Fachhochschule Köln
Cologne University of Applied Sciences

Kölner Beiträge zur Technischen Informatik

Ausgabe-Nr. 1 / Sommer 2011

ISSN 2193-570X

Herausgeber:

Rainer Bartz
Gregor Büchel (Redakteur)
Andreas Grebe
Georg Hartung
Hans W. Nissen
Lothar Thieling
Carsten Vogt

Unter Mitwirkung von:

Sebastian Seegert

Impressum:

Forschungsschwerpunkt Verteilte und Mobile Applikationen
Fachhochschule Köln
Fakultät für Informations-, Medien- und Elektrotechnik
Betzdorfer Str. 2
D-50679 Köln
gregor.buechel@fh-koeln.de
Stand: August 2011

ISSN: 2193-570X

Inhalt

Editorial

Seit dem Jahre 2002 betreibt die Fakultät für Informations-, Medien- und Elektrotechnik am Institut für Nachrichtentechnik einen Bachelor und Master Studiengang der Technischen Informatik. Im Umfeld des Master Studienganges entstanden eine Reihe von Forschungs- und Entwicklungsprojekten auf dem Gebiet der Technischen Informatik. Im Jahre 2009 wurde ein Forschungsschwerpunkt **„Verteilte Mobile Applikationen“** gegründet, der diese Aktivitäten bündelt. Um Ergebnisse laufender Arbeiten aus den Forschungs- und Entwicklungsaktivitäten des Forschungsschwerpunkts nach außen zu kommunizieren und um Informatiker und Informationstechniker außerhalb des Forschungsschwerpunkts z.B. aus dem Kölner Raum einzuladen, neue Ergebnisse aus Wissenschaft und technischer Anwendung zu publizieren, wird diese WEB Schriftenreihe gegründet, in der forschungsorientierte Aufsätze und Monographien veröffentlicht werden können. Der Herausgeberkreis freut sich, die erste Nummer der Reihe der Fachöffentlichkeit zur kritischen Prüfung und zur möglichen Mitwirkung vorlegen zu können.

Rainer Bartz

Gregor Büchel

Andreas Grebe

Georg Hartung

Hans W. Nissen

Lothar Thieling

Carsten Vogt

Systemarchitektur für verteilte, mobile Anwendungen

Anton Gillert

Abstract

In diesem Beitrag wird eine Dienstplattform präsentiert, mit der Smartphones und andere mobile Endgeräte nicht mehr nur als Dienstkonsumenten, sondern auch als Dienstanbieter auftreten können. Zur Optimierung des Datenverkehrs zwischen den Dienstkonsumenten und Dienstanbietern sowie der Abstraktion von Netzbetreibern wird ein Peer-to-Peer Modell realisiert. Für die Nutzer bzw. Endgeräte ist es unbedeutend, ob sie im Netz des Dienstanbieters oder in anderen Netzen z.B. durch NAT und Firewall von diesem getrennt sind.

1 Einleitung

Mobiltelefone und andere kleine mobile Endgeräte werden immer leistungsfähiger: immer mehr Dienste, die früher ein Notebook oder einen stationären Computer erforderten, können heute auf “Smartphones” realisiert werden. Besonders interessant für solche mobilen Plattformen sind verteilte Anwendungen, in denen multimediale Daten ausgetauscht werden. Aber auch die Leistungsfähigkeit von eingebetteten Systemen nimmt immer weiter zu. So verfügen diese aktuell nicht nur über eine hohe Rechenleistung, sondern auch über eine große Anzahl an drahtlosen Kommunikationstechnologien.

Besonders interessant für mobile Plattformen sind verteilte Anwendungen, in denen ortsbezogene und multimediale Daten verarbeitet werden. Ziel des VMA-Projekts ist die Entwicklung einer Architektur mit ressourcensparenden Clientfunktionen für Smartphones, größtmöglicher Kompatibilität zu Diensten in heterogenen Netzen, Unterstützung der Kooperation von verschiedenen Hard- und Softwareplattformen sowie entsprechenden Portierungsmöglichkeiten.

Die Aufgaben die sich dabei stellen, lassen sich in zwei Kategorien unterteilen. Zum einen müssen netzwerktechnische Probleme gelöst werden. Zum anderen muss eine Applikationsbasis geschaffen werden, die Dienste über das Netz anderen Teilnehmern zur Verfügung stellen kann. Im Folgenden werden die Systeminfrastruktur und ihre Elemente erläutert.

2 Systeminfrastruktur

Die Systeminfrastruktur setzt sich aus den im folgenden beschriebenen Systemkomponenten sowie aus der Spezifikation des Kommunikationsprotokolls und des Adressierungsschemas zusammen.

2.1 Systemkomponenten

Die Systemkomponenten ermöglichen es, Dienste auf einem Smartphone anzubieten. Als solche Elemente wurden der VMA Registrar, VMA Proxy, VMA User Agent Server sowie der VMA User Agent Client entworfen und implementiert. Die Interaktion dieser Komponenten ist in **Bild 1** exemplarisch dargestellt.

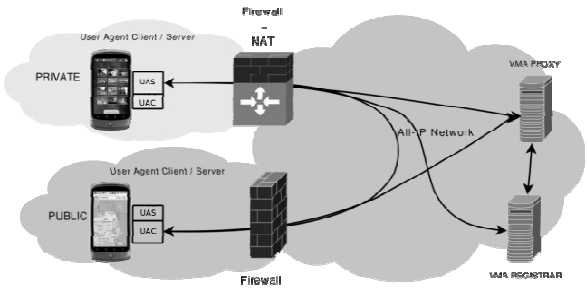


Bild 1 VMA-Systemkomponenten

VMA Registrar

Der VMA Registrar ist die zentrale Anlaufstelle für das Registrieren und Auffinden von Dienstanbietern und der Bezugspunkt für den Bootstrap der Dienstanbieter. Des Weiteren dient der Registrar der Authentifikation, Autorisation und der Buchführung von Nutzern.

VMA Proxy

Der VMA Proxy dient als Relay für den Verbindungsaufbau (und ggf. für den Datenaustausch) mit einem Dienstanbieter. Der VMA Proxy ermöglicht unter anderem eingehende Verbindungen durch NATs bzw. Firewalls.

VMA User Agent Server (UAS)

Die Hauptaufgabe der UAS ist die Bereitstellung von Diensten für User Agent Clients. Des Weiteren steuern sie die Kommunikation mit den User Agent Clients. Erreichbar sind UAS über einen VMA Proxy, zu dem sie eine permanente Verbindung halten. Jeder UAS registriert

einen VMA Proxy beim VMA Registrar und ist dadurch auch hinter NATs oder Firewalls erreichbar.

VMA User Agent Client (UAC)

UACs stellen die Dienstnutzer dar. Sie finden über einen Proxy bzw. beim Registrar mögliche Dienstanbieter und stellen Anfragen an diese.

2.2 Adressierungsschema

Im VMA-Netz werden zwei Arten von Adressen genutzt: permanente und temporäre Adressen. Dieses Adressierungsschema ist an das SIP Protokoll angelehnt, welches in [8] definiert ist. Die Notwendigkeit dieser beiden Adresstypen ergibt sich aus der Network Address Translation (NAT)-Problematik, welche in Kapitel 3 detailliert dargelegt wird.

Die UAS und UAC haben abhängig vom Provider häufig wechselnde IP-Adressen. Um einen UAS dennoch stets unter einer festen Adresse erreichen zu können, muss ein Adressierungsschema eingesetzt werden, welches unabhängig von der momentanen IP-Adresse des UAS ist.

Deshalb erzeugt jeder Benutzer im VMA-Netz bei der Registrierung einen Alias. Aus diesem Alias wird eine permanente Adresse abgeleitet:

mustermann@vma.fh-koeln.de

Die temporäre Adresse setzt sich aus diesem Alias und der aktuell verwendeten IP, unter welcher der UAS erreichbar ist, zusammen:

mustermann@139.6.19.222

Anhand der permanenten Adresse kann beim Registrar erfragt werden über welchen Proxy ein UAS erreichbar ist. Da die UAS eine Verbindung zu den Proxies halten, sind diese in der Lage die permanente Adresse in die temporäre Adresse zu übersetzen.

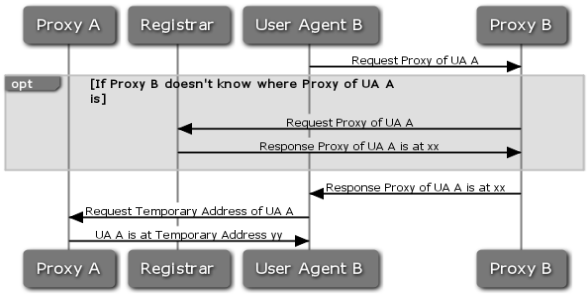


Bild 2 MSC: Adressierung

Bild 2 beschreibt das Protokoll, über das die temporäre Adresse eines UAS ermittelt werden kann. Möchte ein UAC eine Verbindung zu einem UAS aufbauen, so erfragt er zunächst den Proxy des UAS bei seinem eigenen Proxy. Kennt dieser den Proxy des UAS nicht, muss er den Proxy des UAS beim Registrar anfordern. Ist der Proxy des UAS bekannt, erhält der UAC die temporäre Adresse des UAS direkt von seinem Proxy.

Im Folgenden werden die Anforderungen an die Netzwerkinfrastruktur des VMA-Netzes formuliert und die Problematik der im Internet eingesetzten NAT-Technologie erörtert.

2.3 Netzwerkinfrastruktur

Im VMA-Forschungsprojekt ist die Peer-to-Peer-Kommunikation (P2P) der einzelnen mobilen Endgeräten ein Schwerpunkt. Um die P2P-Kommunikation in einem so heterogenen Umfeld, wie sie die mobilen Endgeräte und insbesondere die Smartphones mitbringen, zu ermöglichen, muss die Netzwerkinfrastruktur so ausgelegt werden, dass sie folgende Kriterien erfüllt:

Fehlertoleranz

VMA Proxy und die User Agent Client/Server (UAC/UAS) müssen Übertragungsfehler erkennen und gegensteuern, soweit es die verwendeten Protokolle nicht eigenständig leisten. Ebenso muss eine Fehlertoleranz bezüglich Standortswechsel bzw. IP-Adresswechsel der UAS gewährleistet werden. UAS müssen unabhängig von ihrer momentanen IP-Adresse stets erreichbar sein.

Erkennen der Netztopologie

Falls sich Client und Server im selben Netz befinden, soll der UAS dies erkennen und eine direkte P2P-Kommunikation initiieren.

Erkennung von Firewall und NAT

UAS und UAC müssen in der Lage sein, mögliche Einschränkungen bei NAT bzw. Firewalls im Netz zu erkennen und Maßnahmen einzuleiten, die eine uneingeschränkte Konnektivität gewährleisten.

3 NAT Problematik

Sollen Smartphones über das Internet kommunizieren, so müssen Probleme berücksichtigt werden, die durch den Einsatz von Network Address Translation (NAT) und Firewalls auftreten. Insbesondere durch den Einsatz von NAT treten Hürden beim Verbindungsaufbau auf, die sich ohne einen Proxy nicht umgehen lassen.

NAT kommt im Internet hauptsächlich aus zwei Gründen zum Einsatz: Zum einen um der IPv4-Adressknappheit entgegenzuwirken, zum anderen um Netztopologien zu verbergen. In der Regel kann kein Verbindungsaufbau zu Geräten erfolgen, wenn diese hinter einem NAT-Device positioniert sind. Das liegt daran, dass die hinter einem NAT-Device eingesetzten privaten IP-Adressen nicht bekannt sind bzw. im Internet nicht geroutet werden.

In Kombination mit NAT-Devices verhindern Firewalls einen Verbindungsaufbau zu den UAS aus dem Internet, indem sie eingehende Verbindungen blockieren. Üblicherweise beinhaltet ein NAT-Device auch eine Firewall.

Es gibt Verfahren mit denen man diese Einschränkung teilweise umgehen kann. Eine Möglichkeit die durch NAT verursachten Probleme zu umgehen ist der Einsatz von STUN (Simple Traversal of UDP through NAT) [4]. STUN ist ein Protokoll, welches es ermöglicht den eingesetzten NAT Typen in Erfahrung zu bringen und ggf. zu umgehen. STUN unterscheidet dabei vier verschiedene NAT Typen:

- *Full Cone NAT:*

Ein Full Cone NAT ist eine statische Zuweisung von internen auf externe Sockets.

- *Restricted Cone NAT:*

Ein Restricted Cone NAT erlaubt nur einen Datenaustausch, der von der internen Seite initiiert wird, und filtert von außen kommende Pakete anhand der Quell-IP. Eingehende Pakete können nur das NAT passieren, wenn bereits Pakete von intern an die externe Adresse versendet wurden.

- *Port Restricted Cone NAT:*

Ein Port Restricted Cone NAT verhält sich analog zum Restricted Cone NAT mit dem Zusatz, dass nicht nur nach IP Adressen gefiltert wird, sondern zusätzlich auch nach Portnummern. Pakete von außen können nur das NAT passieren, wenn bereits Pakete von intern an die externe Adresse und Portnummer versendet wurden.

- *Symmetric NAT:*

Ein Symmetric NAT erstellt für jede neue Verbindung ein neues Portmapping. Zusätzlich wird wie beim Port Restricted Cone NAT nach Quelladresse und Portnummer gefiltert.

Die ersten drei NAT-Varianten können mit STUN passiert werden. STUN verwendet hierzu das UDP-Protokoll, da Verbindungen über TCP bei keiner NAT-Variante uneingeschränkt möglich sind. Ein Client hinter einem NAT kann seine zugewiesene IP sowie die jeweilige Portnummer in Erfahrung bringen, indem er Pakete an einen STUN-Server sendet. Die zugewiesene Adresse kann er nun über seinen Proxy nach außen bekannt geben (s. auch **Bild 3**).

Zusätzlich muss er einen Port in der Firewall für ankommende Pakete öffnen, damit ein Verbindungsaufbau von außen möglich ist.

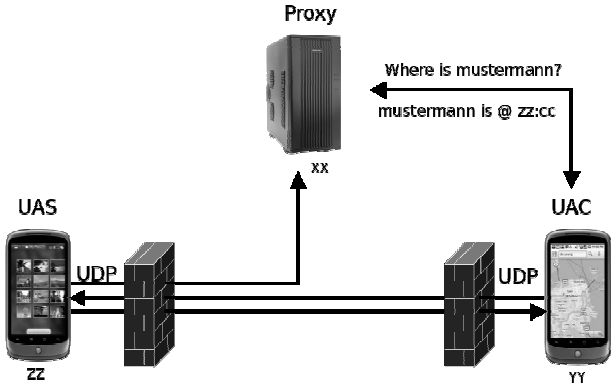


Bild 3 Umgehung Full Cone NAT

Mit Symmetric NAT funktioniert diese Lösung nicht, da die zugewiesene Adresse, die durch den STUN-Server in Erfahrung gebracht wird, nicht der zugewiesenen Adresse, die für die Kommunikation mit einem Peer genutzt wird, entspricht.

Um auch durch ein Symmetric NAT eine Kommunikation zu ermöglichen, muss das STUN-Protokoll erweitert werden. Diese Erweiterung wird in [5] unter dem Namen TURN (Traversal Using Relay NAT) spezifiziert. TURN hat gegenüber dem STUN-Protokoll einen gravierenden Nachteil: Um eine Verbindung zwischen zwei Teilnehmern, die von einem Symmetric NAT getrennt sind, aufzubauen, nutzt TURN einen Relay-Server, über den die Verbindung geführt wird. Die Rolle des Relays übernehmen in der entwickelten Systemarchitektur die Proxies. Der Registrar kann diese Aufgabe nicht übernehmen, da durch das erhöhte Datenaufkommen ein Flaschenhals entstehen könnte. Die Auslagerung der Relay-Funktionalität auf die Proxies vermindert diese Problematik.

3.1 Verbindungsaufbau zwischen Peers

Wie gezeigt, ist der Verbindungsaufbau und Datenaustausch zwischen zwei Peers abhängig davon, ob ein NAT oder eine Firewall eingesetzt wird. Sollte ein NAT-Device eingesetzt werden, ist es ebenso von Bedeutung, um welchen NAT Typen es sich handelt.

Bild 4 auf der folgenden Seite beschreibt den Verbindungsaufbau und anschließenden Datenaustausch

zwischen zwei User Agents. Nachdem UA B wie zuvor beschrieben Proxy A bestimmt hat, sendet er eine Verbindungsanfrage an Proxy A. Diese Verbindungsanfrage beinhaltet u.a. die temporäre Adresse von UA B, Proxy B sowie den NAT Typ (kein NAT, Firewall, Port Restricted NAT, Symmetric NAT, ...), hinter dem sich UA B befindet. Proxy A leitet die Verbindungsanfrage an UA A weiter. UA A kennt den eigenen NAT Typ, hinter dem er sich befindet, und kann anhand der involvierten NAT Typen nun entscheiden, welcher Kommunikationsweg genutzt werden kann. Dabei wird im Allgemeinen zwischen folgenden drei Fällen unterschieden:

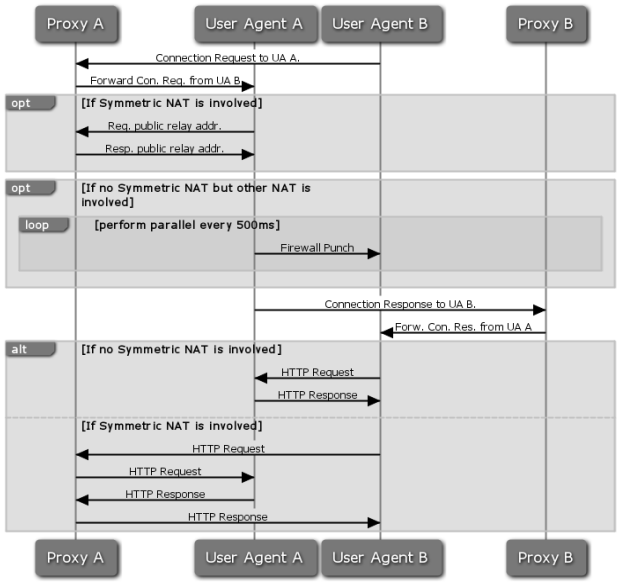


Bild 4 Verbindungsaufbau mit Proxies

1. Befinden sich beide User Agents im selben Netz, können sie direkt miteinander kommunizieren. Diese Information wird vom Registrar geliefert.

2. Ist mindestens auf einer Seite ein NAT bzw. eine Firewall (mit Ausnahme von Symmetric NAT) involviert, so kann UA A mit dem Senden eines Pakets an die temporäre Adresse von UA B einen Port für eingehende Nachrichten in seiner Firewall öffnen. Über diesen Port kann UA B eine Dienstanfrage senden.

3. Ist ein Symmetric NAT involviert, muss ein Relay für den Austausch der Daten eingesetzt werden. UA A allokiert in diesem Fall einen Socket an Proxy A, über den der Verkehr geleitet wird.

UA A teilt UA B den ausgewählten Kommunikationsweg mit, indem er die Antwort auf den Verbindungswunsch an Proxy B sendet. UA B kann nun Dienstanfragen an UA A stellen.

4 Softwarearchitektur

Es wird eine Softwarearchitektur benötigt, die es ermöglicht, komplexe Anwendungen für heterogene Smartphone-Softwarestacks und HTTP-basierte Clients (Browser) einfach zu realisieren.

Der Softwarestack der auf den Smartphones zum Einsatz kommt ist sehr Hersteller abhängig. Das soll in folgender Auflistung verdeutlicht werden. Aufgelistet ist eine Auswahl der derzeit führenden Smartphone Hersteller und den von ihnen verwendete Softwarestacks:

- Nokia - Symbian, maemo (MeeGo)-C/C++, Java ME
- Apple - iPhone OS - Objective C
- HTC - Android - Java
- Palm - WebOS - JavaScript
- BlackBerry - BlackBerry OS - Java ME

Eines haben alle diese Anbieter gemein, jeder hat eine eigene Softwareplattform für seine Geräte. Aus diesem Grund ist ein weiterer Kernpunkt der VMA- Softwarearchitektur eine implementationsunabhängige Schnittstelle zu definieren (s. Auch [2]).

Aufgrund der großen Varianz der auf Smartphoneplattformen eingesetzten Technologien, wurde das REST-Prinzip gewählt um die Schnittstellen zwischen den beteiligten Systemen zu definieren. REST (Representational State Transfer) [1], [6] ist ein Architekturstil, der beschreibt, wie bestehende Technologien, insbesondere HTTP, effizient eingesetzt werden können um verteilte Anwendungen zu implementieren. **Bild 5** skizziert diese Architektur:

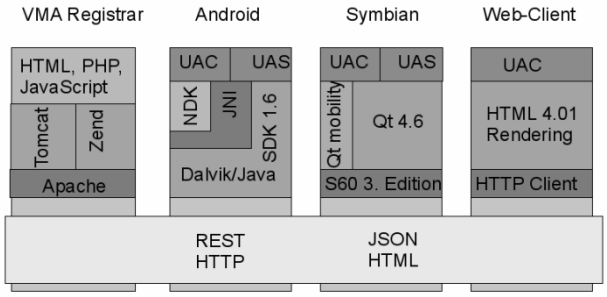


Bild 5 Softwarearchitektur

Im VMA-Projekt hat REST gegenüber SOAP, das sonst häufig zur Implementation verteilter Anwendungen eingesetzt wird, die im Folgenden geschilderten Vorteile [3].

4.1 Unabhängigkeit von Clientsoftware

Die Verwendung von SOAP erfordert für jeden Dienst die Implementierung eines Clients, der XML im SOAP-Kontext interpretieren kann. Wird REST eingesetzt, muss der Client lediglich HTTP-Anweisungen ausführen können, da alle Ressourcen eines REST-konformen Dienstes eine einheitliche Schnittstelle bieten. Definiert wird diese Schnittstelle durch den HTTP-Standard. Dies bedeutet, dass jedes Gerät, welches über einen HTTP-Client verfügt, ein potenzieller Client für die Nutzung eines VMA-Dienstes ist. Da praktisch jedes Internet-fähige Gerät mit einem Browser ausgestattet ist, gewinnt man durch den Einsatz von REST eine große Menge an potenziellen Dienstnutzern. Ebenso bedeutet dies, dass ein und derselbe Dienst einerseits von einer dedizierten Clientanwendung auf dem Smartphone, anderseits von einem beliebigen Endgerät, welches über einen Browser verfügt, genutzt werden kann. **Bild 6** veranschaulicht diesen Vorteil:

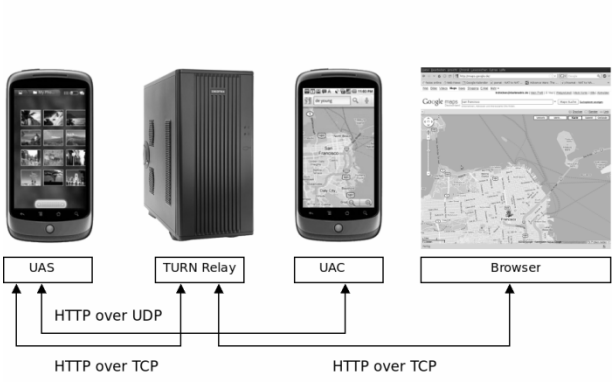


Bild 6 REST Clients

5 Fazit

Durch die VMA-Netzinfrastruktur und den VMA-Softwarestack kann eine Serviceplattform in einem heterogenen Netz-Umfeld, wie es bei Smartphones der Fall ist, angeboten werden. Festzuhalten ist hierbei, dass der Aufwand, der für eine P2P-Kommunikation betrieben werden muss, höher ist als der Aufwand für eine Kommunikation über ein Relay. Der Grund hierfür ist, dass es gilt HTTP über UDP anzubieten. Dabei wird nicht nur eine zusätzliche Schicht zwischen Anwendung und Socket gezogen, es wird zudem eine verlustfreie Kommunikation, wie sie TCP bietet, auch für das UDP-Protokoll geboten. Die verlustfreie Übertragung über UDP wird anlehnend an RUDP [7] realisiert.

6 Literatur

[1] Architectural Styles and the Design of Network-based Software Architectures. Dissertation, University of California, Irvine, 2000

[2] A. Gillert: VMA Mobile Systemarchitektur, Seminarbericht VMA Workshop, FH Köln, 2010.

[3] M. Hüffmeyer: SOAP und REST, Seminarbericht VMA Workshop, FH Köln, 2010.

[4] IETF STUN – Simple Traversal of UDP through NAT. http://tools.ietf.org/html/rfc5389

[5] IETF TURN - Traversal Using Relays around NAT. http://tools.ietf.org/html/draft-ietf-behave-turn-16

[6] S. Tilkov: REST und HTTP, Einsatz der Architektur des Web für Integrationsszenarien, dpunkt.verlag, 2009

[7] IETF RUDP - Reliable UDP Protocol. http://www.ietf.org/proceedings/44/I-D/draft-ietf-sigtran-reliable-udp-00.txt

[8] IETF SIP- Session Initiation Protocol. http://www.apps.ietf.org/rfc/rfc3261.html

Autor

Dipl.-Ing. (FH) Anton Gillert, studiert im Masterstudiengang „Technische Informatik“ der Fachhochschule Köln.

Kontakt: anton.gillert@smail.fh-koeln.de

REST-konforme Web Services im Forschungsschwerpunkt “Verteilte und mobile Applikationen”

Marc Hüffmeyer

Abstract

In diesem Beitrag wird der Einsatz von REST im Forschungsschwerpunkt „Verteilte und mobile Applikationen“ der Fachhochschule Köln präsentiert. Ein Teilgebiet des Forschungsschwerpunkts befasst sich mit der Entwicklung einer Dienstplattform für mobile Endgeräte. Diese Dienstplattform soll eine einfache Erstellung von multimedialen und ortsbezogenen Diensten ermöglichen. Elementar für die Entwicklung einer solchen Dienstplattform ist die Erstellung einer Architektur die ressourcensparende Clientfunktionen, größtmögliche Kompatibilität zu Diensten in heterogenen Netzen, Unterstützung der Kooperation von verschiedenen Hard- und Softwareplattformen sowie entsprechenden Portierungsmöglichkeiten ermöglicht. REST erfüllt diese Anforderungen in besonderem Maße.

1 Einleitung

Verteilte und mobile Anwendungen bekommen eine immer größer werdende Bedeutung in der Informationstechnologie. An der Fachhochschule Köln wurde deshalb der Forschungsschwerpunkt „Verteilte und mobile Applikationen“ (FSP VMA) eingerichtet. Unter anderen befasst sich der FSP VMA mit der plattformübergreifenden Programmierung von Diensten für Smartphones. Von besonderem Interesse sind hier ortsbezogene Dienste sowie Dienste, die multimediale Daten verarbeiten. Neben der Systemarchitektur, welche die Kommunikation zwischen den Endgeräten ermöglicht, ist eine effiziente Softwarearchitektur nötig, um einen schnellen, zuverlässigen und ressourcensparenden Datenaustausch zu gewährleisten. Um diese Ziele zu erreichen bietet sich der Einsatz von REST an. Im Folgenden werden die entscheidenden Prinzipien von REST dargestellt.

2 Ressourcenorientierte Architekturen

Eine ressourcenorientierte Architektur verfolgt ähnliche Ziele wie eine serviceorientierte Architektur. Redundanz in mehreren Systemen soll vermieden werden, stattdessen sollen Informationen an zentraler Stelle verfügbar gemacht werden. Der Hauptunterschied zwischen einer serviceorientierten Architektur und einer ressourcenorientierten Architektur besteht darin, dass nicht einzelne Dienste als Services ausgelagert werden, sondern dass Ressourcen identifiziert werden, die an zentraler Stelle angefragt bzw. bearbeitet werden können. Die Funktionalität einer Anwendung wird nicht auf Services abgebildet, sondern auf Ressourcen. Die Funktionen eines Services werden in eigene adressierbare Ressourcen zerlegt.

Zudem gilt für ressourcenorientierte Architekturen das Prinzip der einheitlichen Schnittstelle. Das bedeutet, dass für die Bearbeitung von Ressourcen ein festgelegter, einheitlicher Satz an Methoden verfügbar ist. Als Folge dessen können sämtliche Ressourcen gleich behandelt werden.

Ein weiterer Unterschied besteht in den Komponenten der Architektur. In einer ressourcenorientierten Architektur wird kein Verzeichnisdienst benötigt. Eine Ressource kann adressiert werden und selbst wiederum auf andere Ressourcen verweisen. So ist es möglich, von Ressource zu Ressource zu navigieren.

Ein klassisches Beispiel für die Verwendung einer ressourcenorientierten Architektur stellt die Bestellabwicklung eines Onlineversands dar. Ressourcen könnten hier Kunden, Artikel und Warenkörbe sein. Mit dem gleichen Set von Bearbeitungsmöglichkeiten (neu erstellen, ändern, löschen) können hier die notwendigen Funktionen abgewickelt werden. Ein Warenkorb kann auf einen Kunden und die enthaltenen Artikel verweisen.

3 REST

Der Begriff REST wurde erstmals in der Dissertation von Roy Thomas Fielding[1] geprägt. REST ist ein Architekturstil, der beschreibt wie verteilte Anwendungen durch bereits bestehende Standards (URI, HTTP, XML, …) umgesetzt werden können. Eine Standardisierung für REST gibt es nicht. REST ist die Abkürzung für Representational State Transfer. Warum dieser Begriff gewählt wurde, lässt sich an folgendem Beispiel gut erkennen: Ein Browser fordert eine Seite (eine Ressource) über eine URL an. Der Server liefert daraufhin ein HTML-Dokument (eine Repräsentation). Das HTML-Dokument kann wiederum Links enthalten, die auf andere Seiten verweisen. Folgt der Client diesen Links, so

verändert er seinen Status. Er macht also einen Transfer seines Status. Bestimmt wird der Transfer durch eine Repräsentation.

3.1 Ressourcen und Repräsentationen

Zentrale Idee von REST ist das Konzept von Ressourcen und deren Repräsentationen. Eine Ressource ist ein Objekt, welches eindeutig identifizierbar ist. Jede Ressource kann ein oder mehrere Repräsentationen besitzen. Beispielsweise kann man eine Person als Ressource identifizieren. Die Repräsentationen einer Person könnten ein Bild, der Personalausweis oder auch ein HTML-Dokument sein. Im Kontext der verteilten Anwendungen stellen Ressourcen die Kommunikationsgegenstände zwischen Client und Server dar. Allerdings wird der Client nie die Ressource zu sehen bekommen, sondern immer nur eine Repräsentation der Ressource.

3.2 Ressourcendesign

Bei der Einteilung von Ressourcen und Repräsentationen stellt sich nun die Frage, wann etwas eine Ressource ist und wann ein Repräsentation. Im obigen Beispiel könnte man das Bild und den Personalausweis einer Person ebenso als jeweils eigene Ressource betrachten, da sie verschiedene Informationen tragen. Der Übergang zwischen einer Ressource mit mehreren Repräsentation einerseits und mehreren Ressourcen andererseits ist fließend. Was in einer verteilten Anwendung als Ressource identifiziert wird, bleibt dem Entwickler überlassen. Generell lässt sich jedoch sagen, dass die Einteilung in eine große Anzahl von Ressourcen vorteilhaft ist. Dies ergibt sich aus der Identifizierbarkeit der Ressourcen, da diese gezielt ansprechbar sind.

3.3 Ressourcenidentifikation

Die Adressierbarkeit von Ressourcen ist elementarer Bestandteil einer ressourcenorientierten Architektur. Im Web wird ein einheitlicher Standard zur Identifikation und Adressierung von Ressourcen verwendet, der Uniform Resource Identifier (URI). Über eine URI wird genau eine Ressource angesprochen. Der Umkehrschluss gilt jedoch nicht, d.h. nicht jede Ressource ist mit genau einer URI ansprechbar. Ressourcen können auch über mehrere Adressen angesprochen werden.

3.4 Zustandslosigkeit von REST

Ein weiterer Kernpunkt für die Umsetzung von REST ist die Zustandslosigkeit. In einer ressourcenorientierten Architektur werden keine Zustände gespeichert. Das bedeutet, dass ein Client seinen Zustand immer selbst verwalten muss bzw. die Abfolge der Ressourcenzugriffe selbst bestimmen kann. REST ist sehr eng mit HTTP verknüpft. Die Zustandslosigkeit von REST ergibt sich bereits durch die Zustandslosigkeit von HTTP.

4 REST und HTTP

Wesentlicher Bestandteil in einer ressourcenorientierten Architektur sind einheitliche Schnittstellen der Ressourcen. Das HTTP-Protokoll definiert eine Reihe von Standardmethoden für die Bearbeitung von Ressourcen und ist deshalb von besonderer Bedeutung für REST. Für die Bearbeitung von Ressourcen sind hier insbesondere die vier Methoden GET, POST, PUT und DELETE des HTTP-Standards von Interesse. Mit diesen vier Methoden lässt sich bereits jegliche Kommunikation zwischen Client und Server realisieren. An dieser Stelle muss man versuchen, sich von der objektorientierten Programmierung oder RPCs zu lösen. Vielmehr ist ein Vergleich mit SQL interessant. SQL bietet die Operationen SELECT, INSERT, UPDATE und DELETE zur Verwaltung von Datensätzen. Für die Verwaltung von Datensätzen hat sich SQL mit seinen vier Methoden als eine hervorragende Möglichkeit herausgestellt. Die Vorstellung, jede Datenbank mit einem eigens für diese Datenbank definierten Set von Methoden zu verwalten wirkt unübersichtlich. Ebenso muss das Verwalten von Ressourcen nicht mit einem eigenen Set von Methoden gemacht werden, sondern kann einheitlich über die HTTP-Methoden geschehen. Deshalb werden im Folgenden die oben genannten HTTP-Methoden detaillierter erläutert und deren Bedeutung für REST dargestellt.

4.1 HTTP-GET

Mit einem HTTP-GET können Repräsentationen von Ressourcen angefragt werden. Das HTTP-GET stellt üblicherweise die meist genutzte Methode eines REST-konformen Service dar, da es für den lesenden Zugriff auf eine Ressource verwendet wird. GET stellt in der SQL-Analogie das SELECT dar, welches für den lesenden Zugriff auf Datenbanken verantwortlich ist. Das HTTP-GET ist eine idempotente Methode. Dies bedeutet, dass für den Fall eines unbeantworteten Requests die Operation gefahrlos erneut ausgeführt werden kann.

```
GET /ressourcen/1234 HTTP/1.1
Accept: image/gif, image/jpeg, */*
Host: www.example.org
```

Listing 1: HTTP-GET

Listing 1 zeigt einen HTTP-GET-Request. Es wird die Ressource unter http://www.example.org/ressourcen/1234 angefragt. Mit dem `accept`-Header des HTTP-Request

kann der Client angeben, welche Repräsentationen er versteht bzw. bevorzugt.

4.2 HTTP-POST

Der HTTP-POST dient zum Anlegen einer neuen Ressource. Er kann daher mit dem SQL-INSERT verglichen werden. Wird ein HTTP-POST verwendet, so wird vom Client keine genaue URI spezifiziert, d.h. der Server legt fest unter welcher Adresse die neu angelegte Ressource zu finden ist. Ein HTTP-POST ist keine idempotente Operation. Ein mehrfaches Versenden des selben HTTP-POST führt dazu, dass eine Ressource auch mehrfach angelegt wird und jeweils auch eine neue URI diese Ressource identifiziert. Der Server wird über den Content-Type-Header informiert, in welchem Format die Ressource vorliegt. Die Ressource selbst wird im Body des HTTP-POST transportiert.

```
POST /ressourcen/ HTTP/1.1
Host: www.example.org
Content-Type: application/xml

<ressource>
        …
</ressource>
```

Listing 2: HTTP-POST

Listing 2 zeigt einen HTTP-POST, der das Anlegen einer Ressource in XML-Form veranlasst. Der Server beantwortet den HTTP-Request für den Fall, dass die Ressource angelegt werden konnte, mit einem HTTP-Response, welcher den HTTP-Header `location` enthalten muss. Der `location`-Header gibt die Adresse an, unter der die angelegte Ressource zu finden ist.

```
HTTP/1.1 201 Created
Location: http://www.example.org/ressourcen/1234
```

Listing 3: HTTP-Response

4.3 HTTP-PUT

Mit einem HTTP-PUT kann eine Ressource aktualisiert werden. Existiert die Ressource noch nicht, wird sie neu angelegt. Anders als beim HTTP-POST wird hier eine genaue Adresse angegeben. Dies ist einerseits nötig, um die zu aktualisierende Ressource zu identifizieren, kann andererseits aber auch genutzt werden, um eine Ressource gezielt unter einer bestimmten Adresse anzulegen. Die HTTP-PUT-Methode ist idempotent und kann mit dem SQL-UPDATE verglichen werden.

```
PUT /ressourcen/1234 HTTP/1.1
Host: www.example.org
Content-Type: application/xml

<ressource>
        …
</ressource>
```

Listing 4: HTTP-PUT

4.4 HTTP-DELETE

Das HTTP-DELETE kann zum Löschen von Ressourcen verwendet werden. Es entspricht seinem Namensvetter, dem SQL-DELETE. Ein HTTP-DELETE ist wie GET und PUT eine idempotente Methode. Existiert eine Ressource nicht mehr, so hat eine erneute Löschanweisung natürlich keinen Einfluß auf diese Ressource. Ebenso muss wie bei GET und PUT eine genaue Adresse angegeben werden, um die zu löschende Ressource zu identifizieren.

```
DELETE /ressourcen/1234 HTTP/1.1
Host: www.example.org
```

Listing 5: HTTP-DELETE

4.5 URI-Design

Wie bereits zu Beginn dieses Kapitels erwähnt ist die Adressierbarkeit von Ressourcen ein Merkmal einer ressourcenorientierten Architektur. Deshalb muss bei der Entwicklung das Design der Adressierung Beachtung finden. Eine grundsätzliche Idee beim Design der URIs ist es, Substantive anstelle von Verben zu verwenden. Die Verben, die eine Operation ausmachen, werden durch den HTTP-Standard bestimmt, nicht durch die URI.

```
http://www.example.org/findPerson?id=1234
http://www.example.org/person/1234
```

Listing 6: Identifikationsregeln

Listing 6 zeigt zwei URIs. Die Erste ist nicht REST-konform. Hier wird ein Verb in der URI abgebildet, welches eine Aktion suggeriert. Dem „Funktionsaufruf“ werden hier noch weitere Informationen in Form eines Query-Parameters übergeben. Mit dieser Art der Adressierung wird das Prinzip der einheitlichen Schnittstelle missachtet. Die korrekte Abbildung zeigt die zweite URI. Hier werden lediglich Substantive verwendet. Die ausgeführte Operation wird durch das HTTP-Protokoll bestimmt.

4.6 Gefahren von REST

In der Abbildung von Anwendungsfunktionalitäten auf nicht REST-konforme URIs liegt die größte Gefahr beim Implementieren von REST-Services. Würde der „Funktionsaufruf“ aus **Listing 6** nicht findPerson, sondern deletePerson heißen, so könnte ein Mensch vermutlich noch über den Kontext erkennen, welche Konsequenzen der Aufruf dieser URI hätte. Eine

Suchmaschine beispielsweise kann dies aber nicht erkennen. Sie geht davon aus, dass ein HTTP-GET einen idempotenten, lesenden Zugriff auslöst und nicht das Löschen einer Ressource verursacht.

4.7 Query-Parameter in REST

Aus den oben genannten Beispielen könnte der Schluss gezogen werden, dass Query-Parameter nicht für den Einsatz in REST geeignet sind. Dies ist jedoch nicht korrekt. REST macht keinerlei Einschränkungen für den Einsatz von Query-Parametern. Es gibt durchaus Situationen, in denen der Einsatz von Query-Parametern sehr hilfreich ist und durchaus REST- konform geschehen kann.

```
http://www.example.org/person/?country=Germany
```

Listing 7: Einsatz von Query-Parametern

Listing 7 zeigt eine URI, die eine Liste von Ressourcen definiert, nämlich genau die Personen, die in Deutschland wohnen. REST-konform können hier die Methoden GET, PUT und DELETE angewendet werden. POST kann in diesem Fall nicht angewendet werden, da es zum Anlegen einer neuen Ressource dient und keine exakte Adresse spezifiziert. Query-Parameter können also in einer ressourcenorientierten Architektur als Filter auf eine Menge von Ressourcen angewendet werden.

4.8 Vor- und Nachteile von REST

Der große Vorteil von REST ist die Unabhängigkeit des Services vom Client. Einzige Bedingung für die Nutzung eines Dienstes ist lediglich der Einsatz mit HTTP. Insbesondere ermöglicht eine ressourcenorientierte Architektur die Verwendung eines Browsers als Client. Dies stellt einen enormen Gewinn dar, da praktisch jedes internetfähige Endgerät mit einem Browser ausgestattet ist. Damit ist eine große Menge von potenziellen Clients bereits gegeben. Einzige Einschränkung bei der Verwendung eines Browsers ist die Repräsentation der Ressourcen in HTML.

Des Weiteren können REST-konforme Services sehr einfach skaliert werden. Durch das Abbilden auf Ressourcen und das Ausliefern von Repräsentationen haben Übergabeparameter von Funktionen einen deutlich kleineren Einfluß auf die Funktionalität der Anwendung. Das Prinzip der einheitlichen Schnittstelle ermöglicht es, den Service beliebig zu erweitern, ohne die bestehende Funktionalität zu beeinflussen oder gar zu verändern.

Der große Nachteil bei der Umsetzung von REST-konformen Services ist die fehlende Standardisierung. Durch das Fehlen eines Standards wird REST oft falsch verstanden. Entwickler setzen die ressourcenorienterte Architektur nicht korrekt um, sondern verwenden einen Mix aus ressourcen- und serviceorientierter Architektur. In Folge dessen wird mehr Schaden als Nutzen angerichtet.

Eine weitere Schwierigkeit bei der Verwendung von REST ergibt sich bei der Abbildung der Anwendungsfunktionalität auf Ressourcen. Auch wenn REST richtig verstanden wurde, ist das Abbilden, zumindest am Beginn der Arbeit mit REST, relativ schwierig. Das Umsetzen von Web Services mit REST bedeutet für einen Entwickler möglicherweise eine große Umstellung in der Denkweise.

5 WADL

Eine Beschreibung eines REST-konformen Services kann mit WADL erfolgen. WADL steht für Web Application Description Language und wurde von Sun Mitarbeiter Marc Hadley entworfen, welcher einer der beiden Leiter bei der JAX-RS-Arbeitsgruppe ist (JAX-RS ist die Java Unterstützung von REST). Die Beschreibung eines REST-konformen Service erfolgt in WADL mittels XML.

Grundsätzlich kann man sagen, dass eine Beschreibung eines REST-konformen Service nicht den gleichen Stellenwert hat wie beispielsweise die Beschreibung eines SOAP basierten Web Service. Dies ergibt sich durch die Tatsache, dass die Nachrichten eines SOAP basierten Web Service in der zugehörigen Beschreibung definiert werden, während die Nachrichten eines RESTful Service im wesentlichen durch den HTTP-Standard bestimmt sind.

6 Einsatz von REST im Forschungsschwerpunkt „Verteilte und mobile Applikationen“

Ziel des VMA-Projektes ist es auf verschiedenen Smartphone-Plattformen einheitliche Dienste anzubieten. Ein wesentlicher Unterschied zu einer reinen ressourcenorientierten Architektur ist, dass es nicht einen Dienstanbieter und viele Dienstnutzer geben soll, sondern viele Dienstanbieter und viele Dienstnutzer. Das heißt, es soll nicht ein Dienst an zentraler Stelle angeboten werden, sondern viele Mobile Clients sollen den gleichen Dienst anbieten. Es soll also möglich sein, von einem System aus einen Dienst auf einem beliebigen mobilen Endgerät zu nutzen.

Des Weiteren soll die Nutzung eines Dienstes möglichst einfach sein, d.h. ein Dienstnutzer soll möglichst wenig über den Dienst wissen müssen, um ihn zu nutzen. Dies erhöht die Anzahl an potenziellen Dienstnutzern. Kann man einen Dienst beispielsweise über einen Browser nutzen, so existiert eine wesentlich größere Menge an Dienstnutzern, als wenn man den Dienst nur über ein spezielles Clientprogramm nutzen kann.

7 Beispieldienst „Location Service“

Als praktisches Beispiel zur Erläuterung wie REST im FSP VMA eingesetzt wird, soll ein „Location Service“ dienen. Dieser Service sendet einen HTTP-Request an ein mobiles Endgerät, welches daraufhin mit einem HTTP-Response antwortet, welcher den aktuellen Standort des Geräts enthält. Der aktuelle Standort kann in diesem Beispiel also als Ressource identifiziert werden und mit einem HTTP-GET abgefragt werden. Der HTTP-Request und der HTTP-Response sind in **Listing 8** und **Listing 9** dargestellt. In **Listing 8** wird als bevorzugte Repräsentation application/xml angegeben. Der Client könnte bspw. ein IPhone oder Android Mobilgerät sein, welches die XML-Daten in irgendeiner Form selber darstellen möchte. Ganz analog zu der Anfrage in **Listing 8** könnte aber auch ein beliebiger Browser eine Anfrage an den Service stellen. Dieser würde vermutlich eine Repräsentation in HTML bevorzugen, würde also den Quality-Parameter im Accept-Header genau andersherum belegen. Als Folge dessen könnte das Gerät, welches den Service anbietet, anstelle eines XML-Dokumentes bspw. einen Link zu Google Maps zurück liefern, also eine andere Repräsentation der Ressource.

```
GET /service/location HTTP/1.1
Host: 192.168.1.254

Accept: application/xml; q=0.9, text/html; q=0.1
```

Listing 8: REST Location Request

```
HTTP/1.1 200 OK
Content Type: application/xml
Content Length: 73

<location>
        <latitude>1</latitude>
        <longitude>2</longitude>
</location>
```

Listing 9: REST Location Response

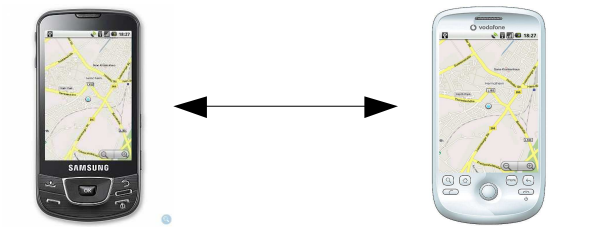


Bild 1 Location Service

8 Literatur

[1] Thomas Roy Fielding: Architectural Styles and the Design of Network-based Software Architectures (http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htmHYPERLINK"http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm"), University of California, Irvine, 2000

Autor

Dipl.-Ing. (FH) Marc Hüffmeyer, studiert im Masterstudiengang „Technische Informatik“ der Fachhochschule Köln.

Kontakt: marc.hueffmeyer@googlemail.com

Präge- und Stempelschrifterkennung auf metallischen Oberflächen

Sebastian Seegert

Abstract

Präge- und Stempelschriften kommen in der Metall produzierenden und verarbeitenden Industrie häufig zum Einsatz. So werden z.B. produzierte Brammen, Vorblöcke, Rundstäbe, etc. durch elektropneumatischen Nadelpräger oder durch einfache Stempelungen mit Identifikationsnummern gekennzeichnet. Bekannte Standardverfahren zur Schrifterkennung sind deshalb ungeeignet. Kernansatz für das optische Erkennungssystem ist die Betrachtung des Schattenwurfes auf der Kenzeichnungsoberfläche, sowie die daraus abgeleitete spezifische Beleuchtung, die in Kombination mit darauf abgestimmten Bildvorverarbeitungsalgorithmen eine deutliche Hervorhebung der Kennzeichnung ermöglicht. Die so erzeugten Ergebnisse können dann anschließend mittels klassischer OCR-Verfahren (Segmentierung, Klassifikation durch neuronale Netze etc.) weiter verarbeitet werden.

1 Einführung

Zur Steuerung von komplexen Produktionsprozessen werden in der Metall produzierenden und –verarbeitenden Industrie wie anderswo leistungsfähige Produktplanungs- und Steuerungskonzepte eingesetzt, die durch entsprechende Logistikkonzepte ergänzt werden. Für die Optimierung der logistischen Prozesse nimmt die Bedeutung von Produktkennzeichnungssystemen immer mehr zu, um die stückweise Verfolgung von Einzelprodukten im Produktionsprozess zu gewährleisten. So werden z.B. produzierte Brammen, Vorblöcke, Rundstäbe, etc. durch elektropneumatischen Nadelpräger oder durch einfache Stempelung mit einer Identifikationsnummer gekennzeichnet. Dies soll Verwechselungen im Produktionsprozess auszuschließen oder sogar ermöglichen Prozesse „live“, in Abhängigkeit von individuell ermittelten Materialgüten, innerhalb der Produktionsanlage zu steuern. Die Kennzeichnung kann im erkalteten sowie im noch glühenden Zustand erfolgen. Jedoch können die oben beschriebenen Kennzeichnungsverfahren häufig zu ungenauen Prägungen bzw. Stempelungen führen. Im weiteren Produktionsprozess ist somit eine eindeutige und fehlerfreie Identifikation nicht möglich.

Vor dem Hintergrund der in der Praxis eingesetzten Kennzeichnungsverfahren entwickelt das Labor für industrielle Bildverarbeitung der Fachhochschule Köln ein optisches Erkennungssystem für Präge- und Stempelschriften auf erkalteten und glühenden metallischen Oberflächen.

Die Qualität der so erzeugten Kennzeichnung ist, bedingt durch die unebene Oberfläche der gestempelten Objekte (erzeugt durch z.B. Sägeriefen, Zunder etc.), oftmals so schlecht, dass diese selbst für den Menschen nicht lesbar sind. Bekannte Standardverfahren zur Schrifterkennung (z.B. auf gedrucktem Papier) sind hierfür deshalb ungeeignet. Es wurden bereits verschiedene Erkennungssysteme vor dem gleichen Hintergrund in [1] und [2] untersucht. Abhilfe schafft die Betrachtung des Schattenwurfes auf der Kenzeichnungsoberfläche, sowie die daraus abgeleitete spezifische Beleuchtung, die in Kombination mit darauf abgestimmten Bildvorverarbeitungsalgorithmen eine deutliche Hervorhebung der Kennzeichnung ermöglicht.

Anschließende Bildverarbeitungsalgorithmen sorgen für eine Lage- und größeninvariante Detektion und Normierung der Kennzeichnung. Die so erzeugten Ergebnisse werden dann mittels klassischer OCR-Verfahren (Segmentierung, Klassifikation durch neuronale Netze etc.) weiter verarbeitet.

2 Lösungsansatz

Da Präge- oder Stempelschriften sich nicht wie eine gedruckte Schrift mit einem spezifischen Farbwert bzw. Grauwert vom Hintergrund unterscheiden lassen, muss ein Verfahrensansatz entwickelt werden, der die geprägte oder gestempelte Schrift deutlich vom Hintergrund abhebt, und so eine Klassifikation der Schriftzeichen ermöglicht. Dies kann beispielsweise durch die Betrachtung des Schattenwurfes in der Riefe der Schrift erreicht werden.

Hierzu wurde im ersten Schritt ein Beleuchtungsprinzip entworfen, das eine passende Ausleuchtung der Schrift und einen möglichst guten Schattenwurf in der Riefe gewährleistet.

In dem nächsten Verarbeitungsschritt werden beide aufgenommenen Bilder jeweils mit Mitteln der digitalen

Bildverarbeitung aufbereitet und zu einem Bild mit Grauwertkanten zusammengefügt, sodass sich die Schrift deutlich von ihrer Umgebung im Bild hervorhebt. Weiter muss das so entstandene Bild für die Segmentierung vorbereitet werden, um eine Lokalisierung des Schriftzugs sowie der Schriftzeichen im Bild zu erreichen.

Nach Vorliegen der Ergebnisse werden die ermittelten Bildausschnitte einer Zusammenhangsanalyse unterzogen und in eine geeignete Datenstruktur aufgenommen.

Aus der Datenstruktur können dann Patternbilder für das Training von neuronalen Netzen erzeugt oder für bereits trainierte neuronale Netze als Eingabevektor für die Klassifikation der Schriftzeichen verwendet werden.

3 Versuchsaufbau und Beleuchtungsprinzip

Für die geforderte Beleuchtung der Oberfläche sind folgende Parameter von Bedeutung:

- der Einfallswinkel α des Lichtes zur Riefenkante
- der Abweichungswinkel β einer Lichtquelle zu einer nicht parallel liegenden Riefenkante

Um einen günstigen Schattenwurf mit dem Beleuchtungsaufbau zu erzielen, muss eine optimale Position der Leuchtmittel ermittelt werden. Hierfür kann die Betrachtung einer idealisierten - rechteckförmigen - Stempel- oder Prägefurche als Beispiel genommen werden, vgl. **Bild 1**.

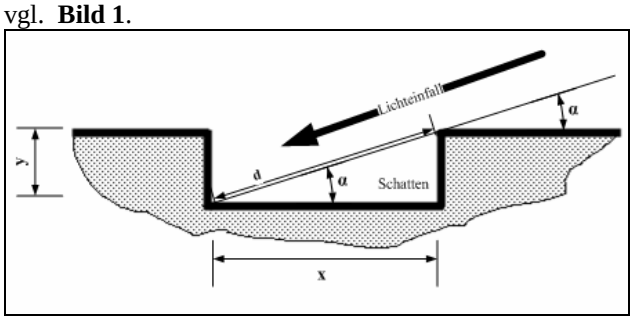


Bild 1 Idealisierte Furche im Profil, zur Darstellung des optimalen Einfallswinkels der Beleuchtung

Der optimale Einfallswinkel α einer zur Kante parallel liegenden Lichtquelle kann durch die Furchenbreite x und Furchentiefe y ermittelt werden. Es ist aus **Bild 1** ersichtlich, dass α <90° sein muss, damit es zu einem Schatten in der Riefe kommt. Ein optimaler Schattenwurf füllt mindestens die gesamte Breite x der Furche. Eine einfache Berechnung erfolgt mit:

$$d = \sqrt{x^2 + y^2},$$

$$\alpha = \arccos\left(\frac{x}{d}\right) = \arccos\left(\frac{x}{\sqrt{x^2 + y^2}}\right)$$

Der Einfluss des Abweichungswinkel β zu einer nicht parallel liegenden Kante soll mit **Bild 2** und **3** verdeutlich werden. Auch die Bedeutung der Orthogonalität der zwei zueinander stehenden Lichtquellen für die Kantendetektion kann hieran erläutert werden.

Bild 2 Frontale Ansicht eines ovalen Objektes senkrecht (links; 0°) und waagerecht (rechts; 90°) ausgeleuchtet. Der weiße Vordergrund entspricht dem Schattenwurf in der Riefe.

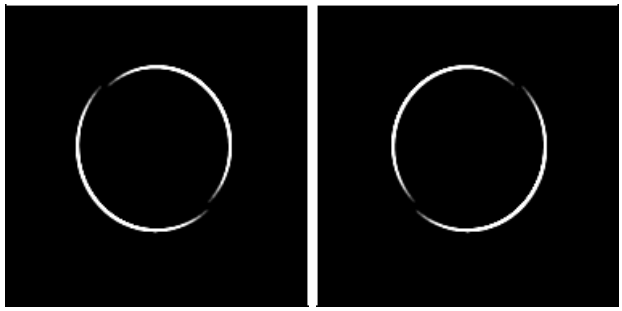


Bild 3 Frontale Ansicht eines ovalen Objektes diagonal (links: 135° und rechts: 45°) ausgeleuchtet. Der weiße Vordergrund entspricht dem Schattenwurf in der Riefe.

In **Bild 2** sowie in **Bild 3** ist zu erkennen, dass bei paralleler Lage der Riefenkanten zur Lichtquelle der Schattenwurf ein Maximum der Fläche aufweist. Bei Abnahme der Parallelität (Lage der Riefenkante zur Lichtquelle) geht die Fläche des Schattens gegen Null. Die so entstehenden „Lücken" – vgl. **Bild 2** und **3** - des Schattens werden durch die spätere Addition beider Bilder wieder ausgeglichen. Die hier dargestellten Beispiele zeigen auch die Rotationsinvarianz des verfolgten Beleuchtungsprinzips. Es ersichtlich, dass die Lage der Riefenkante zum Beleuchtungssystem keinen Einfluss auf den geforderten Schattenwurf hat. Gleiches gilt für eine Rotation der orthogonal zu einander stehenden Lichtquellen.

Für die Bildaufnahme wurden jeweils mit seitlicher und senkrechter Beleuchtung zwei 8-Bit-Grauwertbilder aufgenommen.

Bild 4 Das Beispiel links (rechts) zeigt die Probe mit senkrechter (seitlicher) Beleuchtung

4 Bildvorverarbeitung

Die Bildvorverarbeitung ist ein Kernelement dieser hier beschriebenen Untersuchung. Ziel ist es aus den Bildern mit jeweils abweichender Beleuchtung ein geeignetes Bild für die nachfolgende Segmentierung und Klassifikation zu erzeugen.

Auf beiden Bildern wurde ein Histogrammausgleich durchgeführt. Dieses ermöglicht eine optimale Neuverteilung aller Grauwerte womit die Darstellung des Schattenwurfes im Bild der Riefe verstärkt wird.

In einem weiteren Schritt erfolgt auf dem Bild eine Kantendetektion mit Hilfe eines Kantendetektionsfilters. Bei dem hier verwendeten Kantendetektionsfilter handelt es sich um eine Abwandlung des Prewitt-Operators. Die richtungsabhängige, quadratische und 7 Pixel große Maske wurde mithilfe der Faltung auf beide Bilder angewandt.

Das Ergebnis sind zwei Gradientenbilder, welche die Kanten im Bild als ein Grauwertverlauf von dunkel nach hell, bzw. von hell nach dunkel darstellen. Die Bereiche der größten Intensität sind dort, wo sich die Grauwerte des Originalbildes am stärksten ändern und so die stärksten Kanten darstellt.

Die aus der Faltung resultierenden Fließkommagradientenbilder - dargestellt in Fließkommawerten - können nun in einen positiven und negativen Fließkommawertanteil aufgeteilt werden.

In den erzeugten Gradientenbildern wird die Präge- oder Stempelkante mit einem negativen Fließkommawertanteil dargestellt. Das „Tal“ der Schrift, welches unter der Beleuchtung im Schatten liegt, wird mit einem positiven Fließkommawertanteil gezeigt. Für die weitere Verarbeitung ist nur der jeweilige positive Anteil von Bedeutung.

Beide positiven Fließkommawertbilder, die aus der bis dato erfolgten parallelen Bildvorverarbeitung hervorgehen, werden nun durch eine Addition zu einem Bild zusammengeführt. Das daraus entstehende Bild beinhaltet jetzt alle positiven Fließkommawertanteil beider aufgenommen Bilder. Anschließend wird dies mit einem Gauß-Filter „geglättet“. Das Bildrauschen wird dadurch reduziert und es bleiben größere Bildstrukturen erhalten.

Abgeschlossen wird die Bildvorverarbeitung durch Extraktion der maximalen Grauwerte. **Bild 5** zeigt beispielhaft das oben beschriebene Verfahren während **Bild 6** das Ergebnis der Bildverarbeitung zeigt.

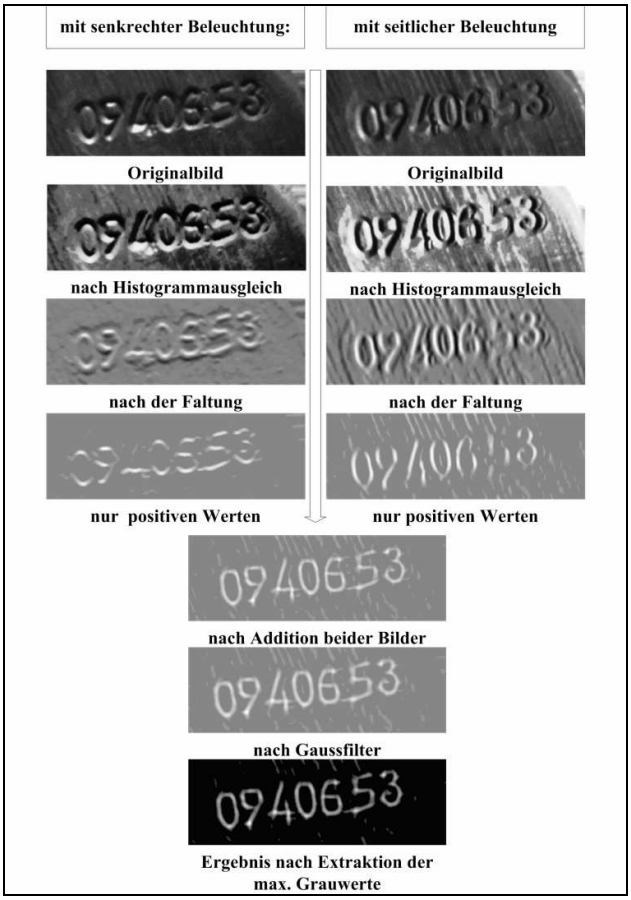


Bild 5 Bsp. für die Zwischenergebnisse der Bildvorverarbeitungskette für die Probe „0940653“

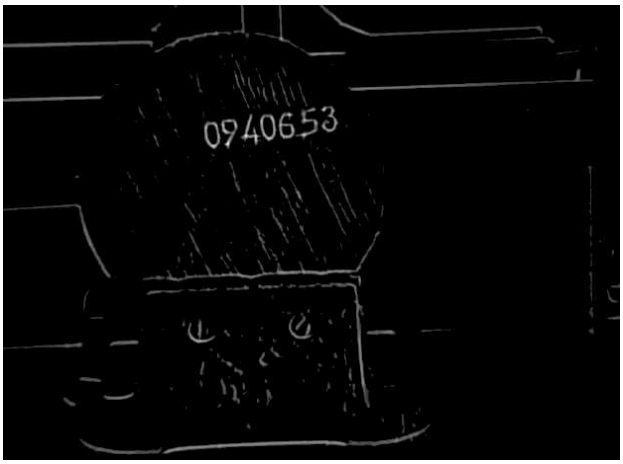


Bild 6 Ergebnis der Bildvorverarbeitung für das

Prägestück „0940653“

5 Lokalisierung und Segmentierung

5.1 Das Regionenbild erzeugen

Aus dem Ergebnisbild der Bildvorverarbeitung wird nun ein Bild mit Regionen erzeugt, das für die Lokalisierung des Schriftzuges benötigt wird. Zunächst erfolgt eine Subtraktion mit einem zuvor aufgenommen Bild, das konstante Bildelemente beinhaltet. Konstante Bildelemente sind immer wieder auftretende Elemente wie Hinter- und Vordergrundobjekte, hier z. B. die Halterung für das Probestück oder die Halterung der Beleuchtung. Durch die Subtraktion erfolgt in einem Schritt die Reduktion der Bildinformation auf das Wesentliche. Als nächstes wird das Grauwertbild binarisiert. Danach erfolgt nun die Anwendung von morphologischen Operatoren. Auf das binarisierte Bild wird zuerst eine Dilatation und anschließend eine Erosion angewandt. Beide verwendeten Strukturelemente sind kreisförmig, um auch hier eine Rotationsinvarianz weiter zu gewährleisten. Das Ergebnisbild zeigt nun weiße zusammenhängende Flächen, die mögliche Bereiche der Schrift im Bild darstellen. Das Ergebnis wird als ein „Regionenbild“ zwischengespeichert.

5.2 Mögliche Region mit Schriftzug finden

Das Auffinden von Regionen im Bild geschieht durch die Anwendung der Methode der bereichsbasierten Zusammenhangsanalyse. Wurde eine Region gefunden, so wird diese auf ihre Größe (Anzahl der Pixel) überprüft. Ist die Region kleiner als ein bestimmter Schwellenwert, ist diese nicht zu berücksichtigen. Ist die Region größer als der Schwellenwert, ist diese verfolgungswürdig und es wird im nächsten Schritt die Orientierung bestimmt.

Die Orientierung einer Region stellt die Richtung der Hauptachse der in der Region beschriebenen Fläche dar. Nachdem der Winkel der Richtung berechnet wurde, wird das Ergebnisbild der Bildvorverarbeitung in die Normallage im Flächenschwerpunkt rotiert. Unter Normallage wird hier die waagerechte Lage der Schrift im Bild verstanden [3].

5.3 Mögliche Schriftsegmente im Bildausschnitt bestimmen

Auf den zuvor ermittelten Bildausschnitt wird ein schwellenwertabhängiger Closing-Filter angewandt. Der Filter reduziert Bildstörungen und verstärkt nochmals die Segmente möglicher Schriftzeichen. Anschließend erfolgt eine Autobinarisierung. Das Auffinden von Regionen geschieht auch hier wieder durch die Anwendung der Methode der bereichsbasierten Zusammenhangsanalyse. Wurde eine Region gefunden, so wird diese auf ihre Größe (Anzahl der Pixel) überprüft. Ist die Region kleiner als eine zuvor definierte Untergrenze oder größer als eine bestimmte Obergrenze, ist sie nicht zu berücksichtigen. Liegt die Region in dem definierten Bereich, wird sie weiterverfolgt. Es kann sich um ein mögliches Schriftzeichensegment handeln und der Bildbereich wird ausgeschnitten, sowie auf eine vorher definierte Größe normiert. Das daraus neu erzeugte normierte Bild wird im Folgenden als Patternbild bezeichnet. Das Patternbild kann nun zur Klassifikation verwendet werden.

6 Klassifikation

Das Klassifizieren der einzelnen Schriftzeichen geschieht durch ein neuronales Netz. Das hier verwendete Netz (Multilayer Perceptron) besteht aus vier Schichten. Die Größe von 35 mal 50 Neuronen des Eingabevektors entspricht der Größe des erzeugten Patternbildes. Die Anzahl der Neuronen in den verdeckten Schichten wurde variiert und in verschiedenen aufgebauten Netzen getestet. Für die Implementierung wurde nach der Auswertung der verschiedenen Netze, ein Netz mit 25 Neuronen in der ersten verdeckten Schicht und 20 Neuronen in der zweiten verdeckten Schicht eingesetzt.

Als Lernalgorithmus wurde der Backpropagation-Algorithmus mit einer sigmoiden Aktivierungsfunktion angewandt [4].

7 Erste Ergebnisse

Erste Ergebnisse dieser Untersuchung sind im Labor entstanden. Es stand somit - naturbedingt - nur eine geringe Anzahl von Proben zur Verfügung. Zudem wurden bewusst nur kritische Proben, mit fehlerhaften Markierungen oder schlechten Schnittkanten für die Untersuchung herangezogen.

Nachfolgend soll nicht auf die Optimierung von neuronalen Netzen eingegangen werden. Im Weiteren soll der Schwerpunkt der Diskussion vor dem Hintergrund einer praxistauglichen Bildvorverarbeitung fortgeführt werden und nur Erkenntnisse und erste Ergebnisse aus den Vorverarbeitungsschritten beleuchtet werden.

7.1 Ergebnis der Bildvorverarbeitung

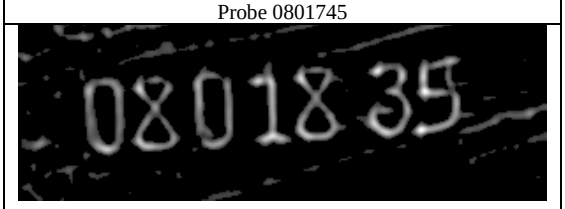
Die Ergebnisse der Bildaufbereitung zeigen deutlich, dass das entwickelte Verfahren sehr Erfolg versprechend ist, um eine anschließende Klassifikation der Schriftzeichen auf gutem Niveau durchführen zu können.

Die dabei eingesetzte Art der Bildaufbereitung erreicht ein deutliches Hervorheben der Prägung vor dem jeweiligen Hintergrund, sowie eine Rotations- und Translationsinvarianz.

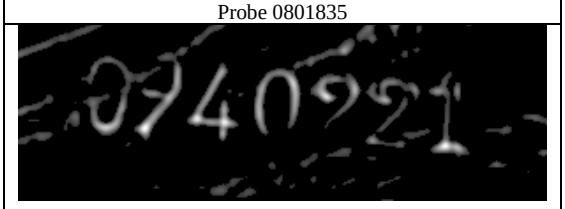
Die Abweichungen der einzelnen Ergebnisse erklären sich durch die nicht ganz homogene Beleuchtung des Prägestückes sowie der nicht hundertprozentig orthogonal zueinander stehenden Lichtquellen. Besonders kommt hinzu, dass die Kantenbildung aufgrund der Furchen auf den Schnittkanten in ihrer Intensität durch die Rotation variiert. Störungen treten so stärker oder weniger im Bild auf. Nachfolgende Ergebnisse - **Tabelle 1** und **2** - der Bildvorverarbeitung von Proben mit einer Lage der Schrift bei 0° und 45° verdeutlichen dies.

Proben mit der Orientierung 0°

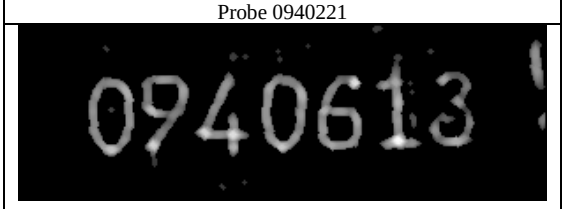
Probe 0801745



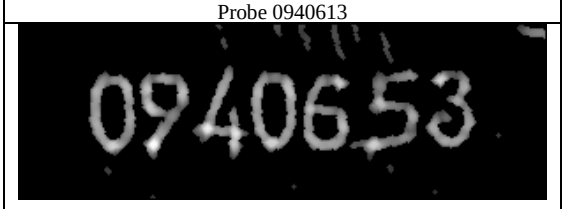
Probe 0801835



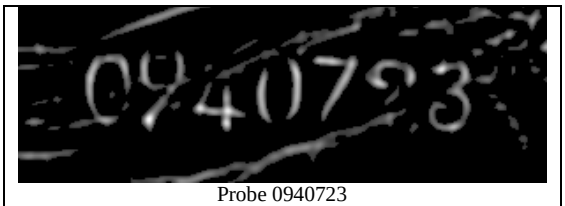
Probe 0940221



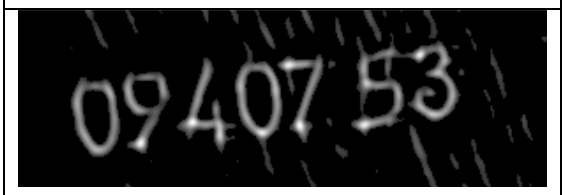
Probe 0940613



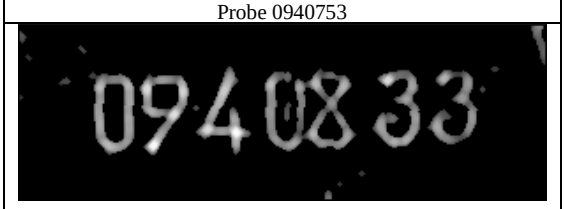
Probe 0940653



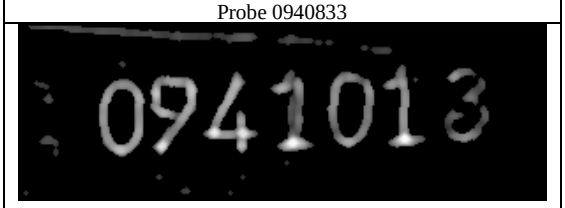
Probe 0940723



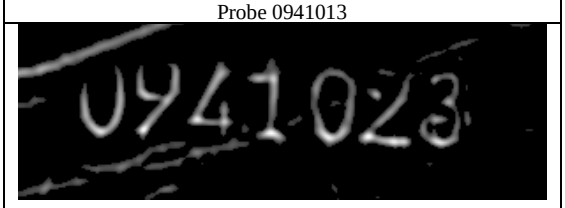
Probe 0940753



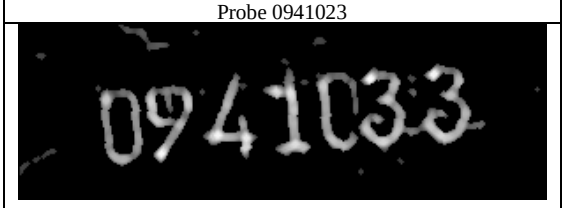
Probe 0940833



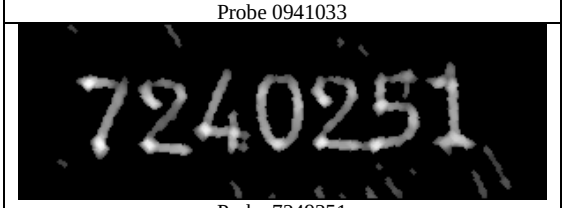
Probe 0941013



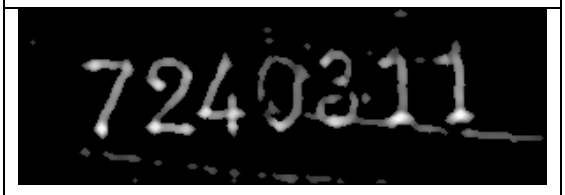
Probe 0941023



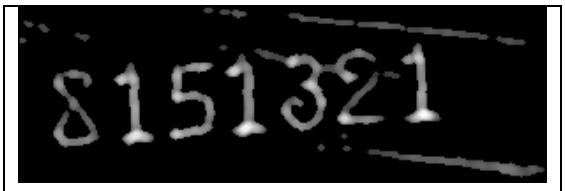
Probe 0941033



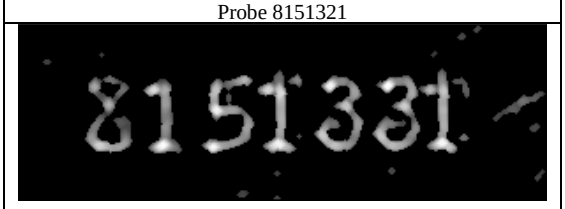
Probe 7249251



Probe 7240311



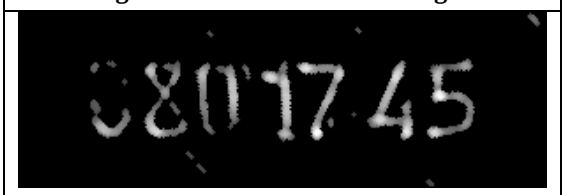
Probe 8151321



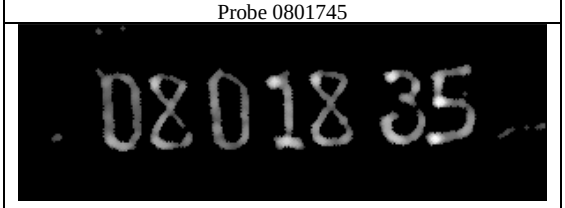
Probe 8151331

Tabelle 1 Ergebnisse der Bildvorverarbeitung aller Proben mit Orientierung der Schrift bei 0°

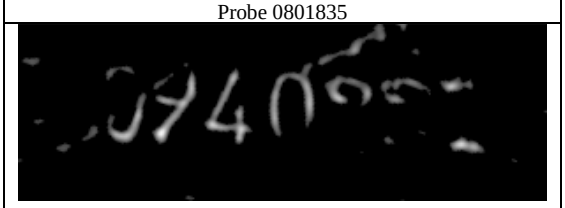
Prägestück mit der Orientierung 45°



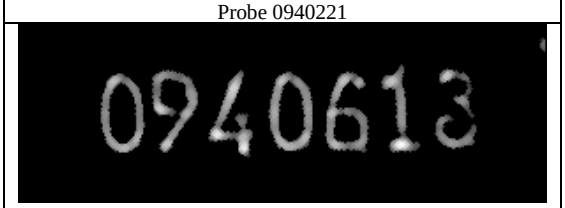
Probe 0801745



Probe 0801835

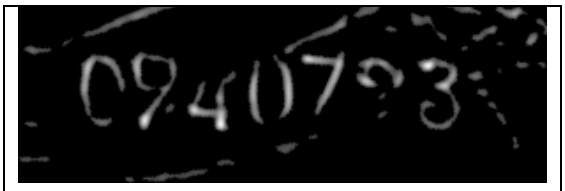


Probe 0940221

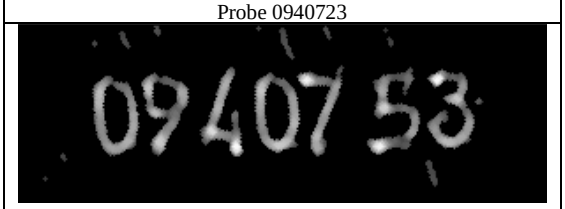


Probe 0940613

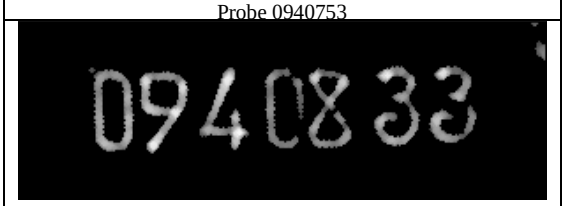
Probe 0940653



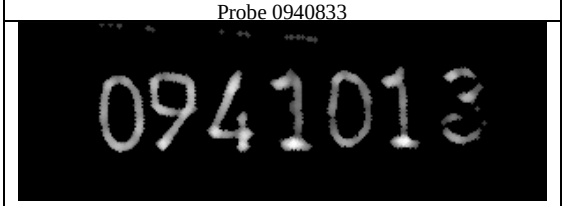
Probe 0940723



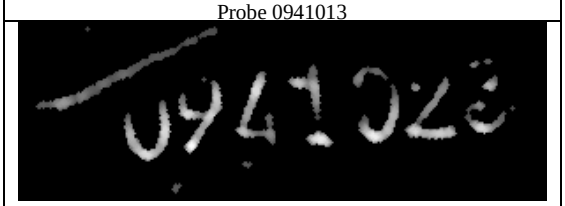
Probe 0940753



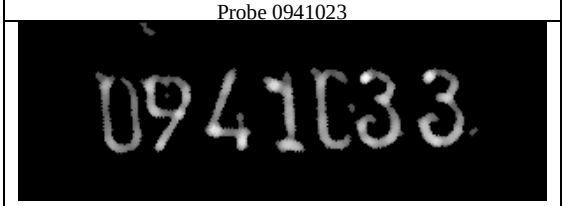
Probe 0940833



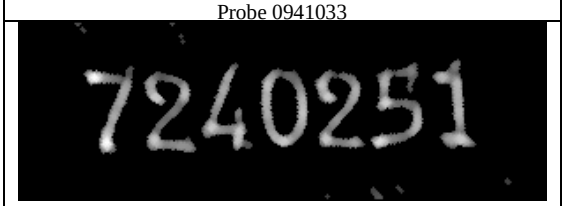
Probe 0941013



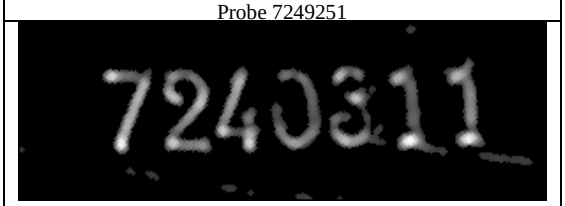
Probe 0941023



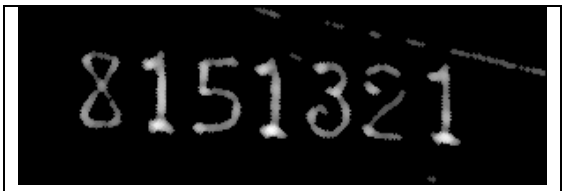
Probe 0941033



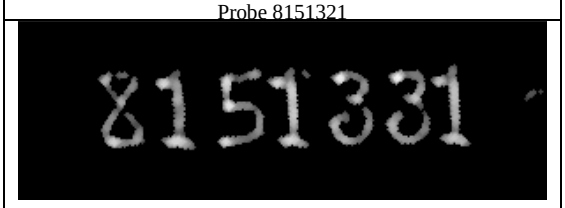
Probe 7249251



Probe 7240311



Probe 8151321



Probe 8151331

Tabelle 2 Ergebnisse der Bildvorverarbeitung aller Proben mit Orientierung der Schrift bei 45°

7.2 Ergebnis der Lokalisierung und Segmentierung

Das hier eingesetzte Verfahren der Bildaufbereitung erreicht ein deutliches Hervorheben der Prägung vor dem Hintergrund und liefert als Ergebnis eine gute Basis für die Segmentierung und anschließende Klassifikation. Allerdings sind dem Verfahren auch Grenzen gesetzt. Ist das Bild stark gestört oder die Prägung fehlerhaft, kann es nur begrenzt segmentiert und klassifiziert werden. **Bild 7** zeigt als ein Beispiel einen stark gestörten Bildausschnitt mit einer fehlerhaften Rotation.

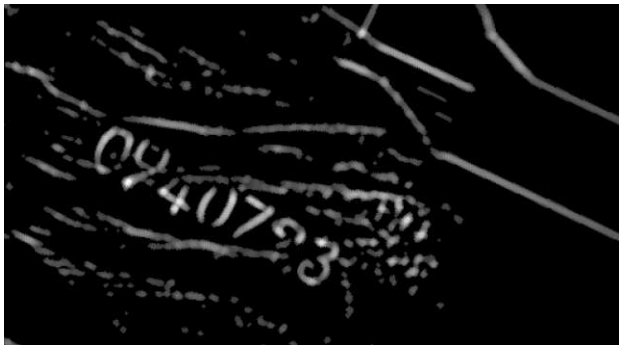


Bild 7 stark gestörter Bildausschnitt mit fehlerhafter Rotation

Störungen im Bild treten dann besonders stark auf, wenn die Oberflächen der Schnittkante tiefe Furchen aufweisen. Die Furchen auf den Schnittkanten ergeben sich beim Schneiden des Stranggusses und dem verwendeten Schneideverfahren (Sägeschnittverfahren oder Brenn-schnittverfahren). Die meisten Furchen traten an den Schnittkanten beim Brennschnittverfahren auf. Sie führten zu einem zusätzlichen Schattenwurf auf der Schnittoberfläche und es ergaben sich unerwünschte zusätzliche Grauwertkanten im Bild. Zudem verursacht die unebene Schnittoberfläche eine Verzerrung der Schrift, die wiederum später zu einer Fehlklassifikation führen kann. Weiter können schlechte Schnittkanten, trotz guter Prägung im Bild, zu einer fehlerhaften Bestimmung der Orientierung und einer falschen Rotation der Prägung in Normallage führen. So kann es dazu kommen, dass die Prägung nicht horizontal im Bild liegt und die erzeugten Patternbilder verzerrte Schriftzeichen aufweisen, die wiederum später fehlerhaft klassifiziert werden. Bei der Auswertung aller Probestücke in Normallage und mit einer Orientierung von ca. 45° kam es in 9 von 30 Fällen zu einer fehlerhaften Rotation. Die meisten Abweichungen nach der Korrektur in die Normallage betrugen jeweils +-5 - 10°.

Eine weitere Problematik ergibt sich, wenn Prägezeichen mit guter Prägequalität, z. B. in der Bildaufbereitung, „zusammenwachsen“ oder Prägezeichen „aufbrechen“.

Bei dem „Zusammenwachsen“ von Prägezeichen, ergeben sich zu große Segmentgrößen. Die Konsequenz ist, dass die Segmentierung wird als nicht verfolgungswürdig eingestuft. Dies geschieht ebenfalls bei „aufgebrochenen“ Prägezeichen, da die einzelnen Segmentteile zu klein sind. Auch in diesen Fällen ist eine Klassifikation nicht möglich.

Tabelle 3 führt für alle Proben die Fehler der Segmentierung und der Rotation auf. Es ist anzumerken, dass die aufgeführten Ergebnisse auf den ersten Blick „dramatisch“ schlecht ausfallen aber diese im Vergleich mit **Tabelle 1** und **2,** sich relativieren.

Bei der Segmentierung liegt der Fehler bei Normallage (0°) bei 26,6% und bei Rotation (45°) bei 27,6%. Dieses Ergebnis zeigt, das der Bildvorverarbeitungsprozess eine gute Rotationsinvarianz aufweist.

Probe	Ergebnis der Bildvorverarbeitung			
	Fehler[1] bei Orientierung 0°		Fehler[1] bei Orientierung 45°	
	Segmentierung	Rotation[2]	Segmentierung	Rotation[2]
0801745	0	1	3	0
0801835	0	1	1	0
0940221	3	1	3	1
0940613	0	0	1	0
0940653	0	0	1	0
0940723	7	1	7	1
0940753	2	0	0	0
0940833	2	0	2	0
0941013	1	0	2	0
0941023	2	1	4	1
0941033	2	1	0	0
7240251	2	0	0	0
7240311	2	1	1	0
8151321	2	1	2	1
8151331	0	0	2	0
Summe:	28	8	29	4
in % :	26,6	53,3	27,6	26,6

[1]: nicht segmentierte Ziffern

[2]: 1: fehlerhafte Rotation; 0: kein Fehler bei Rotation

Tabelle 3 Ergebnisse der Segmentierung und Rotation

Der Fehler der Rotation bei Normallage (0°) ist 53,3% und bei Rotation (45°) 26,6%. Die starke Abweichung entsteht – wie bereits beschrieben – durch die unebenen und zerfurchten Oberflächen, welche zu einem zusätzlichen Schattenwurf auf der Schnittoberfläche führen und die Berechnung des Rotationswinkels erheblich stören.

8 Ausblick

Die ersten Ergebnisse zeigen, dass die Optimierung von Algorithmen zur Lagebestimmung und die Rotation weiter verfolgt werden sollten, ebenso wie die Beschleunigung zum Auffinden von einzelnen Schriftzeichen durch die Einbindung von Kontextwissen.

Der Hauptschwerpunkt weiterer Untersuchungen muss aber bei der Optimierung der Segmentierung liegen, sodass „aufgebrochene“ oder „zusammengewachsene“ Schriftzeichen zuverlässig segmentiert werden können.

Parallel sind erste Tests des hier beschreibenden Verfahren bereits bei heißen und sehr hell glühenden Proben bei ca. 1000°C möglich. Vor diesem Hintergrund ist das Zusammenspiel von Kamera-, Filter- und Beleuchtungssystemen von besonderer Bedeutung. Erste Erkenntnisse zeigen aber, dass das hier beschriebene Verfahren auch für glühende metallische Oberflächen mit entsprechenden Bilderfassungssystemen geeignet ist.

9 Förderung und Partner

Das hier beschriebene Projekt wurde durch den Forschungsschwerpunkt "Verteilte und mobile Applikationen" (FSP VMA) und die Firma Hüttenwerke Krupp Mannesmann GmbH (HKM) unterstützt.

Der Forschungsschwerpunkt VMA bündelt im Bereich der Technischen Informatik Forschungsaktivitäten auf den Gebieten Eingebettete und Autonome Systeme, Security Engineering und Testautomatisierung für verteilte und mobile Systeme und Verteilte mobile Dienste in Next Generation Networks. Die Forschungs- und Entwicklungsarbeiten sind im Arbeitsgebiet 2 „Anwendung für eingebettete Systeme und Robotik“ entstanden, welches sich u.a. mit den Themen intelligente Sensoren für industrielle und mobile Anwendungen befasst.

10 Literatur

[1] Cai, J.; Zhang, G.; Zhou, Z.: Design of On-line Automatic Vision Inspection System for Steel Billet Characters, Chinese Journal of Sensors and Actors, Hangzhou, China, März 2006

[2] Jong-hak, L.; Sang-gug; Soo-joong, K.:Vision Technique for Recognition of Billet Characters in the Steel Plant, 8th Pacific Rim International Conference on Artificial Inteligence, Auchland, New Zealand, 9-13 August 2004, Springer Verlag

[3] Hu , M. K.: Visual Pattern Recognition by Moment Invariants, IRE Transaction on. Information Theory, 1962, Vol. IT-8: 179-187

[4] Nauck, Detlef; Klawonn, Frank; Kruse, Rudolf: Neuronale Netze und Fuzzy- Systeme. 1. Auflage, Braunschweig, Wiesbaden: Vieweg, 1994, S.73 f.

Autor

Dipl.-Ing. (FH) Sebastian Seegert, studiert im Masterstudiengang „Technische Informatik“ und ist wissenschaftlicher Mitarbeiter im Forschungsschwerpunkt verteilte und mobile Applikationen der Fachhochschule Köln.

Kontakt: sebastian.seegert@fh-koeln.de

Zuverlässiges Detector Control System Board (DCSB) für das ALICE Experiment, CERN

Ralph Erdmann, Georg Hartung und Tobias Krawutschke

Abstract

Am CERN, der europäischen Organisation für Kernforschung wurde 2008 der Large Hadron Collider (LHC) in Betrieb genommen. ALICE ist eines der vier großen Experimente und besteht aus zahlreichen Detektoren. Das DCS Board (DCSB) ist ein eingebettetes System, auf dem Linux als Betriebssystem eingesetzt wird. Es wird von 9 verschiedenen Detektoren in 18 Varianten eingesetzt. Die Anpassung an die unterschiedlichsten Einsatzgebiete wird durch eine flexible Hardware Plattform bestehend aus CPU und PLD im Zusammenspiel mit speziell angepasster Software erreicht. Dieser Artikel beschreibt das System und Maßnahmen zur Steigerung der Zuverlässigkeit des Systems, um einen Einsatz im Strahlenumfeld des Teilchenbeschleunigers zu gewährleisten. Redundanz wird auf Systemebene, aber auch durch Kombinationen mehrere DCS Boards erreicht. Ein System, das den Arbeitsspeicher auf Speicherfehler durch Strahlung überprüft, ermöglicht die Abschätzung der erwarteten Fehlerrate im Betrieb.

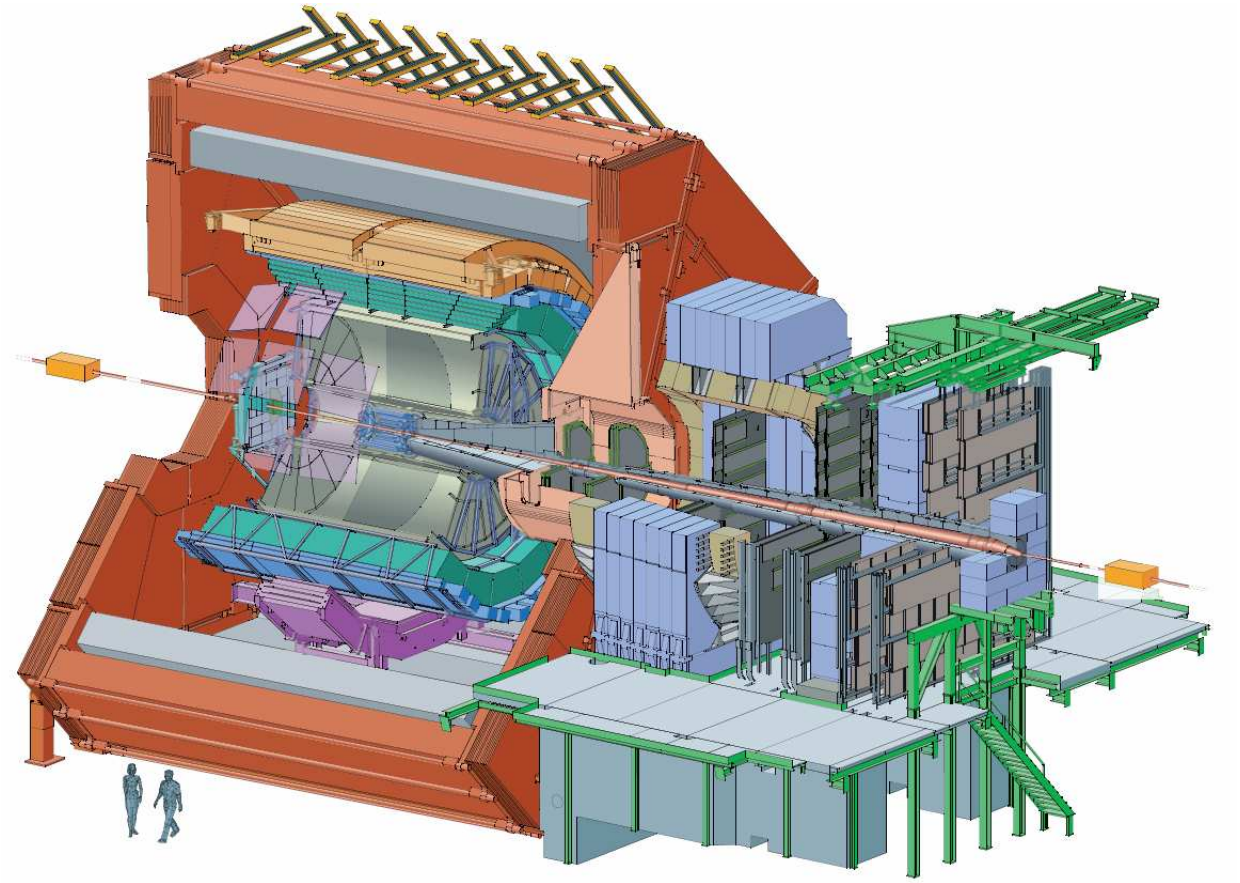


Bild 1: Schematische Ansicht des ALICE Experiments

1 Einleitung

Am CERN, der europäischen Organisation für Kernforschung wurde 2008 der Large Hadron Collider (LHC) in Betrieb genommen. In der noch andauernden Testphase wurden Protonen auf bis zu 3,5 TeV beschleunigt, bis zu 7 TeV ist geplant. Für Ende 2010 sind erste Kollisionen von Bleiionen geplant. Das ALICE Experiment ist eines der vier großen Einzelexperimente am CERN und spezialisiert auf die Untersuchung des Quark-Gluon-Plasmas (QGP), ein Zustand in dem sich das Universum kurz nach dem Urknall befand[1].

Das ALICE Experiment besteht aus mehreren Einzeldetektoren, die größtenteils in Schichten um die Kollisionsstelle herum angeordnet sind (s. **Bild 1**). Weitere Detektoren befinden sich vor und hinter der Kollisionsstelle. Je nach Entfernung zur Kollisionsstelle sind die Detektoren an die unterschiedlichen Anforderungen an Auflösung, Datenrate und Strahlenfestigkeit angepasst. Moderne Detektoren transferieren Teile der Datenauswertung und Vorverarbeitung in die Elektronik am Messgerät (Front End Electronic – FEE), um eine Datenreduktion zu erreichen, aber auch um das Signal/Rauschverhältnis klein zu halten. Daraus folgt eine Aufgabenerweiterung des Detektor Control Systems (DCS).

Die Menge an Anforderungen wird durch das Detektorumfeld erweitert:

- Das gesamte Experiment wird in einem Magnetfeld von bis zu 0,5 Tesla betrieben, so dass induktive Kopplungen, die z.B. bei Ethernet genutzt werden, gestört werden.
- Das Strahlenfeld des Beschleunigers kann zu Einzeleffekten im Silizium der Halbleiter führen und von transienten Störungen bis zur Zerstörung von Halbleitern führen.
- Da im DCS die Steuerung verschiedener Detektoren zusammenläuft, sollte es ein flexibles System sein, das an verschiedene FEEs angepasst werden kann

Das DCS Board (DCSB) ist ein eingebettetes System, auf dem Linux als Betriebssystem eingesetzt wird. Es wird von 9 verschiedenen Detektoren in 18 Varianten eingesetzt. Wenn ALICE voll ausgebaut ist, werden über 1000 DCSBs nah an der Messelektronik eingebaut sein.

Die Anforderungen und Aufgaben des Experimentes werden im Folgenden am Beispiel des TRD (Transition Radiation Detector) erläutert.

2 Der Übergangsstrahlungsdetektor (Transition Radiation Detector - TRD)

Der Übergangsstrahlungsdetektor besteht aus 18 Supermodulen, die kreisförmig um die Strahlenachse angeordnet sind. Jedes Supermodul ist in sechs Schichten aufgeteilt, die aus fünf Auslesekammern bestehen. Ein Supermodul besteht somit aus 30, der ganze Detektor aus 540 Kammern. Die tragende Konstruktion hat einen Durchmesser von ca. 8m und eine Länge von ca. 7m (s. **Bild 2**). Die aktive Fläche des TRD beträgt 751 m². Innerhalb des TRD befinden sich TPC (Time Projection Chamber) und ITS (Inner Tracking System). Letzteres befindet sich dem Kollisionspunkt am nächsten. Jede Kammer ist mit einer Mischung aus Xenon und CO2 Gas gefüllt. Geladene Teilchen hinterlassen eine Spur ionisierten Gases in der Kammer. Die Elektronen treiben in einem elektrischen Feld zu den Anodendrähten und werden nach einer Verstärkung an den Pads, die mit der FEE verbunden sind, gemessen. Durch die unterschiedlichen Zeiten, die ein Elektron in der Kammer bis zu den Kathoden treibt, lässt sich der Vektor der ionisierenden Strahlung rekonstruieren.

Der TRD besteht aus etwa 1,2 Millionen Pads, deren Ladungsinformationen über Vorverstärker und Analog Digital Wandler direkt an etwa 65000 CPUs weitergegeben werden. Aus diesen Informationen werden Spurinformationen berechnet.

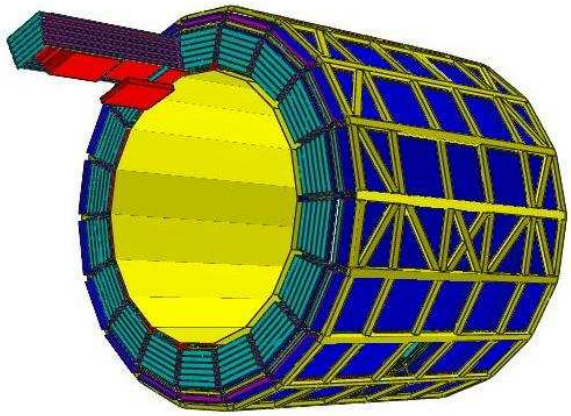


Bild 2: TRD Detektor

Das DCSB übernimmt zu den klassischen Detektor Kontroll Aufgaben (Überwachung von Umgebungs-parametern wie Temperatur, Strom, Spannung, Druck etc.) weitere Aufgaben. Moderne Detektoren erwarten vom DCS die Verteilung globaler Parameter, wie Takt,

Trigger und Informationen über den Zustand des gesamten Experiments bzw. Beschleunigers.

3 Die Hardware

Das in **Bild 3** gezeigte DCS Board kontrolliert das Experiment und wird mit seinen langen Steckverbindern an der Unterseite direkt auf die Detektorelektronik aufgesteckt. Über weitere Steckverbinder oben und an der Seite wird die Kommunikation zur Außenwelt hergestellt. Die verwendeten integrierten Schaltkreise werden im Folgenden erläutert.

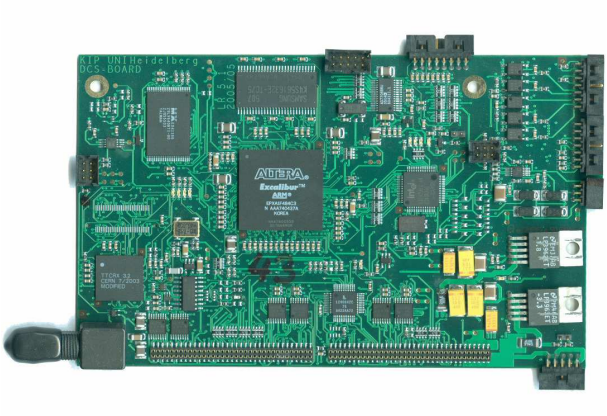


Bild 3: DCS Board

3.1 CPU und PLD

Der EPXA1 vereinigt in einem Gehäuse eine CPU und einen PLD (Programmable Logic Device), sowie weitere Peripherie wie UART, SDRAM Controller, etc.. Die CPU ist ein ARM922T mit 32 Bit Registerbreite, 5 stufiger Pipeline, Daten-, Befehlscache und Memory Management Unit, welche den Betrieb eines Standard Linux Betriebssystems ermöglicht. Die verschiedenen Komponenten auf dem Chip sind untereinander durch Busse verbunden, PLD und CPU teilen sich den Zugriff auf einen Speicher, so dass hoher Datendurchsatz durch DMA möglich ist.

3.2 Speicher

Für Betriebssystem und Programme ist das Board mit 32 MB SDRAM Arbeitsspeicher bestückt. Der im EPXA1 integrierte SD RAM Controller übernimmt Timing und Adressierung des dynamischen Speichers. Zusätzlich sind 8MB nichtflüchtiger Speicher integriert, der zur permanenten Speicherung von Bootloader, PLD Konfiguration, Ethernet MAC Adresse, Betriebssystem und Programmen genutzt wird. Die ersten vier Blöcke werden für Bootloader und PLD Konfiguration genutzt, d.h. nach einem reset führt die CPU Programmcode ab der ersten Adresse im Flash aus. Dort ist der Bootloader positioniert, der vor dem Starten des Betriebssystems den PLD konfiguriert. Ein Block ist für die Boardnummer und MAC Adresse reserviert, 11 Blöcke für den Betriebssystemkern. Die verbleibenden 7MB sind in zwei Laufwerke aufgeteilt, die mit einem Flash Dateisystem (jffs2) formatiert sind, auf die das Betriebssystem lesend und schreibend zugreifen kann.

3.3 Kommunikation

Eine Hauptaufgabe des DCSB ist die Vermittlung von Daten der Bedienebene zur Detektorelektronik und zurück, realisiert durch ein Prozessvisualisierungssystem (PVSS). Dabei wird hauptsächlich TCP/IP über Ethernet vom DCSB zu PVSS eingesetzt.

Jedes DCSB kontrolliert eine TRD Messkammer, die bis zu 138 Multichipmodule(MCM), bestehend aus Vorverstärker, ADC und CPUs, vereinigt. Die Netzwerkstruktur des Slow Control Serial Network (SCSN) besteht aus den in Reihe geschalteten MCMs, die Datenpakete empfangen und entweder weiterleiten oder bearbeiten. Da je ca. 30 MCMs in vier Ketten geschaltet sind, die durch den Ausfall eines Gliedes unterbrochen wären, ist hier Redundanz durch jeweils einen zweiten, gegenläufigen Kommunikationsring realisiert.

Kommunikation findet nicht nur zu den unteren (FEE) und oberen Hierarchien (Bedienebene) statt, sondern ist auch untereinander, zwischen DCSBs, möglich. Es ist ein Funktionseingriff in das benachbarte Board über JTAG[4] und Kontrollleitungen möglich. Viele mögliche Fehlerfälle lassen sich hiermit kompensieren, da über JTAG nicht nur Flash und PLD umprogrammiert werden können, sondern Funktionen indirekt vom kontrollierenden benachbarten Board aus übernommen werden können.

Globale Informationen des Experiments werden über ein optisches Netzwerk an alle Detektoren verteilt (globales Taktsignal, Trigger Informationen, Experiment Zustand, etc.). Um den Zustand des Detektors zu kontrollieren werden Spannungen und Temperatur, je nach Detektor auch andere Sensorwerte, erfasst.

4 Die PLD Komponenten

Der größte Teil der beschriebenen Hardware ist an I/O Pins des PLD angeschlossen, so dass dort die entsprechende Verbindung zur CPU hergestellt wird. Die im PLD realisierte Hardware reicht von einfachen Eingabe/Ausgabe Baugruppen bis hin zu komplexen

Protokollmaschinen. Mit der Möglichkeit beliebige Hardwarefunktionen im PLD zu realisieren ergibt sich die Flexibilität dieses Board nicht nur für einen Detektor einzusetzen, sondern an beliebige FEE anzupassen.

4.1 Protokollmaschinen

Der hier als Beispiel genannte TRD Detektor setzt ein spezielles, für die Prozessoren in der FEE angepasstes, serielles Protokoll ein. Die Protokollmaschine übernimmt die Serial-/Deserialisierung, Flusskontrolle, Bit stuffing, Prüfsummenerzeugung (CRC) und Taktwiederherstellung und stellt den Data Link Layer (DLL), im Sinne des OSI Referenz Modell [2], dar. Die Hardware wird aus dem Betriebssystem über einen Gerätetreiber angesprochen. Weitere Protokollmaschinen sind für I²C, SPI, JTAG realisiert. Ein CAN Controller kann ebenso für den PLD synthetisiert werden und über einen externen Pegelwandler an einen CAN Bus angeschlossen werden.

4.2 Ethernet MAC

Zur Kommunikation mit der Bedienebene ist ein integrierter Schaltkreis für den Physical Layer vorhanden. Das Bindeglied zum Betriebssystem ist der Medium Access Controller (MAC), der im PLD realisiert ist. Es wurde bewusst darauf verzichtet einen Baustein mit integriertem MAC einzusetzen, um aus der PLD Hardware heraus Zugriff auf den Netzwerkverkehr zu bekommen. So besteht zum einem die Möglichkeit einfache Hardware mit Netzwerkfunktion ohne Betriebssystem und CPU Einfluss zu erzeugen, wie sie für die Flash Rekonfiguration genutzt wird, aber auch kritische Nachrichten mit hoher Priorität und minimaler Reaktionszeit zu versenden bzw. zu empfangen.

Die Anpassung des Ethernet MAC an den PLD ist durch die Anzahl der zur Verfügung stehenden logischen Elemente (LE) beschränkt. So wurde ein MAC entwickelt, der im Gegensatz zu kommerziellen (Altera: 4000 LE) oder freien Lösungen[1] (ca. 2500 LEs) nur knapp 1000 LEs benötigt, aber auch mit dem IEEE 802.3 [3] Standard konform ist. Die Platzeinsparung wurde durch die Beschränkung auf den full duplex Modus bei 10 MBit/s erreicht.

4.3 Hardware Monitoring

Auf die Bedeutung des TTC Systems als Verteiler von Informationen ist schon vorher hingewiesen worden. Zur Qualitätskontrolle des Signals und als Feedback für den Versender werden die ausgelösten Funktionen bzw. Daten auf dem DCSB überprüft. Hier zeigt sich der Vorteil von programmierbarer Hardware (hier PLD), da auch kurzzeitige Ereignisse und Zeitfenster verlässlich erfasst und verarbeitet werden können.

5 Zuverlässigkeit und Redundanz

Redundanz wurde bereits bei der Vorstellung der SCSN Hardware erwähnt. Die serielle Kommunikationsleitung zu den MCMs beim TRD ist doppelt ausgeführt, da ein Fehler in der Kommunikationskette bis zu 34 MCMs unerreichbar machen könnte. Die bereits erwähnte JTAG und Kontroll Schnittstelle zum benachbarten DCSB wird genutzt, um unter anderem den Inhalt des Flash Speichers neu zu schreiben. Die Auswirkung von ionisierender Strahlung auf die einzelnen Hardwarekomponenten und insbesondere auf speichernde Elemente, also Register, Arbeitsspeicher und PLD-SRAM Zellen wurde untersucht und führte zu einer Software die im laufenden Betrieb Speicherfehler überprüft.

5.1 Flash Programmierung

Der Flash Speicher und dessen Inhalt ist für den Betrieb des DCSB unersetzlich, denn dieser enthält den Programmcode der CPU. Nach einem Reset wird der PLD mit dem im Flash Speicher stehenden Code konfiguriert. Eine fehlerhafte Firmware kann ein DCSB unbrauchbar machen. Um ein bereits im Detektor eingebautes DCSB, welches nicht mehr erreichbar ist, mit einer neuen Firmware zu versorgen wurde eine Hardware Software Kombination entwickelt, um das Flash vom benachbartem DCSB aus zu beschreiben. Über die JTAG Schnittstelle wird der PLD mit einer besonderen Hardware konfiguriert, um einen Datenpfad vom Ethernet Physical Layer zum Flash einzurichten. Eine Zustandsmaschine regelt den Datenfluss und das Kommunikationsprotokoll.

5.2 Strahlentoleranz, Online Memory Selftest (OMS)

Die Hardware Komponenten sowie das gesamte DCSB wurden im Vorfeld der Entwicklung in Strahlentests untersucht (u.a. in [5]). Hier zeigte sich, dass bei der am CERN zu erwartenden Strahlendosis keine permanenten Schäden durch ionisierende Strahlung auftreten werden. Einzeleffekte (Single Event Upsets – SEU) wurden beobachtet und werden durchaus erwartet. Die Ladung, die ein hochenergetisches Teilchen in das Siliziumgitter eines Halbleiters einbringt, kann zur Änderung von Speicherinhalten führen. Die Strahlentests in Verbindung mit Simulationen des zu erwartenden Teilchenflusses lassen eine Voraussage der mittleren Fehlerrate von ca. einem Tag für ein einziges DCSB zu. Diese Fehlerrate ist akzeptabel, da die DCSBs nur bei der Umkonfiguration genutzt werden. Um einen zuverlässigen Wert während

[1]Opencores ethmac: http://www.opencores.org

des Experiments zu ermitteln, und die Voraussage der Simulation zu bestätigen oder zu widerlegen, ist ein Test auf SEU sinnvoll. Der Online Memory Selftest (OMS) wurde entwickelt, um den ungenutzten Speicher des Systems für die Detektion von SEUs zu nutzen.

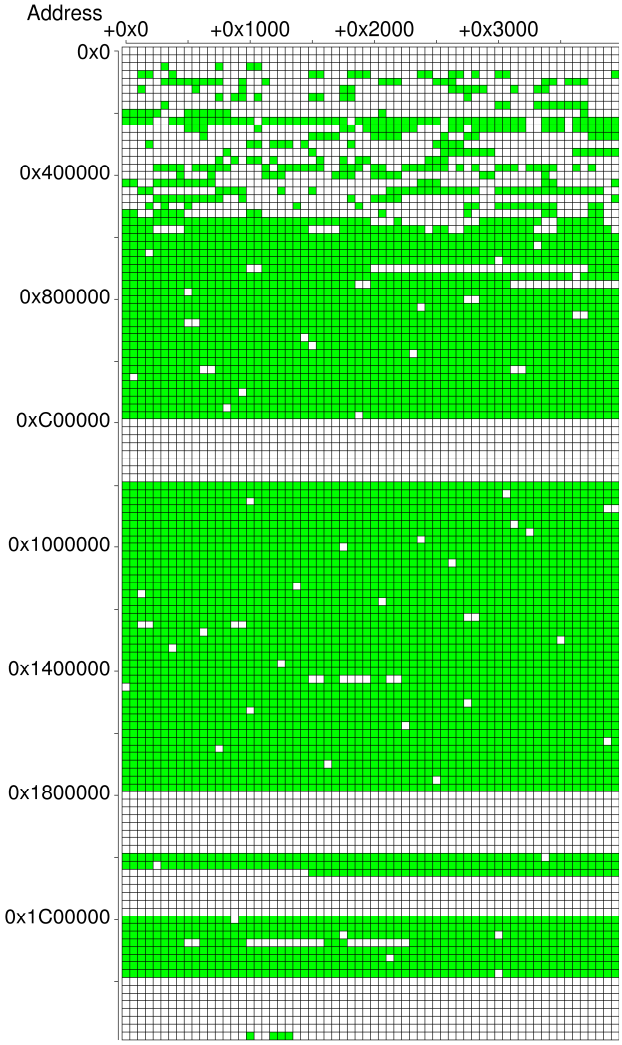


Bild 4: Der freie und vom OMS überprüfte Speicher

Das OMS System wird als Kernel Modul geladen. Über das /proc Dateisystem werden virtuelle Dateien erzeugt, die der Kontrolle des OMS aus dem user space, z.B einem Shell script, dienen. Die Häufigkeit der Überprüfung des Speichers und die Anzahl der Seiten, die auf einmal getestet werden, bestimmen die zusätzliche CPU Auslastung. Diese Parameter können aus dem user space nahezu stufenlos eingestellt werden. Die Maximale Bandbreite zur Überprüfung beträgt 26MB/s.

Findet das OMS einen Speicherfehler so wird er als solcher gemeldet. Die Informationen über Speichergröße, Dauer und gefundenen Fehlern werden gesammelt um eine reale Voraussage über die mittlere Fehlerrate zu treffen und die Periode für regelmäßige Neustarts darauf einzustellen. Das Verteilen der Daten erfolgt über DIM (Distributed Information Management [6]), einem auf TCP/IP aufbauendes Netzwerk zum Datenaustausch.

6 Literatur

[1] „First proton--proton collisions at the LHC as observed with the ALICE detector: measurement of the charged particle pseudorapidity density at sqrt(s) = 900 GeV“, ALICE collaboration, The European Physical Journal C: Volume 65, Issue 1 (2010), Page 111

[2] ISO/IEC 7498-1: Open Systems Interconnection Model, 2nd ed, 1994

[3] IEEE Std 802.3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, 2000 Edition

[4] IEEE Std 1149.1: Standard Test Access Port and Boundary-Scan Architecture for test access ports, 1990

[5] „Irradiation tests of the complete alice tpc front-end electronics chain“, J. Alme and K. Røed, 11th Workshop on Electronics for LHC and future Experiments, 2005.

[6] http://dim.web.cern.ch/dim/

Autor

B. Sc. Ralph Erdmann, studiert im Masterstudiengang „Technische Informatik“ und ist wissenschaftlicher Projektmitarbeiter am Institut für Nachrichtentechnik der Fachhochschule Köln.

Kontakt: ralph.erdmann@gmx.de

Dr. rer. nat. Tobias Krawutschke, ist wissenschaftlicher Mitarbeiter am Institut für Nachrichtentechnik der Fachhochschule Köln und Mitarbeiter bei der European Organization for Nuclear Research CERN.

Kontakt: tobias.krawutschke@koeln.de

Prof. Dr.-Ing. Georg Hartung, ist Dozent für das Lehrgebiet „Technische Informatik“ mit dem Schwerpunkt „Eingebettete Systeme“ am Institut für Nachrichtentechnik der Fachhochschule Köln.

Kontakt: georg.hartung@fh-koeln.de