherheitsmechanismen entwickelt werden, die gewährleisten, dass die Übertragung der Datenpakete ohne Verlust von einzelnen Daten oder sogar ganzen Paketen geschieht. Außerdem muss eine Struktur erstellt werden, die Allgemeingültigkeit besitzt, also auch für zukünftige Anlagentypen gilt.

Anhang A
zur
Bachelorthesis

Spezifikation und Realisierung eines Standard-Interfaces
zur Anlagenvisualisierung und Steuerung

A1: Abbildungsverzeichnis

A2: Abkürzungsverzeichnis

AB	Automation Basic
ACK	Acknowledgement
B&R	Bernecker + Rainer Industrie-Elektronik GmbH
C	ANSI C
CFC	Sequential Function Chart
COM	Component Object Model
CT	Computertomographie
DCOM	Distributed Component Object Modell
DNS	Domain Name System
DX	Data exchange
FBD	Function Block Diagram
FIN	Finish
GUI	Graphical User Interface
ID	Identifikationsnummer
IL	Instruction List
IP	Internet Protocol
LAD	Ladder Diagram
OLE	Object Linking and Embedding
OPC	OLE for Process Control
PC	Personal Computer
PVI	Prozess Variablen Interface
RTT	Round Trip Time
SPS	Speicherprogrammierbare Steuerung
ST	Structured Text
SYN	Synchronize
TCP	Transport Control Protocol
UDP	User Transport Protocol
XML	Extensible Markup Language
X-Ray	Röntgenstrahlung

A3: Literaturverzeichnis

Bücher

[1] Olbrich, A. — Netze Protokolle Spezifikationen, Braunschweig/Wiesbaden: Vieweg, 2003

[2] Reißenweber, B. — Feldbussysteme zur industriellen Kommunikation, 3. Auflage, München: Oldenbourg Industrieverlag, 2009

[3] Roppel, C. — Grundlagen der digitalen Kommunikationstechnik, Leipzig: Carl Hanser Verlag, 2006

[3b] Fritzsche, G. — Theoretische Grundlagen der Nachrichtentechnik, Berlin: Veb Verlag Technik GmbH, 1972

[4] Iwanitz, I., Lange, J. — OPC Grundlagen, Implementierung und Anwendung, 2. Auflage, Heidelberg: Hüthig GmbH & Co.KG Heidelberg

[4b] Schulz von Thun, F. — Miteinander reden, Berlin: Rowohlt Taschenbuch Verlag, 2006

[5] Bettermann, T. — Anwendung von Microsoft Softwarestandards in der Automatisierungstechnik, Renningen: expert verlag, 2002

[6] Henshall, J., Shaw, S. — OSI praxisnah erklärt, London: Carl Hanser Verlag, 1992

[7] Comer, D.E. Computernetzwerke und Internets mit Internetanwendungen, 3. Auflage, München: Pearson Studium, 2002

[7b] Mahnke, W., Leitner, S., Damm, M. OPC Unified Architecture, Berlin: Springer Verlag, 2009

Betriebsdokumente

[8] B&R Speichermanagement und Datenhaltung TM250

[9] B&R PVI Kommunikation TM710

[10] B&R PVI OPC TM730

[11] B&R Automation Studio Diagnose TM223

[11.b] B&R Automation Studio Basis TM210

Internetquellen

[12] http://www.cobocards.com/pool/de/cardset/4065027/online-karteikarten-rechnernetze-1/
15.07.2012, 17:15 Uhr

[13] http://www.academic.ru/dic.nsf
15.07.2012, 17:18 Uhr

[14] http://fbim.fh-regensburg.de/~hab39652/PG1/skriptum/…/03_Sockets.ppt
15.07.2012, 17:20 Uhr

[15] http://www.plt.rwth-aachen.de/fileadmin/plt/events/2007/Sommerkolloquium/Otto.pdf
15.07.2012, 17:20 Uhr

[16] http://www.opcfoundation.org/Default.aspx/01_about/01_whatis.asp?MID=AboutOPC
15.07.2012, 17:21 Uhr

[17] http://www.hjr-verlag.de/imperia/md/content/hjr/produktinfo/huethig/978-3-7785-2903/Leseprobe.pdf
15.07.2012, 17:22 Uhr

[18] http://www.ipcomm.de/protocol/OPC/en/sheet.html
15.07.2012, 17:23 Uhr

[19] http://www.sps-lehrgang.de/sps-systeme/
15.07.2012, 17:24 Uhr

[20] http://www.ebookbrowse.com/folien-sys-vorlesung-sammlung-ws0708-pdf-d246657459
15.07.2012, 17:25 Uhr

A4: Daten-CD

Auf der letzten Seite dieser Bachelorthesis befindet sich eine eingeklebte Daten-CD, die alle Messergebnisse von Versuchsdurchführungen, sowie jeglicher, zur Erstellung dieser Arbeit, genutzter Quellcode beinhaltet. Außerdem befinden sich auf ihr Datenblätter und Übersichtstabellen. Zusätzlich befindet sich diese Thesis im PDF-Format auf der Daten-CD.

7 Eidesstattliche Erklärung

Name: Wüst
Vorname: Carina
Matrikel-Nr.: 1937456
Studiengang: Informations- und Elektrotechnik

Hiermit versichere ich an Eides statt und durch meine Unterschrift, dass die vorliegende Arbeit von mir selbstständig und ohne fremde Hilfe angefertigt worden ist. Inhalte und Passagen, die aus fremden Quellen stammen und direkt oder indirekt übernommen worden sind, wurden als solche kenntlich gemacht. Ferner versichere ich, dass ich keine andere, außer der im Literaturverzeichnis angegebenen Literatur verwendet habe. Diese Versicherung bezieht sich sowohl auf Textinhalte sowie alle enthaltenen Abbildungen, Skizzen und Tabellen. Die Arbeit wurde bisher keiner Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Hamburg, den 21. Juni 2012

Carina Wüst