Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Nicolas Bänisch

Taktiles Sensorsystem zur Schlupferkennung an einem Assistenzroboter

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Nicolas Bänisch

Taktiles Sensorsystem zur Schlupferkennung an einem Assistenzroboter

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Andreas Meisel
Zweitgutachter: Prof. Dr. Franz Korf

Abgegeben am 14. September 2015

Nicolas Bänisch

Thema der Masterarbeit
Taktiles Sensorsystem zur Schlupferkennung an einem Assistenzroboter
Stichworte
Assistenzroboter, Taktiler Sensor, Mikrocontroller, Schlupferkennung Datenübertragung, Mensch-Roboter-Interaktion, Rutscherkennung, Sensorarray, Taxel, Schlupf, Mikrocontroller, kostengünstig
Kurzzusammenfassung
Die taktile Sensorik ist eine der wichtigsten Forschungsgebiete im Bereich der Assistenzroboter. Um dieses weiter voranzutreiben, beschreibt diese Arbeit die Entwicklung eines einfachen und kostengünstigen taktilen Sensorsystems zur Schlupferkennung. Dabei liegt der Fokus besonders auf einer hohen Abtastrate (von über $6kHz$) und einer guten Auflösung der Sensoren ($< 5mm$).
Vor der Umsetzung des Systems, werden zuvor mehrere mögliche Ansätze vorgestellt, erarbeitet und evaluiert, um das bestmögliche Ergebnis zu erzielen. Eine Testanwendung demonstriert am Ende der Arbeit die Leistungsfähigkeit des Sensorsystems.

Title of the paper
Tactile sensor system for slip detection for assistant robots
Keywords
assistant robots, tactile sensors, slip detection, sensor arrays, taxel, force-sensitive materials, human manipulation research, human-computer interaction, human-robot interaction, modular high-speed tactile sensor, microcontroller, lowcost
Abstract
Tactile sensing is one of the important fields of research in the domains of assistant robots. To provide appropriate tactile sensing capabilities, this work presents the development of a new low cost and easy to use tactile sensor system, basically designed for slip detection. It is focusing particularly on high frame rates (over $6kHz$) and good sensor resolution (< 5 mm).
Before the final implementation starts, it compares, prepares and evaluates different sensor layouts, to achieve the best result. An example application demonstrates the capability of the developed sensor.

Inhaltsverzeichnis

1 Einleitung

Diese Master Arbeit beschäftigt sich im weitesten Sinne mit dem Thema des autonomen Greifen eines Assistenzroboters. Genauer spezifiziert, handelt es sich um rutschsicheres Greifen. Diese Fähigkeit soll mithilfe von taktilen Sensor Arrays ermöglicht werden, welche in dieser Arbeit ausführlich erarbeitet werden. Dabei steht der Fokus auf der Erfassung der Sensordaten, deren Verarbeitung und Übertragung an eine zentrale Stelle und legt damit den Grundstein für eine Vielzahl an möglichen autonomen Greifoperationen, in unbekannten Umgebungen.

1.1 Motivation

Die Entwicklung von Assistenzrobotern schreitet stetig voran und zeigt die Wichtigkeit dieser. Großes Potential liegt dabei besonders auf Roboter, im Gesundheitsmarkt und in der Altenpflege. So können die Roboter alten und hilfebedürftigen Menschen bei der Arbeit im Haushalt helfen, wie zum Beispiel Staub saugen, Medikamente reichen, aber auch im Notfall den Arzt verständigen. Das der Einsatz assistierender Roboter nicht nur sinnvoll, sondern in vielen Fällen sogar unvermeidlich ist, zeigt zum einen die stark steigende Zahl von Ein-Personen Haushalten [vgl. 89]. Viele Menschen sind auf sich alleine gestellt und kommen ohne fremde Hilfe kaum mehr zurecht. Besonders bei älteren Menschen ist dieses Problem steigend[vgl. 61]. Zum anderen zeigt auch der Markt, durch die bereits vorhandenen vielen kleinen elektronischen Helfern die Menschen im Haushalt unterstützen, dass der Bedarf steigt. So kommt der Einsatz von Unterstützenden Robotern nicht nur den hilfsbedürftigen zugute, sondern kann den stark steigenden Gesundheitskosten entgegenwirken [vgl. 11].

Damit es diesen Assistenzrobotern möglich ist, in ihrer Umwelt zu agieren und die Aufgaben bei der Unterstützung in Haushalten durchführen zu können, ist eine Vorrichtung, die das Greifen und Interagieren in der Umwelt ermöglicht, eine zwingende Voraussetzung. Dabei spielt nicht nur die Motorik eine wichtige Rolle, sondern ganz besonders die taktile Wahrnehmung der Umwelt.

Diese Wichtigkeit spiegelt sich auch durch die stark steigenden Forschungen in diesem Gebiet wieder. Denn die Herstellung von taktilen Sensoren für Greifer an Assistenzrobotern ist ein sehr aktives und stetig wachsendes Forschungsgebiet, welches unter anderem von Cutkosky in [24] vorgestellt wird. Dabei gibt es keinen Greifer der annähernd , gemessen an seinen Fähigkeiten, einer menschlichen Hand entspricht. Es gibt aber eine Vielzahl die ein oder mehrere

dieser Fähigkeiten erfüllen. So gibt es auch im Bereich des rutschsicheren Greifens, schon eine beachtliche Menge an Lösungsansätzen, auf welche im Verlauf dieser Arbeit teilweise Bezug genommen wird.

Diese Arbeiten haben gezeigt, dass eine sensible Manipulation eine schnelle Antwort der taktilen Sensoren in nahezu ähnlichen Taktraten, wie des Robotersystems, benötigt. Dieses erlaubt dann die schnelle Reaktion auf den Kontakt mit Objekten, auch 'First Touch' genannt. Dieses ermöglicht es wiederum, durch verschiedenste Methoden, das Aufbauen des benötigten Drucks herzustellen, um auch unbekannte Objekt sicher in dem Greifer zu halten. Laut [40] ist es sogar eine der wichtigsten Voraussetzung um sicher, auch unbekannte Objekte zu greifen.

So ist ein Verfahren welches von diesen schnellen Reaktionszeiten profitiert, die Schlupferkennung beim Greifvorgang. Dadurch ist es möglich, genau diese unbekannten Objekte sicher zu greifen.

Und genau diese Aspekte dienen der Motivation für die Entwicklung taktiler Sensoren in dieser Master Arbeit. Konkrete Ansätze und Möglichkeiten zu den zuvor beschriebenen Punkten sollen im Folgenden genauer verdeutlicht und beschrieben werden.

1.2 Ziel der Arbeit

Das übergeordnete Ziel dieser Arbeit ist es dem Roboter zu ermöglichen, den Menschen zu helfen. Konkreter wird versucht, den Roboter durch taktile Sensorik so zu erweitern, dass er im menschlichen Umfeld interagieren und manipulieren kann. Um dieses umzusetzen, soll das taktile Greifsystem des Assistenzroboters durch Sensoren ergänzt werden. Dieses System soll ihm bei der autonomen Interaktion, in einer unstrukturierten und sich ständig ändernden Umwelt, unterstützen. Wie bereits erwähnt, gibt es für dieses Problem und Aufgabe diverse Lösungsansätze. Um das Ziel zu fokussieren, gilt es in dieser Arbeit ein Sensorsystem zu entwickelt, dass das ***rutschsichere Greifen*** beherrschen soll. Dieses Sensorsystem stellt damit auch den Kernpunkt der vorliegenden Arbeit dar.

Durch bereits abgeschlossene Arbeiten aus diesem Feld, die im Vorfeld erarbeitet wurden, konnten bereits erste Erkenntnisse gesammelt werden. Hierbei fließen besonders die Arbeiten [15] und [16], aber auch [13] mit ein. Diese befassen sich detailliert mit der Erfassung von Sensordaten und geben einen kurzen Überblick, welche Punkte und Aspekte für ein fertiges Gesamtprodukt benötigt werden. Auch auf die Sammlung, Verarbeitung und Übertragung von Sensordaten, die mit Hilfe eines taktilen Sensors erfasst werden, wird besonders in [16] eingegangen. Diese Arbeiten und das damit verbundene neue Wissen, erlauben eine präzisere Zielsetzung für dieses Sensorsystem.

Ein solches System hat eine Vielzahl an Anforderungen und schafft damit viele mögliche Ziele. Nachfolgend werden detailliert die Ziele dieser Arbeit und an das damit zu entwickelnde System beschrieben.

Hohe Taktrate: Für eine Schlupferkennung wird eine hohe Abtastrate der Sensoren benötigt. Aus diesem Grund steht das Ziel, zur Erreichung einer hohe Geschwindigkeit, mit an oberster Stelle. Es wird eine *Framerate* von über $500Hz$ angestrebt, welche besonders für den *First Touch* und wie sich später zeigen wird, auch für das Erkennen des Schlupfs zum rutschsicheren Greifen, essentiell ist.

Druckkraftmessung: Es geht bei dieser Arbeit nicht primär um die Druckkraftmessung, allerdings ist für die Schlupferkennung eine gute Wiederholgenauigkeit der Messergebnisse erforderlich. So wird als Ziel auch eine gute Präzision von mehreren Druckkraftwerten gesetzt.

Konstruktion: Neben den Zielen an die Performance stellt auch die Konstruktion und der technische Aufbau einen wichtigen Teil dieser Arbeit dar. Das gesamte System soll einfach, mit preiswerten und gut beschaffbaren Komponenten erstellt werden. So soll sich ein Sensor aus diesem System von den bereits vorhandenen und sehr teuren Produkten zur Druckmessung deutlich abheben. Zudem soll das Sensorsystem so gebaut werden, dass der Roboter in seiner Bewegung nicht eingeschränkt wirkt. Hier spielt auch eine maximale Minimierung der Verkabelung eine wichtige Rolle. Auch die Integration des Sensorsystems soll auf einfachen Protokollen basieren und das System nicht zu stark belasten.

Modularität: Da sich die Anforderungen an Assistenzroboter ständig ändern und neue Aufgaben zu erfüllen haben, wird sich dieses auch auf den Greifer und damit auf die taktilen Sensoren ausüben. Um durch eine Änderung der Sensoren das gesamte System möglichst wenig zu beeinflussen, soll es hoch modular aufgebaut sein. Dadurch wird es zum einen einfach zu nutzen und integrieren sein, zum anderen aber auch leicht erweiterbar. So kann sich die Form und Anzahl der Sensoren ändern, ohne Einfluss auf das Gesamtsystem zu nehmen.

Zusammengefasst besteht die zu lösende Aufgabe darin, mit einem einfachen Verfahren einen schnellen, kostengünstigen und einfach zu handhabenden taktilen Sensor zu erstellen. Dieser soll in einem System integriert ermöglichen, durch die Erkennung des Schlupfs an seiner Sensorfläche, einen dem Assistenzroboter unbekannten Gegenstand sicher greifen zu können.

So soll der Assistenzroboter *SCITOS G5* am Ende der Masterarbeit über ein neues Sensorsystem an seinem Zwei-Finger-Greifer verfügen, mit dem er in der Lage ist selbstständig, selbst fragilste Objekte, sicher in seiner Umgebung greifen und damit manipulieren zu können. Genauere Anforderungen werden im späteren Teil dieser Arbeit erarbeitet und aufgeführt.

Der detaillierte Aufbau dieser Arbeit wird nachfolgend im Abschnitt *Inhaltlicher Aufbau der Arbeit* beschrieben und gibt damit einen weiteren Einblick in das Ziel dieser Arbeit.

1.3 Inhaltlicher Aufbau der Arbeit

Die große Gesamtaufgabe der Masterarbeit und das zu entwickelnde Sensorsystem, wird in mehrere kleine Teile unterteilt. Teilkomponenten werden dabei auch unabhängig von dem Gesamtsystem erarbeitet, analysiert und getestet. Einige dieser Teilabschnitte wurden bereits in der Arbeit [15] und [16] erarbeitet. Die Ergebnisse sowie Erfahrungen fließen dabei in diese Arbeit stark mit ein.

Die vorliegende Arbeit teilt sich grob in fünf Bereiche auf. Beginnend mit einer Einführung in die Thematik. Nachfolgend eine ausführliche Analyse der entscheidenden Kriterien und verwandter Arbeiten. Darauf folgt das Konzept, welches als ganzes aber auch für jede Einzelkomponente erstellt und evaluiert wird. Anschließend folgt die Umsetzung des Systems, worauf die Auswertung und Zusammenfassung folgt. Die fünf Bereiche werden in 9 Kapitel gegliedert. Die folgende Gliederung gibt zu jedem dieser Kapitel eine kurze Zusammenfassung der Inhalte wieder.

1. **Einleitung:** Hier wird mithilfe einer Motivation, eine kurze Einführung in das Thema gegeben. Des Weiteren wird der Zweck dieser Ausarbeitung erläutert.
2. **Grundlagen:** In diesem Kapitel gibt es einen kleinen Einblick in die Technik und Funktionsweise der taktilen Sensoren, digitalen Schnittstellen, Mikrocontrollern und abschließend eine Kleine Beschreibung des Assistenzroboters. Damit soll das Verständnis der nachfolgenden Kapitel verstärkt werden.
3. **Analyse:** In diesem Kapitel werden vorab verwandten und vergleichbaren Arbeiten vorgestellt, welche durch bereits gewonnen Ergebnisse, als Einstieg in die Analyse helfen sollen. Anschließend wird analysiert, welche Eigenschaften, Fähigkeiten und Übertragungsgeschwindigkeiten, für die Entwicklung eines gescheiten Sensorsystems, wichtig sind. Dazu werden, unter anderen mithilfe des Greifverhalten und Tastsinns des Menschen erste Erkenntnisse gesammelt. Am Ende werden die daraus gewonnen Anforderungen an das System erfasst und verbindlich aufgestellt.
4. **Grundkonzept des Sensorsystems:** Dieses Kapitel beschreibt den konzeptionellen Grundaufbau des Sensorsystems. Dieses wird mithilfe der vorherigen Analyse entworfen und berücksichtigt die vorher definierten Anforderungen. Die einzelnen Teilkomponenten werden dabei ausführlich in den drei nachfolgenden Kapiteln konzipiert und evaluiert.
5. **Evaluierung der Erfassungseinheit:** In diesem Kapitel wird die zuvor im Konzept angesprochene Erfassungseinheit, erarbeitet und evaluiert. Dabei werden in einigen Teilen mehrere Ansätze und verschiedene Methoden entwickelt, umgesetzt und Ergebnisse zusammengefasst.
6. **Evaluierung der Verarbeitungseinheit:** In diesem Kapitel wird die zuvor im Konzept angesprochene Verarbeitungseinheit, erarbeitet und evaluiert. Dabei werden in einigen Teilen

mehrere Ansätze und verschiedene Methoden entwickelt, umgesetzt und Ergebnisse zusammengefasst.

7. Evaluierung des Sensorsystem: In diesem Kapitel wird das abschließend der Evaluierung, das gesamte Sensorsystem dargestellt und als ganzes beschrieben und evaluiert.

8. Das taktile Sensorsystem: Hier wird das zuvor erarbeitete Konzept, mithilfe der getätigten Evaluierung, in einem gesamten umgesetzt. Abschließend werden die Ergebnisse in einer Auswertung präsentiert.

9. Zusammenfassung und Fazit: Dieses Kapitel gibt eine Zusammenfassung der Arbeit wieder schließt sie mit einem Fazit ab.

2 Grundlagen

Die Grundlagen geben einen kleinen Einblick in die Technik und Funktionsweise verschiedenster Systeme, die für diese Arbeit relevant sind und sollen so das Verständnis für die nachfolgenden Kapitel stärken. Zu Beginn werden die Technologien, die im Rahmen von taktilen Sensoren für diese Arbeit von entscheidender Rollen sind, beschrieben. Anschließend werde digitale Schnittstellen, im Hinblick auf eine Übertragung von Sensordaten zwischen hardwarenahen Systemen, vermittelt. Besonders auf den *Universal Serial Bus* (USB) wird hier ausführlich eingegangen, da dieser im Vergleich zu den anderen, deutlich komplexer ist. Aber auch das *Serial Peripheral Interface* (SPI) erhält, aufgrund seiner Wichtigkeit in dieser Arbeit, eine ausführlichere Beschreibung. Nachfolgend gibt es eine kleine Einweisung in die Technologie der Mikrocontroller und abschließend eine Darstellung des Zielsystems, den Assistenzroboter *Scitos*.

2.1 Taktile Sensor Technologie

Diese Sektion gibt einen kleinen Einstieg in die taktile Sensorik. Dazu wird, beginnend bei den Grundlagen der Sensorik, das Wissen Schritt für Schritt aufgebaut.

Die hier vorgestellten Technologien, wurden in fundamentaler Weise schon im Jahre 1990 und 1992 von Russell [67] und Nicholls [52] beschrieben. Doch durch die voranschreitende Technik, besseren Designs und jahrelanger Forschung, bietet die heutige Zeit eine deutlich fortgeschrittene Technik und ein große Anzahl an verschiedenster Methoden, um diverse Probleme bei der taktilen Aufnahme zu lösen. Die hier vorgestellte Auswahl an Systemen spiegelt dabei nur einen kleinen Teil, der für diese Arbeit relevant ist, des gesamten Umfangs der taktilen Technologie wieder. Eine Ausführliche Beschreibung und Vorstellung im großen Umfang wird unter anderem in [24], [25], [26] und [32] getätigt. Neben einer kleinen Einführung in die Sensorik, werden zuerst einige Verfahren zu Druckmessung vorgestellt, die im Rahmen von taktilen Sensoren oft zum Einsatz kommen. Nachfolgend wird der Begriff *Schlupfsensor* erläutert und beschrieben. Am Ende wird dann konkret auf den Taktilen Sensor eingegangen.

2.1.1 Sensor

Ein Sensor (Messfühler) erfasst zeitvariable physikalische nichtelektrische Zustandsgrößen. Der Sensor ist also ein technisches Bauteil, das eine physikalische nichtelektrische Zustandsgröße in eine elektrische Zustandsgröße umwandelt [vgl. 69, S. 11].

2.1.2 Drucksensor

Der Drucksensor oder auch Kraftsensor erfasst die physikalische Druckkraft und wandelt diese in elektrische Zustände um. Er gehört zu der Gruppe der Druckmessgeräte, welche die physikalische Größe Druck in eine elektrische Ausgangsgröße umwandelt. Dabei lässt sich Druck wie folgt beschreiben: Wirkt eine Kraft senkrecht und gleichmäßig verteilt auf eine Fläche, dann bezeichnet man den Quotienten aus dem Betrag der Kraft F zur Größe der Fläche A als Druck p.

$$p = \frac{F}{A} \tag{2.1}$$

Die SI-Einheit von Druck ist Pascal (Einheitszeichen Pa), welche sich gemäß der Formel 2.1 als N/m^2 darstellen lässt. Stellt man die Formel zur Kraft F um, lässt sich mithilfe eines Drucksensors auch die Kraft messen. Dabei muss die Sensorfläche A gleichmäßigem mit Druck p beaufschlagt werden. So wirkt auf den Sensor eine Kraft $F = p * A$. Die Einheit von Kraft wird in Newton angegeben $N = \frac{Kg*m}{s^2}$. Welche Kraft welchem Druck entspricht, ist dabei allerdings von dem jeweiligen Drucksensor abhängig.

Es stehen eine Vielzahl verschiedener Methoden zur Kraftübertragung zur Verfügung. Diese können in zwei Kategorien eingeteilt werden: Erstere sind Methoden, die auf ein Zusammenspiel zwischen mechanischen und elektrischen Einfluss basieren. So wird bei den resistiven, kapazitiven und piezoelektrische Methoden, die elektrische Änderung des jeweiligen Sensor Materials, durch die mechanische Einwirkung auf den Sensor gemessen. Die zweite Kategorie besteht aus Methoden die dieses Zusammenspiel, aus mechanischer Berührung und das daraus folgende elektrische Verhalten, nicht vollführen. Dazu zählen unter anderem optische, Ultraschall basierende und magnetische Verfahren. Nachfolgend werden einige Methoden kurz beschrieben, die für diese Arbeit von Bedeutung sind.

Resistive Druckmessung

Das Prinzip der resistiven Druckmessung beruht auf der Messung der Widerstandsänderung elektrischer Widerstände durch druckabhängige Verformung [33]. Hier wird ausgenutzt, dass sich der Widerstand eines Leiters durch Verformung, Dehnung oder Stauchung ändert.

$$R = \rho * \frac{l}{A}$$

Eine mögliche Umsetzung dieses Prinzips, ist die Verwendung eines Grundkörpers, der sich unter Druck kontrolliert verformt. Auf diesem Grundkörper befinden sich dann z.B. Dehnungsmessstreifen (DMS), die diese Verformung messen (siehe Abbildung 2.1).

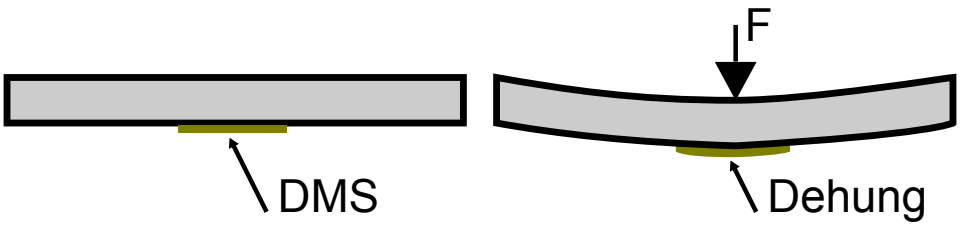


Abbildung 2.1: *Verformung der Sensormembran unter Druck*

Ein in der Robotik verbreitertes Verfahren, sind Drucksensoren die aus Materialien bestehen, die ihren Widerstand durch Einfluss von Druck/Kraft ändern. Daher werden sie auch als piezoresistive Sensoren (Piezo altgriechisch für Druck) bezeichnet.

Eines dieser Materialien, mit dem piezoresistive Verhalten, ist das *Conducive Polymer Composites* (CPC), Deutsch *leitfähiger Polymer-Verbundstoff*, welches sich bei der Druckaufnahme, im Bereich der Entwicklung von Taktilen Sensoren, immer mehr Beliebtheit verschafft [25]. Es überzeugt besonders durch seine Flexibilität und großflächige Verwendung. Das CPC besteht aus einem Trägerstoff, dem Polymer oder auch Elastomere, welcher mit leitende Mikro-/Nanopartikel gefüllt ist. Das Verhältnis der Partikel ist ungefähr 16% [25] des Volumens, je nach Einsatz, Größe und Form kann dieses aber im Bereich von $1 - 30\%$ variieren [48]. Die gängigsten Partikel sind *Metal Nanoparticle Fillers*, *Conductive Polymer Fillers*, *Carbon Micorcoil Fillers*, *Graphite Nanosheet Fillers*, *Carbon Black Nanoparticle Fillers* und *Carbon Nanotube Fillers*, welche in [25] detailliert beschrieben werden. Diese Partikel bewirken, dass durch die Ausübung von Druckkraft auf das CPC, die leitenden Partikel enger zusammengetrieben werden, wodurch die Leitfähigkeit steigt. Dieses Verhalten ist in Abbildung 2.2 dargestellt und zeigt, wie die Leitfähigkeit mit Zunahme der Druckkraft zunimmt.

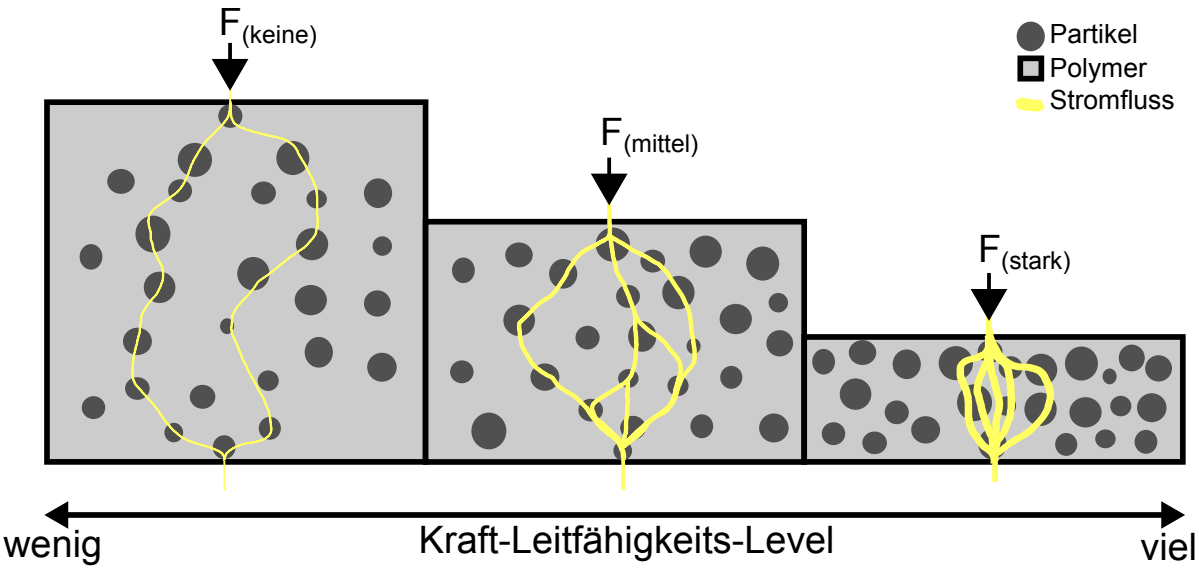


Abbildung 2.2: *Leitfähigkeitsänderung durch Ausübung von Druckkraft.*

Kapazitive Druckmessung

Das Prinzip der kapazitiven Druckmessung beruht auf der Messung der vom Plattenabstand d abhängigen Kapazität C eines Kondensators. Die Kapazität eines Zweiplattenkondensators ergibt sich aus folgender Gleichung:

$$C = \varepsilon * \frac{A}{d}$$

Umgesetzt wird das Prinzip der kapazitiven Druckmessung mit einem Grundkörper, dessen metallische oder leitend beschichtete Membran eine der beiden Platten eines Plattenkondensators bildet. Wird die Membran unter Druck ausgelenkt, verringert sich der Plattenabstand des Kondensators, wodurch dessen Kapazität - bei bleibender Plattenfläche A und Dielektrizitätskonstante ε - zunimmt (siehe Abbildung 2.3). Auf diese Weise lassen sich Drücke mit hoher Empfindlichkeit messen [33]. Sie lassen sich sehr klein herstellen und finden damit besonders in der Maschinen-Mensch Interaktionen großen Einsatz. So wird diese Methode nicht nur in diversen Forschungsbereichen angewandt, sonder auch in vielen kommerziellen Drucksensoren, wie den *RoboTouch*, *DigiTacts* [59] und im *iPodtouch* [2], kommen die kapazitive basierende Technologie zum Einsatz.

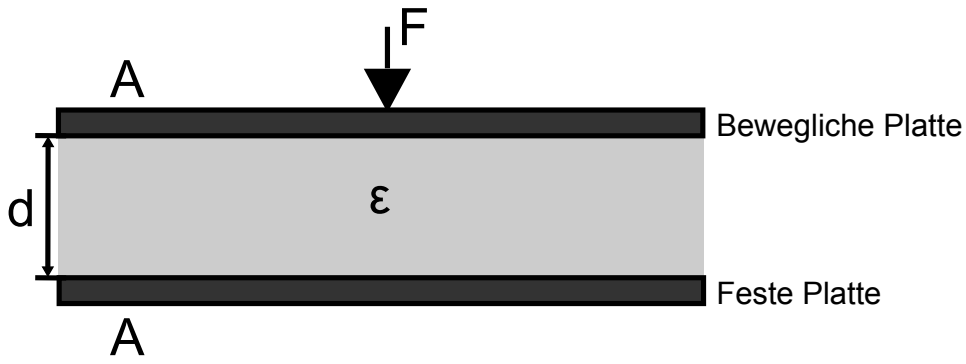


Abbildung 2.3: *Kapazitives Druckmessung Prinzip*

Piezoelektrische Druckmessung

Die piezoelektrische Druckmessung beruht auf dem gleichnamigen Kristall physikalischen Effekt, der z.B. bei Quarzkristallen auftritt. Der Druck, der auf ein piezoelektrisches Material ausgeübt wird, verursacht eine mechanische Deformation und somit eine Verschiebung der Ionen im Kristall, wodurch sich an der Oberfläche elektrische Ladung proportional zur Kraft bildet. Allerdings zeigen die Messelemente nahezu keine Verformung (typischer Weise werden die Messelemente nur um wenige Mikrometer komprimiert). Die dabei entstandene Ladung wird durch einen Ladungsverstärker in eine proportionale elektrische Spannung umgeformt.

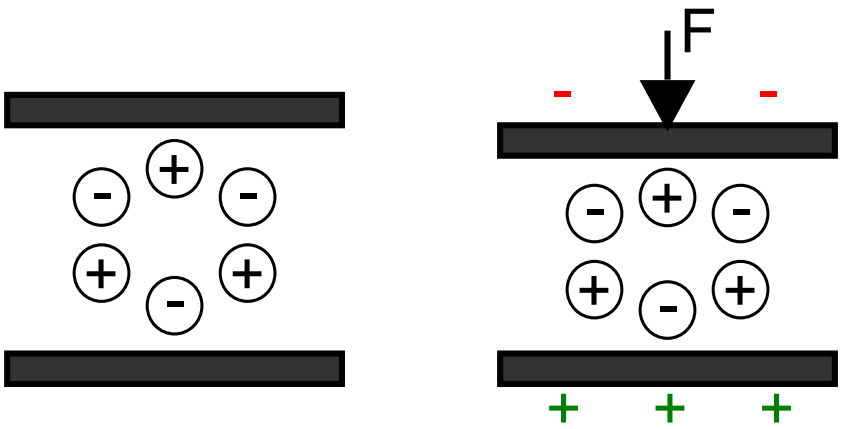


Abbildung 2.4: *Piezoelektrische Druckmessung Prinzip. Links: neutrale Ladung Rechts: durch Ausübung von Druck mit eine Ladungsverschiebung.*

Besonders vorteilhaft sind die hohe Empfindlichkeit und der Einsatz bei hohen Temperaturen, was einen Messbereich piezoelektrischer Sensoren bis zu $1000\,℃$ ermöglicht. Ein erwähnenswerter Nachteil der piezoelektrischer Sensoren tritt beim Einsatz für reine statische Messungen auf. Eine statische Kraft führt zu einer definierten Ladungsmenge an der Oberfläche des

piezoelektrischen Materials. Durch den Einsatz konventioneller Elektronik und nicht perfekt isolierender Materialien, gehen kontinuierlich Ladungen verloren, was letztendlich zu einem kontinuierlichen Signalabfall führt [33]. Ein weit verbreitetes Material mit dem piezoelektrische Effekt ist das Polyvinylidenfluorid (PVDF).

2.1.3 Schlupfsensor

Schlupf bedeutet, physikalisch gesehen, eine Abweichung der Geschwindigkeit zweier tangential belasteter und in Reibkontakt stehender mechanischer Elemente und ist damit die Voraussetzung zur Energieübertragung zwischen nicht fest miteinander verbunden Objekten. Großen Einsatz findet der Schlupfsensor in der Automobil Branche, um das bekannte Antiblockiersystem (ABS) zu ermöglichen. So wird durch Detektion von Schlupf das blockieren der Räder verhindert. Eine Simulation zu diesem Verfahren und eine genauere Beschreibung ist in [14] dargestellt.

In dem Fall der Robotik werden Schlupfsensoren eingesetzt, um die Relativbewegung, das heißt den Schlupf zwischen Roboterhand und berührten Gegenständen, zu detektieren. Dieses wird in vielen Fällen benötigt, um das autonome Greifen unbekannter Gegenstände zu ermöglichen. So wird ein möglicherweise auftretender Schlupf zwischen Objekt und Greiferanordnung erkannt und diesem sofort entgegengewirkt. Für ein kraftschlüssiges Greifen von Gegenständen mit unbekannter Geometrie und Oberflächenzustand ist es fundamental zu wissen, ob die gemessene Greifkraft genügt, um das Objekt stabil zu halten. Einerseits würde eine zu niedrige Greifkraft das Objekt rutschen lassen, andererseits könnte eine zu hohe Kraft das Objekt beschädigen [12].

Die meisten vorgeschlagenen Lösungskonzepte basieren darauf, über die Messung der tangentialen Kontaktkräfte mit Hilfe taktiler Sensoren oder reibungsbedingter Vibrationen mit Hilfe von Beschleunigungssensoren den Schlupf indirekt zu detektieren oder zu schätzen. Aber auch der Einsatz eines optischen Sensors wurde vom Fraunhofer IITB entwickelt [12], der eine direkte Messung der relativen Schlupfbewegung ermöglicht.

2.1.4 Taktile Sensoren

Taktile Sensoren bzw. Systeme sind dem menschlichem Tastsinn nachgebildet und ermöglichen es mechanische Berührungen wahrzunehmen. Sie basieren oft auf Arrays von Druck-/Kraftsensoren, die eine Überwachung über eine definierte Fläche ermöglichen und nicht nur einzelne Diskrete Punkte erfassen [19], sondern auch ortsaufgelöste Druckverteilung wahrnehmen können. Umgesetzt werden kann dieses auf verschiedenste Weise. So gibt es eine Vielzahl an Sensortypen, Materialien und Implementationen, wie resistive, piezoresistive, kapazitive, optische und viele weitere. Die große Auswahl der zur Verfügung stehenden Systeme

bietet für viele Einsatzmöglichkeiten den passenden Sensor, da sich diese in vielen Punkten unterscheiden. In vielen Fällen spielen die Sensitivität, Kosten und Flexibilität eine wichtige Rolle. Aber auch Eigenschaften wie die Geschwindigkeit, Hysterese, Temperaturempfindlichkeit, Komplexität, Genauigkeit, Auflösung usw. sind bei der Wahl oft ausschlaggebend. Der Einsatz dieser Sensor Systeme, ermöglicht dabei neuartige Ansätze im Bereich der Mensch-Maschine Interaktion und spielt in dieser eine sehr große und bedeutende Rolle. Einige dieser dadurch entstehenden essentiellen Möglichkeiten, die durch so ein taktiles Sensorsystem ermöglicht werden, sind nachfolgend aufgelistet [64]. Diese beziehen sich dabei auf den Einsatz an einem Assistenzroboter:

- Reagieren wenn der Greifer irgendetwas in seiner Umgebung beschädigen könnte, wenn er mit maximaler Kraft arbeiten würde.
- Schließen des Greifers solange bis die Finger keinen Kontakt mit dem Gegenstand oder der Umgebung haben, und anschließend sofortiges stoppen des Greifens.
- Erkennen wenn irgendetwas in der Umgebung die Hand/Arm des Roboters berührt.
- Erkennen wenn irgendetwas in der Umgebung die Finger des Roboters berührt.
- Erkennen wenn irgendetwas in den Fingern der Roboters anfängt zu rutschen oder vibrieren.
- Erkennen an welcher Position irgendetwas den Greifer oder Fingern berührt.
- Gegenstände mit dem Minimum an benötigter Kraft in seiner Umgebung zu manipulieren ohne dass diese aus den Fingern des Roboters rutschen.
- Druckkontrolle mit etwas, was sich zwischen den Fingern des Greifers befindet.
- Energie sparen, indem nur die minimale benötigte Kraft erforderlich ist, um ein Objekt zu halten
- Den Greifer und seine Finger durch Kraftkontrolle vor Beschädigungen schützen.

2.2 Digitale Schnittstellen

In der Computerwelt stellt eine Schnittstelle (Port) eine Möglichkeit zur Kommunikation mit anderen Computern, Mikrocontrollern und Geräten dar. Dazu werden diese über eine oder mehreren physikalischen Leitungen, mit Ausnahme der kabellosen Übertragung, verbunden. Dabei sind die meisten Computer Ports digital, wobei jedes Signal bzw. Bit, eine 0 oder 1 darstellt [3]. Unterschieden wird bei der Übertragung zwischen serieller und paralleler Übertragung (siehe Abbildung 2.5). Ein weiterer essentieller Unterschied liegt in der Signalübertragung, welche großen Einfluss auf die Reichweite und Geschwindigkeit der Übertragung hat. Diese Übertragungskriterien werden nachfolgend beschrieben und in den Vergleich gestellt.

Abschließend werden zwei für dieses Arbeit wichtige Schnittstellen, USB und SPI, kurz vorgestellt. Alle Sektionen beziehen sich dabei ausschließlich auf eine Kommunikation ausgehend von einem Mikrometer.

2.2.1 Parallel Übertragung

Wie der Name schon vermuten lässt, werden hier parallel mehrere Bits gleichzeitig übertragen. Die gängigste parallele Übertragung von Daten ist über eine Anzahl von mindestens 8 Datenleitungen [70]. Häufige Typen sind hier 8-Bit, 16-Bit (z.B. ISA) und 32-Bit (z.B. PCI). Durch die gleichzeitige Übertragung mehrerer Bits zur selben Zeit, wird eine hohe Datenrate mit hohem Durchsatz erreicht. Allerdings werden dafür viele Mikrocontroller Pins benötigt und es ist mit hohem Verkabelungsaufwand verbunden. Deshalb wird es häufig eingesetzt, wenn die zu kommunizierenden System nah bei einander liegen (z.B. CPU und RAM) [70].

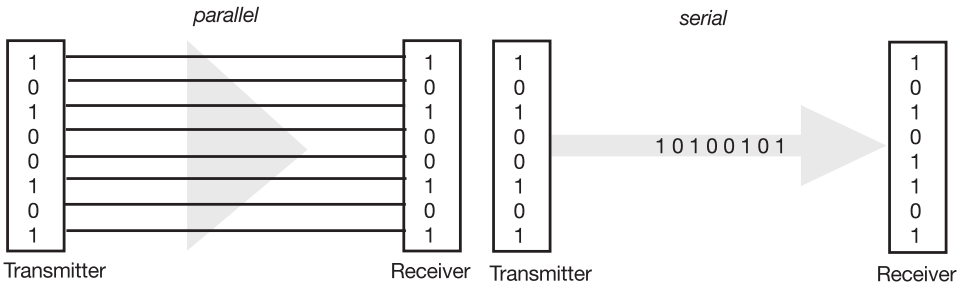


Abbildung 2.5: *Parallele und serielle Datenübertragung im Vergleich. [70]*

2.2.2 Serielle Übertragung

Eine serielle Schnittstelle bezeichnet die Übertragung eines Bits zur Zeit (siehe Abbildung 2.5). Die meisten Verfahren sind dabei bidirektional: Sie können Daten senden und empfangen. Die Übertragung eines Bits zur Zeit klingt ineffizient, hat aber den Vorteil, dass es mit wenigen einfachen physikalischen Verbindungen auskommt [4]. Dieses hält das System klein und beansprucht nur wenige Pins am Mikrocontroller. Serielle Datenübertragung kommt häufig zum Einsatz, wenn große Entfernungen überbrückt werden müssen oder eine hohe Übertragungsgeschwindigkeit nicht unbedingt erforderlich ist [95].

Wenn Daten übertragen werden, muss sicher gestellt sein, dass der Empfänger mit der selben Taktrate arbeitet, mit der der Sender die Daten abschickt und empfängt. Dieses nennt man Synchronisation der Kommunikationspartner und bedeutet, dass der Takt auf beiden Seiten identisch sein muss. Dazu gibt es zwei verschiedene Methoden der Übertragung:

Synchron

In einem synchronen Protokoll steht eine Taktleitung (Clock) zur Verfügung, die typischerweise von einem der Sender kontrolliert wird, mit der alle zu übertragende Bits synchronisiert übertragen werden. Ob das Bit bei fallender oder steigender Taktflanke akzeptiert wird, ist in Protokollen festgehalten oder muss im Vorfeld definiert werden. Beispiele für synchrone serielle Schnittstellen sind *SPI*, *I²C* und *Microwire* [4]. Der Vorteil dabei ist, dass große Datenblöcke schnell und einfach Übertragen werden können.

Asynchroner

Im Gegensatz zu der synchronen Übertragung, existiert bei der asynchronen keine Taktleitung. Jeder Teilnehmer generiert seinen eigenen Takt (oft als Baudrate[1] angegeben), welcher mit den anderen Takten der Kommunikationspartner übereinstimmen muss. Ein übertragendes *Start Bit* synchronisiert z.B. die jeweiligen Leitungen des Senders und Empfängers miteinander und das *Stop Bit* beendet die Synchronisation und damit die Übertragung. Die Vorteile von diesem Typ der Übertragung, sind die geringen Hardware Kosten und einfachen Protokolle (vgl. Tabelle 2.1).

Übertragung	***Vorteil***	***Nachteil***
Asynchron	Einfach und günstig	Hohe Verwaltungsdaten (Overhead)
Synchron	Effizient	Komplex und teuer

Tabelle 2.1: *Einfache Darstellung der Vor- und Nachteile bei der Asynchronen und Synchronen Datenübertragung. [95]*

2.2.3 Signalübertragung

Bei der Signalübertragung wird im allgemeinen zwischen asymmetrischer und symmetrischer unterschieden. Die Abbildung 2.6 zeigt diese beiden Verfahren, auf die in den beiden nachfolgenden Abschnitten näher eingegangen wird.

[1] Baud ist die so genannte Schrittgeschwindigkeit, ein Maß für die Übertragungsgeschwindigkeit (Übertragungsrate).

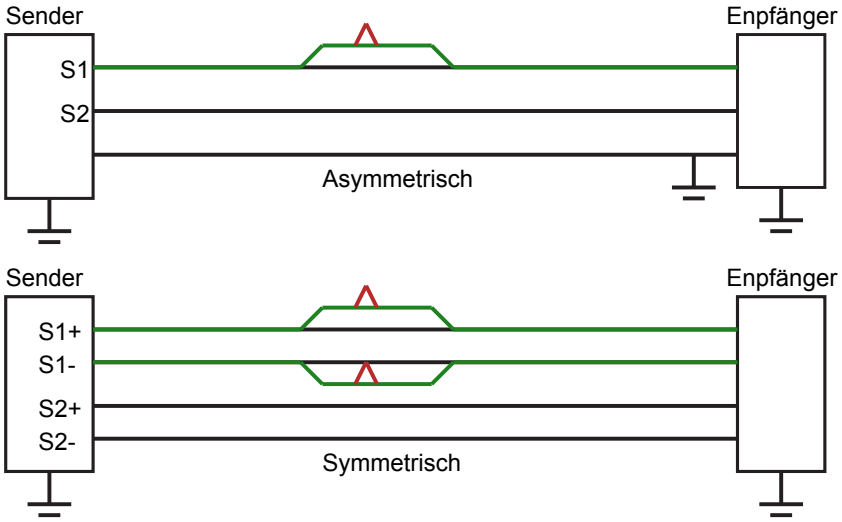


Abbildung 2.6: *Symmetrische und Asymmetrische Signalübertragung*

Asymmetrische Signalübertragung

Asymmetrische Signalübertragung (single-ended) ist die einfachste Art der Datenübertragung und wird in vielen bekannten Schnittstellen verwendet (z.B. SPI, UART und I²C). Eine Leitung zwischen Sender und Empfänger überträgt eine Spannung, die sich je nach Signal ändert, in Bezug auf eine gemeinsame Referenzleitung (häufig Masse). Das hat den Vorteil, dass man für N Datenleitungen nur $N + 1$ Leitungen benötigt und sich somit der Verkabelungsaufwand klein hält. Der große Nachteil ist die Störanfälligkeit. Denn aufgrund von nicht vorhanden redundanten Datenleitungen, ist eine Fehleranalyse nur sehr bedingt möglich.

Symmetrische Signalübertragung

Bei der symmetrischen (auch als differenziellen bezeichnet) Signalübertragung, besitzt jede Datenverbindung, zwischen Sender und Empfänger, immer ein Paar Leitungen. Dabei führen beide das gleich Signal, das eine das positive und das andere das invertiere negative Signal, welches in der Abbildung 2.6 durch den grünen Signalverlauf dargestellt ist [8]. Durch das Leitungspaar werden zwar für N Datenleitungen $2 * N$ Leitungen benötigt, allerdings hat dieses den großen Vorteil, sehr störungsunanfällig zu sein. Da es sich bei den beiden Leitungen um das selbe Material handelt, die Leitungslänge identisch ist und beide unter den selben Störungen beeinflusst werden, kann das Störsignal, welches in der Abbildung 2.6 in rot dargestellt ist, durch die Differenzbildung der beiden Leitungen eliminiert werden. Besonders bei kleinen Übertragungsspannung kann das sehr hilfreich sein. Dieses Verfahren erlaubt, gegenüber der asymmetrische Signalübertragung, eine deutliche Steigerung der Übertragungsgeschwindigkeit und Leitungslänge. Bekannte differenziellen Schnittstellen sind USB, RS-485 oder LVDS.

2.2.4 SPI - Serial Peripheral Interface

SPI steht für *Serial Peripheral Interface* - zu Deutsch serielle Peripherie Schnittstelle - und beschreibt einen synchronen seriellen Bus. Der dazugehörige Hardware Aufwand hält sich dafür in Grenzen. Es werden neben der *Slave-Select* oder auch *Chip-Select* Leitungen nur eine Steuerleitung für den Takt und zwei Datenleitungen benötigt. Des Weiteren wird für die Kommunikation, wie in Abbildung 2.7 dargestellt, meist nur ein einfaches *Shift-Register* gebraucht.

SPI ist lizenzfrei, da es niemals mit Patenten belegt wurde. Mittels SPI und ganz nach dem Master-Slave-Prinzip, welches im folgenden Abschnitt genauer erläutert wird, können digitale Schaltungen miteinander verbunden werden und kommunizieren. Hierbei ist der Mikrocontroller oft als Master anzusehen und weitere Peripherie-Chips (ICs, integrierte Schaltkreise/Schaltungen) werden als *Slaves* behandelt.

Funktionsweise

SPI ist wie bereits erwähnt als *Master-Slave* Bus ausgelegt, das heißt es muss immer ein *Master* und mindestens ein *Slave* vorhanden sein. Der Master stellt das Taktsignal zur Verfügung und leitet über die Selektierung eines *Slaves* die jeweilige Übertragung ein. Die Daten werden dabei über die beiden Datenleitungen übertragen. Eine dieser Leitungen ist für Daten von dem *Master* zum *Slave* (oft als MOSI bezeichnet) und die zweite für die entgegengesetzte Verbindung (MISO).

Synchron zum Taktsignal vom *Master* werden die Daten an den Datenleitungen ausgegeben. Dieses wird über ein Schieberegister realisiert. Meist wird das höchstwertigste Bit (MSB) zuerst ausgegeben, die niederwertigen Bits folgen. Möglich ist auch zuerst das LSB auszugeben. Die empfangenen Daten liegen nach dem Datentransfer im gleichen Register wie die Sendedaten. Es existiert also nur ein Register für Sende-/Empfangsdaten [17]. Schiebt der Master ein Datenwort in dieses Register, so wird automatisch eine Datenübertragung eingeleitet und die Bits aus dem Register des *Masters* in das des *Slaves* geschoben. Die Daten des *Slaves* werden dadurch aus seinem Register gedrückt und in das des *Mastes* übertragen (vergleiche Abbildung 2.7. Der *Slave* sendet somit nur dadurch seine Daten, da der Master seine in das Schieberigister des *Slaves* überträgt. Der *Slave* selber sendet nicht *aktiv*.

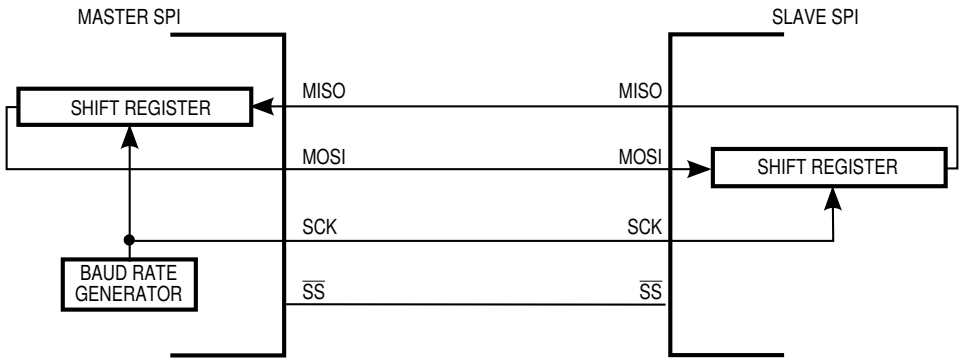


Abbildung 2.7: *SPI Master/Slave Übertragungsblock Diagramm.*

Da es kein einheitliches Protokoll gibt, muss der *Master* so konfiguriert werden, dass er mit dem *Slave*, Daten austauschen kann. Neben der Bit-Reihenfolge (MSB oder LSB) muss noch die Taktleitung konfiguriert werden. Dabei gibt es die Möglichkeit den Takt als *High* oder *Low* aktive Leitung zu realisieren. Dieses wird über den *CKPHA* Parameter getätigt. Die andere Einstellung ist der Zeitpunkt, wann ein Datenbit übernommen werden soll. Dieses kann am Anfang oder Ende des Taktsignals geschehen und wird mit dem Parameter in *CKPOL* bestimmt. Ob es sich dabei um die steigende oder fallende Flanke handelt, ist davon abhängig, wie das *CKPHA* eingestellt wurde. Durch diese zwei Parameter stehen vier Modi zur Verfügung. Bei der Taktrate ist im allgemeinen keine Grenze gesetzt, jedoch muss bei der Konfiguration die maximale Rate des *Masters* und vor allem des *Slave* berücksichtigt werden.

Eigenschaften

SPI ist eine einfache serielle Schnittstelle um Daten zu Übertragen. Dabei ist es schneller als vergleichbare asynchrone, serielle Schnittstellen. Es sind problemlos Datenraten im zweistelligen Megahertz Bereich möglich. Die Hardware ist sehr einfach und besteht in der Regel nur aus einem einfachen *Shift-Register*. Des Weiteren werden mehrerer Teilnehmer unterstützt, hingegen in der Regel aber nur ein *Master*. Bei der Verwendung mehrere Teilnehmer werden allerdings häufig mehr Signalleitungen benötigt als bei anderen Kommunikationsmethoden. Jeder dieser weiteren Teilnehmer benötigt eine separate *SS* Leitung, was bei einer hohen Anzahl, mangels I/O Leitungen am *Master*, problematisch werden kann. Des Weiteren besteht nur eine Verbinden vom *Slave* zum *Master*. Die anderen Teilnehmer können untereinander keine Daten austauschen. Die komplette Übertragung muss dabei klar definiert sein, Daten können nicht beliebig in Größe und Format verschickt werden.

Anwendungsgebiete

SPI verwendet man hauptsächlich dafür, wofür es auch entwickelt worden ist, zur Anbindung von Peripherien an einen Mikrocontroller bzw. zur Verbindung mehrerer Mikrocontroller untereinander. Dabei sollten sich alle Bauteile auf der gleichen Platine befinden. SPI eignet sich nur bedingt zur Verbindung von Platinen über ein Flachbandkabel, da es keinerlei Störschutz gegen externe Signale gibt [17].

SPI wird häufig für folgende Peripherien eingesetzt:

- Sensoren (Temperatur, Druck)
- Wandler (ADC, DAC)
- Speicher (EEPROM, Flash, MMC)
- Sonstiges wie Potentiometer, LCD-Controller, ...

2.2.5 USB - Universal-Serial-Bus

Dieses Kapitel widmet sich dem *Universal Serial Bus*, kurz USB. Es wird ein kleiner Überblick gegeben, welche Möglichkeiten USB zur Datenübertragung anbietet und wie diese Funktioniert.

Das *Universal Serial Bus* ist ein Bussystem, welches es ermöglicht eine Vielzahl unterschiedlicher USB Geräte anzuschließen. Dabei kann ein *USB-Device* über eine oder mehrere Funktionen verfügen. Pro Funktion wird vom *USB-Device* ein Interface zur Verfügung gestellt. Jedes Interface verfügt wiederum über ein oder mehrere Endpunkte. Heutzutage ist an fast jedem Computer und auch an vielen Mikrocontrollern ein USB Anschluss verfügbar. Die Kommunikation erfolgt immer zwischen *USB-Controller* und *USB-Device*, die über eine physikalische Verbindung, welche aus nur vier Leitungen (bezogen auf USB2.0) besteht. Zwei von diesen dienen zur Datenübertragung, eine stellt die gemeinsame Masse dar und eine stellt eine $5V$ Spannungsversorgung vom *Host* zum *Device* bereit. An einen *USB-Host-Controller* können bis zu 127 physische USB Geräte angeschlossen werden, wobei jedes wiederum aus mehreren logischen USB-Geräten bestehen kann [42].

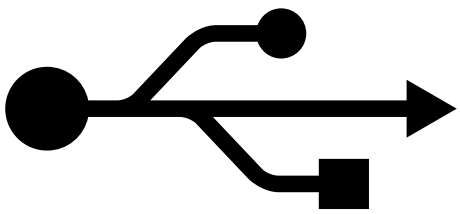


Abbildung 2.8: *USB-Symbol*

Topologie

Beim USB muss zwischen physikalischer und logischer Struktur unterschieden werden. Im Gegensatz zu Bussystemen, bei denen alle Knoten an eine gemeinsame Leitung angeschlossen werden, handelt es sich beim USB physikalisch um eine Baumstruktur. Der *USB-Host*, beziehungsweise dessen *Root-Hub*, ist der Wurzelknoten. Die Endgeräte bilden die Blattknoten und die Hubs ermöglichen Verzweigungen [5]. Logisch sind die USB-Geräte jedoch sternförmig um den *Host-Controller* angeordnet. Dabei ist jedes Gerät (auch jeder Hub) eindeutig adressiert. Der *Host-Controller* ist der einzige Bus-Master. Er sendet seine Nachrichten immer an alle Endgeräte aus, wobei nur jenes, an das die Nachricht adressiert wurde, reagieren darf. Die Endgeräte senden ihre Nachrichten direkt an den *Host*, wobei Hubs diese Nachrichten nur nach oben, und nicht an die anderen Endgeräte, weiterreichen [63].

Datentransfer

Bei dem Datentransfer werden, abhängig von ihrer Geschwindigkeit, die USB-Geräte in Klassen unterteilt. Drei dieser Klassen, beginnend bei der langsamsten, sind *Low-*, *Full-* und *High-Speed-Geräte*. Weitere, meist deutlich schneller Klassen, kommen stetig hinzu. *Low-Speed-Geräte* haben eine maximale Datenübertragungsrate von $1,5 MBit/s$, *Full-Speed-Geräte* eine maximale Datenübertragungsrate von $12 MBit/s$ und *High-Speed-Geräten* können eine Datenrate von $480 MBit/s$ erreichen [22].

Die Übertragung per USB zwischen PC(*USB-Host*) und Gerät(*USB-Device*) besteht aus mehreren Protokollschichten, auf welche hier nicht detaillierter eingegangen wird. Diese Übertragung wird im ganzen als USB Transfer bezeichnet. Dabei definiert der USB Standard vier verschieden Transfertypen: Control Transfer, Isochronous Transfer, Bulk Transfer und Interrupt Transfer, welche sich in der Richtung des Datenflusses, der Paketgröße, Verzögerung, Buszugriff, Datenformat und Fehlerbehandlung unterscheiden. Dabei werden von den Endpunkten jeweils immer nur genau einer dieser Transfertypen unterstützt. In der Tabelle 2.2 sind die aufgeführten Transferarten noch einmal in kurzer Form gegenübergestellt und werden nachfolgend detaillierter beschrieben.

Transfertyp	Beschreibung	Anwendung
Control	• Langsame Transferrate • Es wird kein weiterer Endpunkt benötigt	• Konfiguration von Geräten
Interrupt	• Sehr langsame Transferrate • Frequenz zugesichert	• Echtzeit und langsame Bandbreiten
Bulk	• Hohe Transferrate • Transfer gesichert (ACK)	• Große Datenmengen
Isochronous	• Sehr hohe Transferrate • Transfer nicht gesichert	• Streaming Anwendungen

Tabelle 2.2: *Gegenüberstellung der vier verschiedenen USB Transfertypen.*

Control Transfer : Der Control Transfers wird zur Identifikation, Steuerung und Übertragung von Konfigurationsdaten benutzt. Dies erfolgt bidirektional über so genannte *Message Pipes*. Control Transfers sind zahlreichen Einschränkungen unterworfen, was die Bus Zugriffsrechte, Dauer des Buszugriffes und Paketgröße anbetrifft. Jedes *USB-Device* benötigt für die Konfiguration mindestens einen Endpunkt mit dem[vgl. 22, S.43].

Interrupt Transfer : Der Interrupt Transfer Modus ist dafür gedacht, regelmäßig kleinere Datenvolumen zu übertragen (zum Beispiel: Tastatur oder Maus). Die Übertragung erfolgt unidirektional über *Stream Pipes*. Ein Datentransfer im Interrupt Modus erfolgt mit konstanter, also garantierter Latenz. Eine feste Bandbreite wird hingegen nicht garantiert[vgl. 22, S.44].

Isochronous Transfer : Im Isochronous Transfer Mode können große Datenmengen bei konstanter Datenrate und konstanter Latenz übertragen werden. Der Transfer erfolgt auch hier über unidirektional *Stream Pipes*. Der Nachteil dieses Modus liegt darin, dass es keinen Schutz vor Datenverlust gibt. Schlägt die Übertragung eines Datenpaketes fehl, erfolgt kein erneuter Versuch. Dieses bietet sich zum Beispiel zum Streaming von Audiodaten an[vgl. 22, S.44].

Bulk Transfer : Der Bulk Transfer wird zur Übertragung großer Datenmengen ohne garantierte Geschwindigkeit eingesetzt. Bulk Daten sind dabei größere Datenmengen die bei Druckern oder Massenspeicher anfallen. Einem Datentransfer im Bulk Transfer Modus steht keine Bandbreite von vornherein zur Verfügung, sondern wird abhängig von der Auslastung der anderen Modi zugewiesen. Demzufolge lässt sich für die zu übertragenden Daten auch keine Verzögerung ermitteln und kann, wie schon angesprochen, keine Bandbreite garantieren. Die einzige Zusage die im Bulk Transfer Modus getroffen werden kann ist, dass die Datenpaket übertragen werden. Der Transfer erfolgt auch hier unidirektional über *Stream Pipes*, was bedeutet, dass wie auch beim isochronen Modus für einen bidirektionalen Datentransfer zwei Pipes benötigt werden [vgl. 22, S.52].

Enumeration

Der Vorgang bei dem der *Host* dem Gerät eine Adresse zuordnet, alle Informationen über das Gerät erfragt, den richtigen Treiber lädt und dann eine Konfiguration auswählt, wird Enumeration genannt. Mittels Standard Anfragen (standard request) wird vom PC die Adresse des Gerätes festgelegt und mehrere so genannte Deskriptoren (Beschreibungen) erfragt. Jedes Gerät muss mindestens 4 Deskriptoren liefern, die das Gerät beschreiben.

2.3 Mikrocontroller

Im allgemeinen vereint ein Mikrocontroller verschiedene Komponenten wie

- Mikroprozessor als CPU (Central Processor Unit)
- Speicherbausteine oder -module (RAM, ROM, PROMIEPROM)
- parallele und serielle Ein/Ausgabebausteine oder E/A-Module
- Timer, Zähler, AnaloglDigital-Wandler, Interruptcontroller.

auf einem Chip. So besteht die Aufgabe des Entwicklers darin, den geeignetsten Mikrocontroller aus der angebotenen Vielfalt für seine Aufgabe auszuwählen [95]. Viele sind dafür ausgelegt mit anderen Geräten, wie Sensoren, Motoren, Displays, Speicher und auch anderen Mikrocontrollern, zu kommunizieren. So bieten sie ein ganzes Arsenal an verschieden Schnittstellen zur Datenübertragung an, welche die komplexen Probleme, wie Kosten, Größe, Gewicht, Erreichbarkeit, Geschwindigkeit, Zuverlässigkeit, Stromverbrauch und viele mehr lösen.

2.3.1 Familien

Es gibt eine Vielzahl an Herstellern die Mikrocontroller produzieren und ständig weiter entwickeln. Dabei unterscheiden sie sich Hauptsächlich durch die unterschiedlichen Prozessoren. Für diese Arbeit sind drei Hersteller relevant, die nachfolgend kurz beschrieben werden und auf ihre Prozessoren eingegangen wird. Ein weiterführender Vergleich findet sich in der Arbeit [86].

Atmel AVR *Atmel AVR* Chips sind wohl die verbreitetsten und beliebtesten Mikrocontroller im Hobby Bereich. Hierfür spricht auch, dass sowohl die Firma *Arduino* als auch *Teensy*, beides weit verbreitete Entwicklungsboard Hersteller, in vielen von ihren Boards auf

einen *Atmel AVR* setzten. Sie basieren auf einer Harvard Architektur und bieten alle bekannten Bauformen und Features an. In der Regel handelt es sich um 8 oder 32 Bit Systeme. *Atmel* stellt mit dem *AVR Studio* eine eigene IDE bereit, welche aber Aufgrund des freien *GCC* einfach durch jede beliebe Entwicklungsumgebung ersetzt werden kann.

Microchip PIC Die *Microchip PIC* Mikrocontroller waren einer der ersten die für den Hobby Bereich vermarktet wurden und schon früh in einfachen Bauformen zur Verfügung standen. Sie basieren auf einer Harvard Architektur und bieten alle bekannten Bauformen und Features an. In der Regel handelt es sich um 8 oder 32 Bit Systeme. *Microchip* stellt sowohl die IDE *MPLAB* bereit, als auch den Compiler, für welchen es bisher keine freie Alternative gibt.

Freescale *Freescale Semiconductor*, entstanden nach der Ausgliederung des Halbleiterbereichs von Motorola, sticht besonders durch seine *Kinetis Low Power 32-bit Mikrocontroller*, basierend auf einem *ARM® Cortex®-M* Kern hervor. Dieses spiegelt sich auch durch den großen Erfolg der *Teensy3.X* Plattformen wieder, die genau mit diesem Mikrocontroller bestückt sind. Das *Freescale* Entwicklungs-Kit enthält die IDE *CodeWarrior* in der auch mit dem freien GCC der ausführbare Programmcode erzeugt werden kann.

2.3.2 Schnittstellen

Jeder Mikrocontroller besitzt in der Regel diverse Schnittstellen. Einige wurden bereits im Abschnitt 2.2 vorgestellt. Zu den wohl am verbreitetsten Schnittstellen zur Datenübertragung zählen U(S)ART, SPI, I2C (TWI), CAN und USB. Aufgrund ihrer Wichtigkeit in dieser Arbeit, wird auf die SPI Schnittstelle, mit Bezug auf den Mikrocontroller, nachfolgend ausführlicher eingegangen.

SPI

Das Übertragungsverfahren SPI, welches bereits ausführlich im Kapitel 2.2.4 beschrieben wurde, steht bei einer Vielzahl der Mikrocontrollern, teilweise sogar in mehrfacher Anzahl, zur Verfügung. Die Übertragungsgeschwindigkeit ist in der Regel von der System-Taktrate abhängig. So steht der SPI Schnittstelle im normalen Fall ein Viertel dieser Taktrate zur Verfügung.

Jeder Mikrocontroller muss individuell über seine Register konfiguriert werden, damit die Übertragung vollzogen werden kann. Bei der Übertragung gibt es mehrere Möglichkeiten die jeweiligen Daten zu senden und zu empfangen. Eine Möglichkeit ist es durch so genanntes **Polling**, darauf zu warten, dass die Daten übertragen wurden bzw. die Daten angekommen sind. Dieses geschieht über das ständige Abfragen eines Bits in einem der SPI Register. In diesem Fall ist die CPU komplett damit beschäftigt das Bit abzufragen und kann währenddessen nichts anderes tun. Um diese Zeit dennoch zu nutzen, kann man statt des ständige Abfragen, einen

Interrupt konfigurieren. Dieser löst aus, sobald die Daten verschickt oder empfangen wurden. Eine noch effizientere Lösung, welche allerdings bei vielen Controllern nicht zur Verfügung steht, ist das **Direct Memory Access** kurz *DMA*, welches im Abschnitt 2.3.3 kurz beschrieben ist. Hier wird dem *Shift-Register* des SPI ein Speicher zugeordnet. Dadurch kann nicht nur CPU unabhängig gelesen, sondern auch gesendet werden. In einigen Fällen können so auch ganze Byte Ströme gespeichert werden. Die CPU muss nun nur nachfragen oder per Interrupt darüber informiert werden, ob die Übertragung fertig ist. Eine ausführliche Beschreibung ist am Beispiel des *MK20DX256* in [30] vorhanden.

2.3.3 DMA Controller

DMA steht für *Direct Memory Access* und ist ein Feature von einigen Mikrocontrollern. Das Grundprinzip von DMA ist, dass Transfers zwischen Speicher und Peripherie (UART, SPI, AD-Wandler) automatisiert im Hintergrund ablaufen, ohne die CPU nennenswert zu belasten, um so z.B. parallel andere Dinge zu tätigen. Dieses ist möglich, da die Übertragung nun komplett hardwarebasierend läuft. Zusätzlich zu der Möglichkeit nun parallel auch anderen Code auszuführen, ist die Übertragung, von z.B. SPI Daten, in vielen Fällen schneller, da direkt auf den Speicher zugegriffen wird.

2.3.4 Compiler

Da in dieser Arbeit der Großteil des Codes auf Mikrocontrollern und in der Sprache C erstellt wird, wird auch kurz auf die Compiler eingegangen. Ein Compiler ist ein Programm, dass Programmcode einer bestimmten Programmiersprache in ein für das Zielgerät ausführbare Form übersetzt. Da es diverse verschiedene Plattformen und Zielsysteme gibt, stehen auch fast genauso viele Compiler zur Verfügung. Eine freie Compiler Sammlung steht mit dem **GCC** (GNU Compiler Collection) kostenlos bereit. Hiermit lassen sich unter anderem Zielsysteme wie die ARM-Architektur, Atmel AVR und viele weitere Prozessoren compilieren. Ausführlichere Informationen zum *GCC* sind in [93] beschrieben.

Aber nicht alle Plattformen sind in der *GCC* enthalte, so wird für Mikrocontroller der Firma *Microchip* ein kommerzieller Compiler aus dem eigenen Hause benötigt. Je nach Prozessortyp stehen mehrere Compiler (z.B. XC16 und XC32) zur Auswahl. Auch eine eingeschränkte kostenlose Version, welche im Rahmen dieser Arbeit ausreichend ist, ist vorhanden [51].

2.4 Assistenzroboter Scitos

Die in dieser Arbeit entwickelten taktilen Sensoren, sollen in das System des Assistenzroboters *Scitos*, aus dem *Robot Vison Lab* der *Hochschule für Angewandte Wissenschaften*, integriert werden. Dieser Roboter besteht aus einer mobilen Plattform *SCITOS G5* der *Firma MetraLabs GmbH* und einem 5-DOF-Roboterarm mit einem Zwei-Finger-Greifer der Firma *SCHUNK* (siehe Abbildung2.9). Er wird hier benutzt, um den Einsatz und die Erprobung diverser Techniken und Funktionen im Bereich der Robotik, mit besonderem Fokus auf Anwendungen in menschlicher Umgebung, zu testen, forschen und erproben. So ist er darauf ausgelegt, sich in einem menschlich geprägten Umfeld zu bewegen und dort Hilfstätigkeiten auszuführen (z.B. Servieren, Anreichen, Suchen, Bringen, etc.) vgl. [49]. Nachfolgend wird zuerst auf die Hardware eingegangen und anschließend das Software System beschrieben.

Abbildung 2.9: *Der Assistenz Roboterarm*

2.4.1 Hardware

Hier werden die technischen Daten der Mobilen Plattform *SCITOS G5*, des Roboterarmes und des Greifers aufgeführt.

Mobile Plattform SCITOS G5

Der SCITOS G5 wurde als multifunktionale Industrie- und Forschungsplattform entwickelt und kann in Abhängigkeit vom Einsatzgebiet mit unterschiedlichen Aufbauten bestückt werden. Die Plattform wiegt 60kg und verfügt über eine Nutzlast von 50kg. Der integrierte On-Board-Computer, ein Mini-ITX PC mit x86-Architektur, hat folgende technische Daten:

- 2.048 MB DDR2-RAM
- 250 GB HDD, SATA
- RS232, VGA, LVDI
- Ethernet 10/100 RJ-45
- 5 x USB 2.0
- 2 x Firewire
- WLAN: On-board IEEE 802.11a/b/g

Zur Energieversorgung sind zwei Batterien mit jeweils $12V$ und 38/42 Amperestunden in Reihe geschaltet und ergeben eine Haupt Betriebsspannung von $25,2$ bis $25.8V$ bei voller Batterie. Das Ladegerät ist in die mobile Plattform integriert. Des Weiteren verfügt der *Scitos G5* über 24 integrierte Ultraschallsensoren zur Distanzmessung. Die Reichweite dieser beträgt ca. $20-300cm$. Die drei Räder zur Fortbewegung erreichen eine maximale Geschwindigkeit von $1,4m/s$. Die Rotationsgeschwindigkeit beträgt bis zu $360^\circ/s$. Hierbei ist aber zu bedenken, dass diese Werte sich nur auf die Mobile Plattform beziehen, ohne zusätzliche anbauten. Um die Geschwindigkeit und zurückgelegte Strecke zu bestimmen, steht ein Tickzähler zur Verfügung der pro Radumdrehung 460 Signale für die Messung bereitstellt. Für die Kollisionserkennung besitzt er einen umlaufenden Bumper.

Zusätzlich zur Grundausstattung verfügt er noch über einen an der Front montierten 2D-Laserscanner des Herstellers *Leuze Electronis* (Modell ROTOSCAN RS4-2E). Dieser wird für präzise Entfernungsmessungen in der Ebene verwendet. Weitere Informationen zu der Plattform sind in [44] dargestellt.

Roboterarm

Der auf der Mobilen Plattform montierte Roboterarm besteht aus Gelenkmodulen und Verbindungsstücken der Firma *Schunk*. Er besitzt zwei Rotationsgelenke, drei Knickgelenke und einen Zwei-Finger-Greifer (siehe Abbildung 2.10).

Der Arm ist vom Typ LWA[2] wiegt circa $13Kg$, hat eine max. Nutzlast von $5kg$ und besitzt eine Wiederholgenauigkeit[3] von $0.01mm$.

Der Arm verfügt über eine getrennte Energieversorgung für die Steuerelektronik und die Antriebsmotoren des Arms. Im Falle eines Spannungsabfalls greifen selbstständig die integrierten Magnetbremsen. Als Schnittstellen stehen ihm *RS-232*, *Profibus-DP* und *CAN-Bus* zu Verfügung [78]. Des Weiteren wurde im Zug der Arbeit von Struss [90] eine USB Verbindung von der Plattform zum Greifer gelegt.

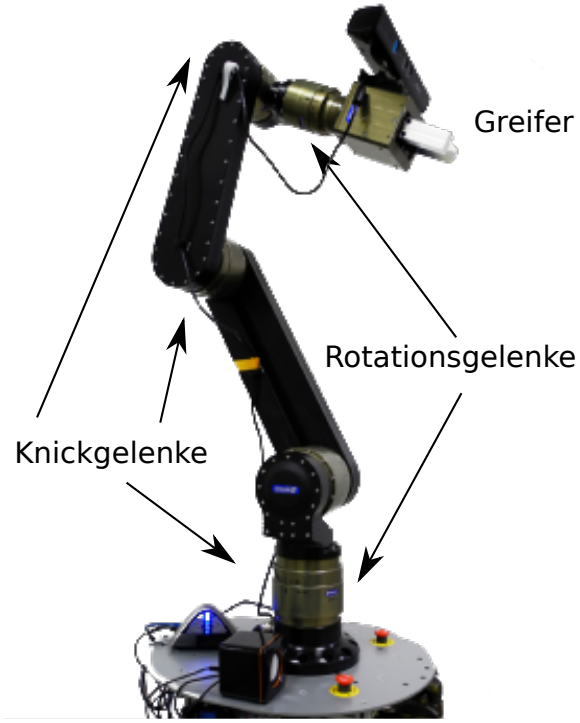


Abbildung 2.10: *Gelenke des Assistenz Roboterarm*

Zwei-Finger-Greifer

Bei dem Greifer handelt es sich um einen servoelektrischen Zwei-Finger-Parallel-Greifer, Typ PG Baugröße 70 der Firma *Schunk*. Er hat ein Eigengewicht von $1,7kg$. Das empfohlene Werkstückgewicht beträgt $1Kg$. Die Greifkraft beläuft sich zwischen minimal $30N$ und maximal $200N$, wobei diese von der Fingerlänge abhängig ist (siehe Abb. 2.11). Daraus resultiert auch die maximal zulässige Fingerlänge von $140mm$. Die maximale Greifbreite ist $7cm$. Die

[2] Lightweight Arm, auf Deutsch Leichtbauarm

[3] Die Wiederholgenauigkeit, auch Präzision, ist wie die Positioniergenauigkeit eine Kennwert der Genauigkeit bei Robotern

Wiederholgenauigkeit beträgt hier $0.05mm$. Die maximale Geschwindigkeit ist $82mm/s$ mit einer maximalen Beschleunigung von $328mm/s^2$ [77].

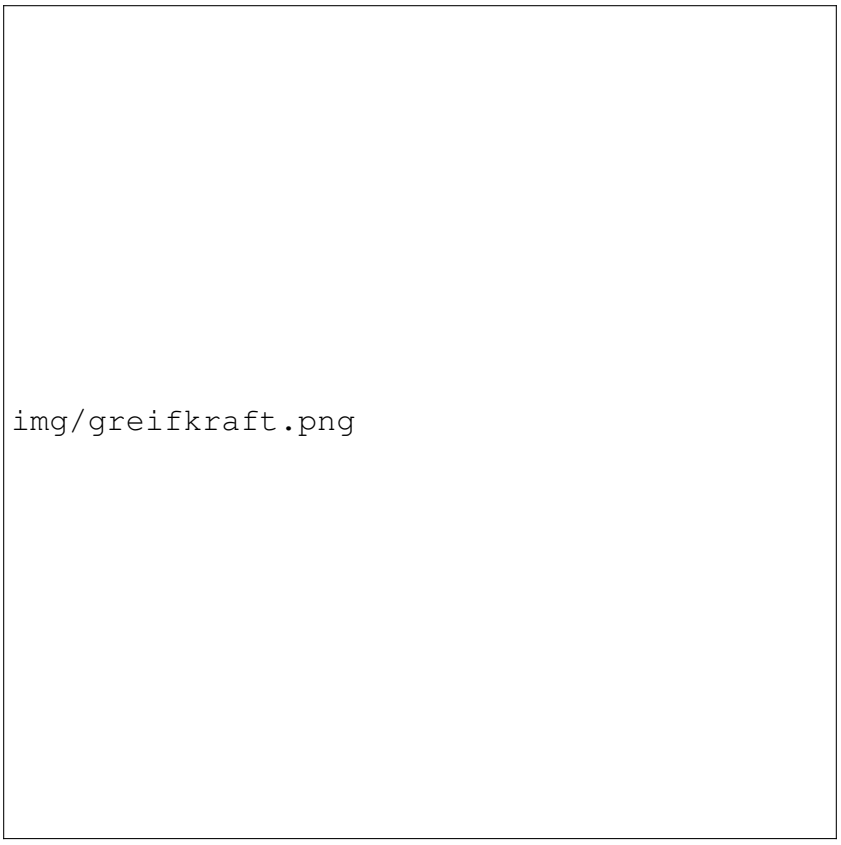


Abbildung 2.11: *Greifkraft in Abhängigkeit der Fingerlänge [77]*

2.4.2 Robot Operating System (ROS)

Wie im Abschnitt Hardware beschrieben handelt es sich bei dem On-Board-Computer des *SCITOS G5* um ein übliches System mit x86-Architektur. Auf diesem System ist das Betriebssystem Fedora, eine Linux Distribution, installiert. Dieses System beinhaltet das ROS Framework.

Es ist eines der am weitesten verbreiteten auf dem Markt [vgl. 34, S. 40] und ermöglicht eine Komponenten-basierte Entwicklung unter Wiederverwendung bereits bestehender Hardware- und Softwarekomponenten. Durch ein solches System verspricht es einen großen Fortschritt bei der Entwicklung von neuen Assistenzrobotern und deren Anwendungen zu erlangen [vgl. 94, S. 46].

Dieses und die Tatsache, dass die bereits vorhandenen Software-Module für die Steuerung und die Handhabung des *Scitos G5* mit ROS realisiert worden sind [vgl. 74, S.15], spricht für die Verwendung von ROS in dieser Arbeit.

Bei ROS handelt es sich um ein *open-source* Framework für Roboter [66], welches 2007 am *Artificial Intelligence Labaratory* der Stanford Universität im Zuge des Projekts *STAIR 4*[4] entwickelt worden ist und dessen Weiterentwicklung seither durch das Robotikunternehmen *Willow Garage 5* vorangetrieben wurde. Seit dem Jahr 2013 wird diese Aufgabe von der *Open Source Robotics Foundation 6* (OSRF) übernommen. Es liefert verschiedenste Roboter-Bibliotheken und eine Vielzahl an Programmen und Anwendungen um diesen zu steuern [45][74]. Des Weiteren lassen sich die unterschiedlichsten Software Komponenten miteinander Verbinden und können so untereinander Kommunizieren [60]. Im Folgenden wird diese Kommunikation im Abschnitt ROS Kommunikationskonzept beschrieben.

ROS Kommunikationskonzept

Wie bereits erwähnt beinhaltet ROS Bibliotheken, Schnittstellen und Werkzeuge für die Entwicklung diverser Anwendungsgebiete im Bereich der Robotik. Jede Funktion ist dabei in ein *Node* gekapselt. Dieses erlaubt es hoch modular zu bleiben. Jeder dieser *Nodes* befinden sich im so genannten *ROS Computation Graph*, welcher die einzelnen Module hält (siehe Abbildung 2.12).

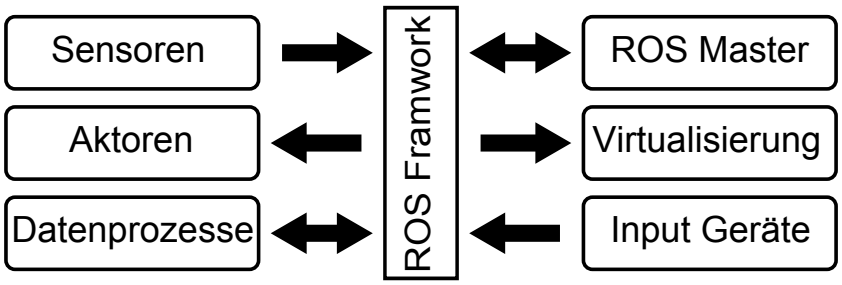


Abbildung 2.12: *ROS Computation Graph, ein Peer-to-Peer Netzwerk mit einem zentralen Koordinator, bezeichnet als ROS Master [45].*

ROS-Nodes kommunizieren in der ROS-Umgebung untereinander um Daten auszutauschen. Zur Kommunikation stehen standardisierte *Topics* (*Kanäle*) und *Services* zu Verfügung. Die Vermittlung der *Nodes*, *Services* und *Topics* untereinander erfolgt über einen zentralen *ROS-Master*.

Für die Teilnahme an der Kommunikation ist eine Anmeldung am *ROS-Master* nötig. Beim Start meldet sich ein *Node* mit den von ihm angebotenen *Topics* und Services am zentralen *ROS-*

[4] **STAIR** (**ST**anford **A**rtitificial **I**ntelligence **R**obot) - `http://stair.stanford.edu`

Master an. Dieser fungiert als Vermittler und speichert die Registrierung von *Nodes*, *Topics* und *Services*.

Topics sind namentlich identifizierte Datenkanäle, über welche *Nodes* miteinander kommunizieren und Daten austauschen. Jedes *Topic* hat einen festen Nachrichtentyp, welcher durch den sendenden *Node* festgelegt wird. Hierbei können zu jeder Zeit mehrere *Nodes* Daten senden (*publishen*) und empfangen (*subscriben*). Der Versand von Nachrichten über *Topics* erfolgt asynchron [74].

Wie in Abbildung 2.13 dargestellt, können *Nodes* nicht direkt mit anderen *Nodes* Kommunizieren. Stattdessen *subscriben* sie zu einem *Topic*, dessen Daten sie interessieren. Nodes die hingegen Daten produzieren, stellen ihre Daten über *Topics* für andere *Nodes* öffentlich (*publischen*). Interessiert sich ein *Node* nun für ein *Topic*, so stellt er eine Anfrage an den ROS-Master, der ihn an den entsprechenden *Node* vermittelt. Die anschließende Verbindung und der Datenaustausch erfolgt danach direkt zwischen den *Nodes*. Der *ROS-Master* greift nur dann informierend ein, wenn sich an den registrierten Informationen Änderungen ergeben.

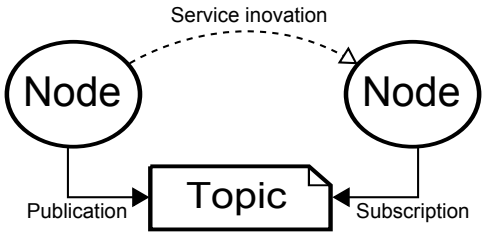


Abbildung 2.13: *Die Basis des ROS Kommunikationskonzepts zwischen zwei Nodes basierend auf Topics und Services [65].*

3 Analyse

In diesem Kapitel wird das gesamte Sensorsystem, von den einzelnen Sensoren bis zum Zielsystem, analysiert. Dabei stehen die Geschwindigkeit der Erfassung und Übertragung, neben der Einfachheit des Systems, besonders im Fokus. Auf die einzelne Sensorik wird in diesem Teil der Arbeit nicht ausführlich eingegangen, da dieses bereits in [13] und [15] getan wurde.

Als Referenz stehen nicht nur verwandte Arbeiten wie [82] zur Verfügung, sondern es wird auch hier, wie auch schon in [15], [16] und vielen anderen Arbeiten die sich mit dem Gebiet befassen, die menschliche Hand mit seinen Tastsinnen und Bewegungsabläufen, als Inspiration genommen. Dieses gibt einen guten Einstieg, um das richtige Verhältnis aus Geschwindigkeit, Präzision und Genauigkeit zu erlangen. Damit werden beste Voraussetzungen geschaffen um das übergeordnete Ziel, die Erweiterung eines Assistenzroboters im menschlichem Umfeld (vgl. 1.2), zu erreichen. Denn genau dieses beschreibt ein taktiles Sensorsystem dem menschlichen Tastsinn nachzubilden, um ihm zu ermöglichen, durch bestimmte technische Elemente mechanische Berührungen wahrzunehmen und so in einer menschlichen Umwelt optimal agieren zu können.

Vor dieser Exkursion in das menschliche Greif- und Tastverhalten, werden zum Einstieg einige verwandte und vergleichbare Arbeiten vorgestellt. Diese sollen erste Einblicke in die Materie und den stand der Technik geben.

Aus diesen beiden Sektionen wird nachfolgend die Problemstellungen aufgestellt, die aus den bereits gewonnen Erfahrungen und Einblicken in die verschieden Systeme gewonnen wurden. Durch die anschließend weiterführende Analyse der verschiedensten taktilen Sensoriken und Übertragungssysteme, wird versucht einen Lösungsansatz zu finden. Bei allen dieser Analysen ist es wichtig, nie das Gesamtsystem und deren Ziele aus den Augen zu verlieren.

Abschließend folgt eine verbindliche Aufstellung der Anforderungen an das Sensorsystem dieser Arbeit.

3.1 Verwandte und vergleichbare Arbeiten

In diesem Abschnitt wird eine Auswahl von verwandten und vergleichbaren Arbeiten vorgestellt, die sich mit den für diese Arbeit relevanten Themen beschäftigen.

Das Gebiet der taktilen Wahrnehmung für Roboter steht seit Jahren im Fokus und es gibt eine Menge an verschiedenster Arbeiten und Forschungsgruppen die sich damit beschäftigen. Auch die Schlupferkennung (*slip detection*) mit Hilfe von taktilen Sensoren, zum autonomen und sicherem Greifen, rückt immer weiter ins Interesse. Dabei wurden eine Vielzahl an neuen unterschiedlichsten Sensoren entwickelt. Der größte Anteil dieser Arbeiten legt das Hauptinteresse dabei auf die Hardware bzw. die Entwicklung des Sensors. In dieser Arbeit spielen nicht nur die reinen Aspekte der Sensorik, wie Aufbau und Material, sondern auch viel weitreichende Dinge, wie die Übertragung von Daten und die Wahl einer gescheiten Prozessoreinheit, eine wichtige Rolle. Aus diesem Grund, wird neben den verwandten Arbeiten zur reinen Sensorik, auch auf Arbeiten eingegangen, in denen der Schwerpunkt in eben genau diesen anderen Aspekten liegt.

Die Auswahl an Arbeiten soll eine Vorstellung davon geben, wie der Stand der Technik ist. Dabei liegt der Fokus allerdings ausschließlich auf den Teilen der Arbeiten, die für diese Arbeit relevant sind. So handelt es sich vor allem um die Sensorik (Material und Anzahl), die Übertragung, die damit verbundene Geschwindigkeit und Verarbeitung der Daten.

Schon früh demonstrierte unter anderem Tremblay in [91] wie wirksam die frühe Schlupferkennung ist. So konnte mithilfe eines einfachen Drucksensors an der Fingerspitze der Schlupf kontrolliert werden. Zur Überwachung wurde dabei ein Controller mit einer Operationsfrequenz von $320Hz$ eingesetzt.

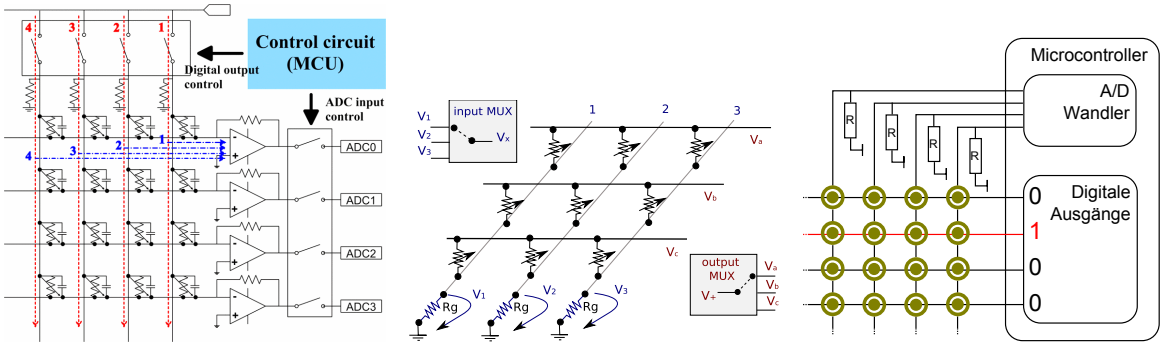
Eine weitere Arbeit [27], unter anderem von Damian, beschreibt einen Sensor aus einer $8x4cm$ großen, mit Noppen besetzter künstlicher Haut. Unter dieser $4mm$ dicken Haut befinden sich zwei FSR Sensoren um den Druck, zur Schlupferkennung aufzunehmen. Die Sensordaten werden dabei mit einer Frequenz von $80Hz$ abgefragt. Damit kann eine Rutscherkennung bei einem Druck von mindestens $70N$ und einer maximalen Rutschgeschwindigkeit von $15mm/s$ gut erkannt werden.

In [23] wurde ein Schlupfsensor basierend auf entstehenden Vibrationen beim rutschen eines Objektes entwickelt. In einer Testumgebung wurde auf einem Schmirgelpapier, mit der stärke von $P100$, ein Schlupf, bei einer Geschwindigkeit von $0.025ms^{-1}$ und einem konstantem Druck von $1.25N$, erkannt. Um die Signale einfach zu verarbeiten und die Daten zu sammeln, wurde eine *National Instruments 6036E data acquisition card (DAQ)* genutzt. Somit konnte eine Abtastrate von $10kS/s$ erreicht werden, was einer Frequenz von $5kHz$ entspricht. Ein ähnliches Verfahren wird auch in der Arbeit [97] vorgestellt.

Ein ganz anderer Ansatz ist das Einsetzen von optischen Sensoren. Hier wird die taktile Erfassung durch die sich ändernde Reflexion von Lichtstrahlen gemessen, die sich durch die Verformung der Oberfläche ergeben. Dieses Verfahren wurde bereits 1988 in [7] vorgestellt. Ein weiterer interessanter und deutlich fortschrittlicherer Ansatz, der auch im Bereich der Optik liegt, ist in [62] vorgestellt. Bei diesem System wird mithilfe einer LED, Licht erzeugt. Die durch

dieses Licht erzeugten Reflexionen werden mit einem CMOS[5] Sensor, bis zu 1500 mal pro Sekunde, erfasst. Dadurch ist das System in der Lage das Rutschen des Gegenstandes zu detektieren.

Unter anderem beschreibt Byungjune in der Arbeit [20] einen taktilen Sensor zur Schlupferkennung mithilfe eines *PVDF* Materials[6]. Das Sensorsystem besteht aus 24 Sensoren, mit jeweils der Größe von $0.5mm * 0.5mm$, welche in einer Matrix aufgebaut sind. Zu dem System gehört ein *Signal Processing Board*, welches mithilfe von A/D-Wandlern eines Mikrocontrollers (C8051F311) die Sensordaten digitalisiert. Aufgrund der Matrixschaltung (siehe Abbildung 3.1(a)) werden nur vier A/D Wandler benötigt. Die erfassten Daten werden anschließend von dem *Signal Processing Board* per *RS-232* Schnittstelle an den PC übertragen, an dem diese dann ausgewertet werden können. Auch mit dieser Methode soll erfolgreich Schlupf erkannt werden.



(a) (b) (c)

Abbildung 3.1: *Matrix Schaltungen von [20] in (a), [29] in (b) und [15] in (c)*

Eine Arbeit in der das Kernziel zwar nicht die Schlupferkennung ist, aber auch eine Matrix-Schaltung eingesetzt ist, wird in [29] vorgestellt. Um eine hohe demultiplexing Geschwindigkeit und hohe Anzahl an Eingängen zu ermöglichen, wird hier der 16-Bit Mikrocontroller *PIC33FJ256* verwendet. Durch geschicktes demultiplexing wird für jedes Sensormodul nur jeweils ein analoger Eingang und digitaler Ausgang benötigt. Das beschriebene *data-acquisition module* kann taktile Arrays mit bis zu 512 Zellen, bei 100 fps und acht Bit Daten pro Zelle auswerten, welche dann per USB-Verbindung an den Computer übertragen werden. In diesem Fall wird dieses Modul genutzt um 64 Zellen (8x8) auszuwerten, um eine Abtastung aller Daten in $10ms$ zu ermöglichen. Eine Erhöhung der Abtastrate auf bis zu $400Hz$, z.B. für eine Schlupferkennung, wäre laut dem Autor der Arbeit möglich. Wie in [87] beschrieben, ist es aber auch schon bei $100Hz$ möglich Schlupf, mit einer Erfolgsquote von bis zu $98,8\%$, zu erkennen.

[5] **CMOS**: **C**omplementary **M**etal-**O**xide-**S**emiconductor

[6] **PVDF**: Ein piezoelektrisches Material aus Polyvinylidenfluorid, welches mechanischer Energie in elektrische Energie, und umgekehrt, umwandeln kann.

Ein taktiles *Low-Cost* Sensor Array System mit einer Auflösung von $16 * 16$ *Taxel*[7] und einer Abtastfrequenz von $10Hz$, wurde unter anderem von Schopfer in [76] vorgestellt. Jedes Sensor *Taxel* kann dabei Druck zwischen 4 bis $120kPa$ messen und liefert das Ergebnis in dynamischen $12Bit$ Werten an die zentrale Verarbeitungseinheit.

Eine Arbeit die einen ganz anderen Schwerpunkt hat, wurde von Nilsson in [53] beschrieben. Hier wurde ein taktiler Sensor entworfen, der mit möglichst wenigen Leitungen zur Ansteuerung und Datenerfassung auskommen soll. Zusätzlich wird auch eine einfache, mit kostengünstigen Bauteilen umgesetzte Konstruktion angestrebt. Dennoch soll es aber auch das komplette Druckprofil erfassen können. Entstanden ist ein System, in dem es durch elektronische Filter möglich ist, mit letztendlich zwei Leitungen, das Ziel der Arbeit (siehe Abbildung 3.2) zu verwirklichen. Dieses funktioniert allerdings nur durch große Einbußen in der Übertragungsgeschwindigkeit.

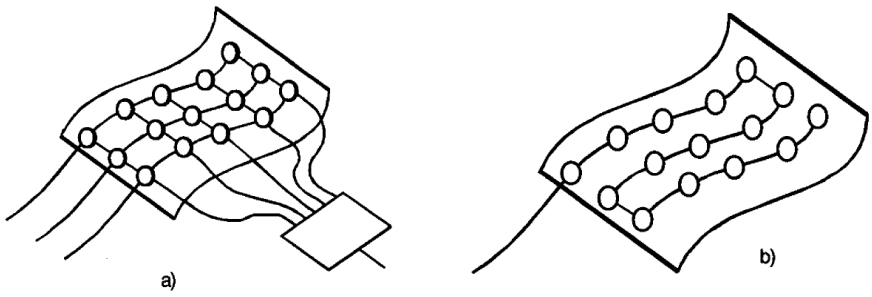


Abbildung 3.2: *(a) Sensoren Ansteuerung durch ein demultiplextes Array und (b) eine Substitution der Leitungen durch eine Einzige, zur Datenerfassung (vgl. [53]).*

Ein flexibler taktiler Sensor wird in [21] beschrieben. Er besteht aus einem System mit zwei Sensoren zur Schlupferkennung und einem *readout circuit module* zur Datenerfassung und Weiterleitung der Signale an den Computer. Dieses Module besitzt einen Mikrocontroller (*LM3S811, Texas Instruments Inc.*), um die analogen Daten zu digitalisieren und so mit einer *Framerate* von $40kHz$ zu erfassen. Über eine USB-Verbindung (*RS232 to USB*) können bis zu 100 Werte pro Sekunde an den Computer übertragen werden. Ein weiterer flexibler und deutlich schnellerer taktiler Sensor zur Schlupferkennung wird in [92] vorgestellt. Dieser Arbeitet in einer Testumgebung, mit einer Abtastrate von bis zu $1kHz$ und nutzt zur Auswertung *MATLAB/Simulink (The MathWorks, Natick, MA)*.

In [79], [81] und [82] wird u. a. ein modulares Sensorsystem vorgestellt. Das Ziel dieses ist es existierende taktile Sensoren, mit typischerweise einer Abtastrate von unter $100Hz$ bei einer Größe von $16 * 16$ Aufnahmepunkten, deutlich zu verbessern. So ist dieses modulare System, am Ende der Arbeit, in der Lage mit einer Taktrate von bis zu $1300Hz$ zu arbeiten, bei einer guten Ortsauflösung von $5mm$. Wie in Abbildung 3.3(a) dargestellt, besteht ein Modul aus einem

[7] **Taxel**: **TA**ctile pi**XEL** - Ein taktiles Sensor-Element in der Robotik

$16x16$ Sensor-Array und einem *PIC32* Mikrocontroller. Dieser nimmt mithilfe von 16 externen AD-Wandlern, die jeweils 16 Ports besitzen, die Sensordaten über die SPI Schnittstelle auf. Sind alle 256 Sensordaten gesammelt, wird der ganze Datensatz über einen $12Bit$ parallelen Port an die Zentrale Einheit (*AVR32*) verschickt. Der Bus erlaubt es dabei mit geringerer Taktrate zu arbeiten und ist dadurch weniger gegen äußerliche Störungen anfällig, zusätzlich wird aber trotzdem eine hohe Datenrate gewährleistet. Die zentrale Einheit empfängt die Datensätze der vorhandenen Module und sendet diese, mithilfe des Protokolls *USB video device class*, über eine USB (*USB2.0*) Verbindung, an den Computer weiter. Die Abbildung 3.3(b) illustriert das beschriebene System, der parallelen Sensordaten Aufnahme, an einem Beispiel von zwei Modulen, in einem Zeit-Diagramm. Die Arbeit [75] zeigt wie auftretender Schlupf mit Hilfe dieses Sensorsystems erkannt und verhindert werden kann. Zur Schlupferkennung wird dazu die Schnelle Fourier-Transformation (FFT) genutzt, um die Daten vor zu verarbeiten. Des Weiteren ist es durch dieses Sensorsystem möglich, Rückschlüsse auf das Material des gegriffenen Objektes zu schließen.

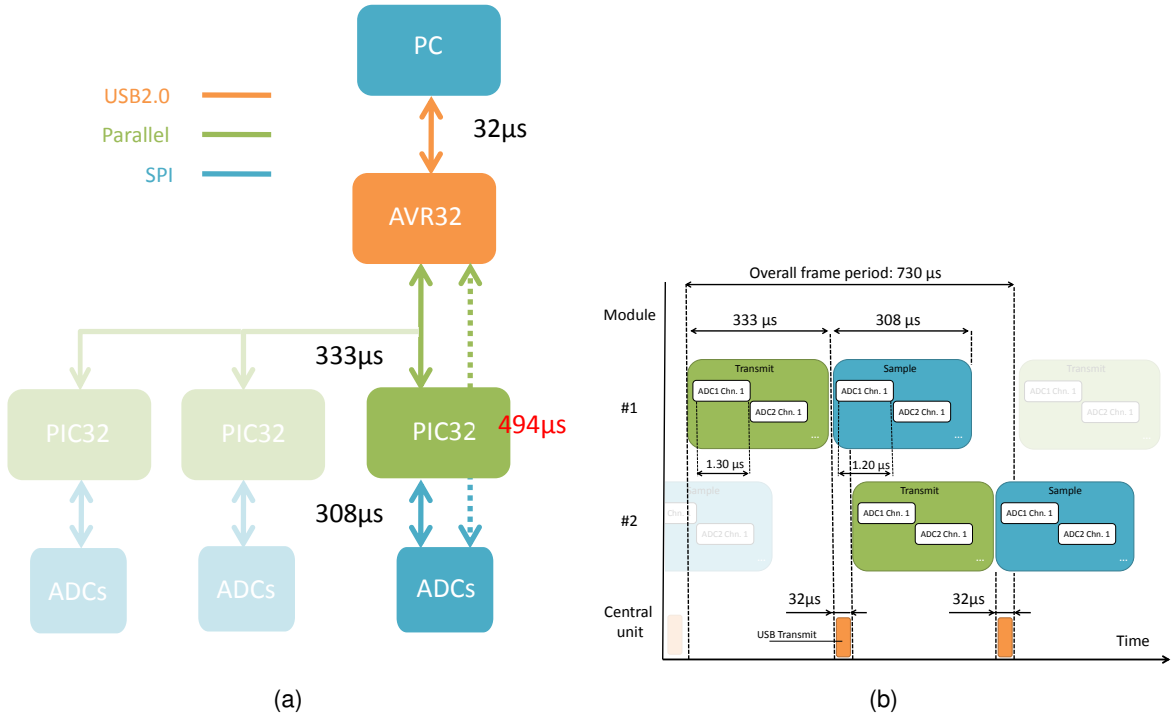


Abbildung 3.3: *(a) Parallele Datenerfassung und Datenverarbeitung in Kommunikation-Diagramm und (b) Zeitablauf-Diagramm, aus der Arbeit [79].*

Die hier aufgeführten Sensorsysteme umschließen nicht den gesamten Forschungsumfang, sondern geben nur einen kleinen Überblick in welche Richtung die Entwicklungen gehen und dienen, als Anregung für diese und fortlaufende Arbeiten. Eine ausführliche Sammlung von

aktuellen Arbeiten und Techniken der taktilen Sensoren im Bereich der Robotik, sind unter anderem von Dahiya in [25] und [26] aufgeführt.

3.2 Menschlicher Tastsinn

Die menschliche Haut ist das Organ mit der größten Fläche am menschlichen Körper. Einer der wichtigsten Aufgaben dieser ist der Kontakt zur Umgebung, in dieser zu erkunden und in ihr zu manipulieren und spielt deshalb auch im Bereich der Service- und Assistenzroboter eine wichtige Rolle. Laut [73] ist die Hand das wichtigste Sinnesorgan unseres Körpers. Genauer gesagt ist das wichtigste Tastorgan die Haut an der Palmarseite [8] der Hand; das beste Auflösungsvermögen findet sich dabei an den Fingerspitzen. Diese Tatsachen machen einen, wenn auch nur einen kleinen, Einblick in den menschliche Tastsinn zu einer unabdingbaren Voraussetzung.

Die menschliche Haut besitzt u. a. vier unterschiedliche Typen von taktilen Sensoren (Übersicht in 3.1 dargestellt), den so genannten Rezeptoren, welche sich u. a. in ihrer Struktur, Empfindlichkeit, Adaptionsgeschwindigkeit und rezeptiven Feldgröße unterscheiden. So können Tastempfindungen schon auf der Rezeptorebene unterschieden werden.

Rezeptoren	**Merkel**	**Ruffini**	**Meissner**	**Pacini**
Anzahl	25%	19%	43%	13%
Dichte (Units/cm²) Ø	40	15	60	15
Typ	statisch	statisch	dynamisch	dynamisch
Adaptivität	langsam	langsam	schnell	schnell
Reagiert auf	Druck	Berührung	Dehnung	Vibration
Lage	oberflächlich	tief	oberflächlich	tief
rezeptive Feldgröße	klein	groß	klein	groß
Innervationsdichte	hoch, variabel	gering, konstant	hoch, variabel	gering, konstant
Reiz-Frequenz (Hz)	0.4-3	100-500+	3-40	40-500+

Tabelle 3.1: *Taktile Sensoren (Rezeptoren) der menschlichen Haut und deren Eigenschaften (angepasst aus [72] und [73].*

Druck und Berührung werden hauptsächlich von zwei Rezeptortypen wahrgenommen. Zum einen die *Ruffini Kolben* welche großflächige Berührungen und Druck übermitteln, zum anderen die *Merkel-Zellen*, kleine Tastscheiben in der oberen Hautschicht, die auf genau lokalisierte Berührungen reagieren. Durch das Zusammenspiel der unterschiedlichen Berührungs- und Druckrezeptoren können Intensität, Dauer und Bereich der jeweiligen Berührung genau

[8] Als Palmarseite bezeichnet man die Beugeseite der Hand

bestimmt werden. Dabei reagiert die Haut mit Impulsfrequenzen, welche entlang der schnell leitenden Nervenbahnen über verschiedene Schaltstationen an die sensorischen Bereiche im Gehirn weitergeleitet, verarbeitet und als Tast- und Druckempfindung wahrgenommen werden und vermitteln so die Stärke und Geschwindigkeit eines Reizes.

Die menschliche Haut ist in der Lage die Druckstelle recht genau zu lokalisieren. Das liegt daran, dass der Druckrezeptor nur einen sehr kleinen, scharf umgrenzten Bereich der Haut, sein rezeptives Feld, an eine sensorische Zelle im Gehirn weiterleitet. Die Empfindlichkeit der Druckrezeptoren ist dabei, wie bereits erwähnt, besonders an den Fingerspitzen sehr hoch. Hier besteht eine Tastschärfe an den Fingerspitzen von etwa $1mm$, womit die Fähigkeit, räumlich eng benachbarte Berührungsreize als separate Reize wahrzunehmen, ermöglicht wird. An den Handflächen ist diese simultane Raumschwelle schon in etwa $3-5mm$ [73]. Die Geschwindigkeit, mit der die vier Rezeptoren reagieren und arbeiten, variiert sehr stark. So ist aus der Tabelle 3.1 zu entnehmen, dass die *Merkel-Zellen* mit einer Rate von $0,4-3Hz$ arbeiten und die *Ruffini Kolben* hingegen bis über $500Hz$.

3.3 Menschliche Greifanalyse

Ein weiterer Aspekt der eine große Rolle beim Greifen eines Objektes darstellt, ist das Verhalten, welches durch eine Greifanalyse mit dem Vorbild Mensch durchgeführt wird. Greift eine menschliche Hand ein ihm unbekanntes Objekt, wobei in diesem Fall die Erfahrung und intuitive Einschätzung vernachlässigt wird, wird zunächst eine Druckkraft auf das Objekt aufgebaut, die so stark sein muss, dass der Reibungskoeffizient des Objektes und die Gewichtskraft ein rutschen verhindern ((*Load Phase*)). Sollte die Kraft nicht ausreichen, erkennen die sogenannten *Meissner-Körperchen*[80], im Falle des Roboters wären das die taktilen Sensoren, das rutschen des Objektes und der Druck wird erhöht. Dieses wird so lange wiederholt, bis sich das Objekt sicher in der Hand befindet. Erst nach dem stabilen Griff des Objektes, kann dieses in seiner Umwelt sicher bewegt werden (*Lift Phase*). Genau dieses Verhalten geht es nachzubilden, um dem Roboter das autonome Greifen von unbekannten Objekten zu ermöglichen. Hierbei geht es im Grunde um die Schlupf Detektion und Griffkontrolle. Die nachfolgende Abbildung 3.4 zeigt nochmals ausführlich die eben angesprochene Taktile Sensorkontrolle bei der Objektmanipulation eines Menschen.

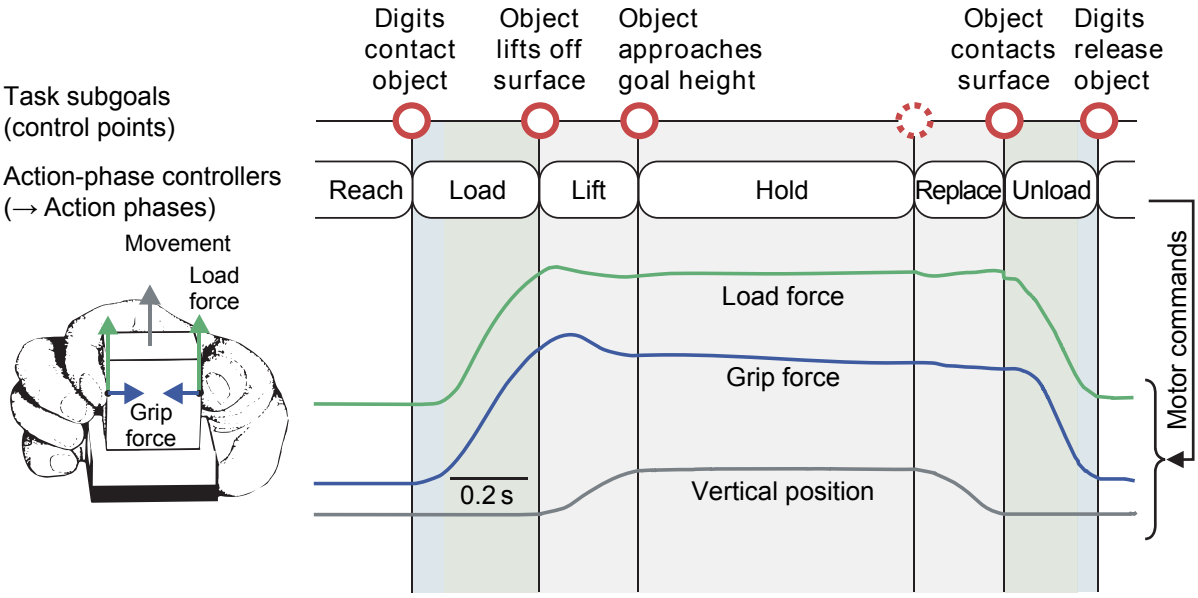


Abbildung 3.4: *Taktile Sensorkontrolle bei der Objektmanipulation eines Menschen. Eine Person greift und hebt ein Testobjekt von einem Tisch, hält es in der Luft und setzt es wieder ab [39].*

Problemlösungen für die zuvor angesprochenen Rutschdetektion, sind zum Beispiel der Einsatz eines optischen Maussensors [12], die Erfassung der reibungsbedingen Vibration mit Hilfe von Beschleunigungssensoren [47] oder durch ein großes taktiles *highspeed* Sensorarray [82]. In den ersten beiden Beispielen wird die erwähnte Sensorik durch Druckkraftsensoren unterstützt, es handelt sich also um eine Kombination aus verschiedenen Sensoren. Das letzte Beispiel [82] schafft die Problemlösung, des rutschischeren Greifen alleine mit einem Sensortyp und ist deshalb für diese Arbeit besonders interessant.

3.4 Problemstellung

Um das Ziel der Arbeit zu erreichen, gilt es diverse Problemlösungen zu finden. Dazu werden nun nachfolgend, auch mithilfe der bereits getätigten Analyse die Problemstellungen an ein taktiles Sensorsystem zur Schlupferkennung, erfasst und definiert.

Da für eine Schlupferkennung, wie sich gezeigt hat, eine hohe Taktrate (deutlich über $100Hz$) benötigt wird, müssen die Daten schnell erfasst werden und ebenso schnell an das Zielsystem gelangen. Hinzu kommt noch, dass es sich dabei in den meisten Fällen noch zusätzlich um eine hohe Anzahl an Sensordaten handelt. Um dieses Problem etwas besser zu veranschaulichen wird es in drei Kernprobleme unterteilt:

Zum einen die **Erfassung** der rohen Sensordaten. Dabei spielt der taktile Sensor selbst zuerst eine wichtige Rolle. Welches Material ist am besten geeignet und mit welcher Methode sollen die Druckkräfte erfasst werden. Die erfassten Daten müssen dabei mit einer hohen Präzision aufgenommen werden und dürfen nicht durch fremde Störungen, aus dem menschlichen Umfeld, beeinflusst werden. Bei der Erfassung der Sensordaten handelt es sich in den meisten Fällen, z.B. [81] [82] [46] [96] oder [79], nicht nur um einen Datenwert, sondern in der Regel um einen ganzen Satz an zu erfassenden Daten. Auch diese Daten müssen aus den vielen Sensoren schnell, zuverlässig und möglichst parallel ausgelesen und erfasst werden. Des Weiteren muss jeder Wert eindeutig identifizierbar sein und in einer Auflösung vorliegen, in der mehrere Bereiche von Druckkräften eindeutig zu identifizieren sind.

Ein weiterer Punkt, welcher wohl eine ebenso große Herausforderung darstellt, ist der **Transport** der Sensordaten. Auch dieser muss in einer hohen Datenrate erfolgen. Dazu müssen die Daten an den richtigen Stellen verpackt werden und über das für den Transport gescheite Übertragungsmedium geschickt werden. Zu analysieren gilt es hier, welche zu übertragenden Daten relevant sind und welcher Inhalt benötigt wird. Des Weiteren ist bei der Wahl des Übertragungsmediums, die Leitungslänge und maximale Datenrate zu berücksichtigen, damit zuverlässig übermittelt werden kann. Die Verpackung der Daten sollte so gewählt werden, dass sie möglichst wenig *Overhead* bildet und zum anderen auch von der Gegenstelle leicht und schnell verarbeitet werden kann.

Der letzte Punkt ist die **Verarbeitung** der Daten. Wie eben schon erwähnt, sollten die Daten so verpackt werden, dass der Teil in dem Zielsystem, die Sensordaten leicht verarbeiten kann. Ein weiterer wichtiger Punkt hierbei ist die eindeutige Identifizierung. Das System muss nicht nur genau wissen zu welchem Sensormodul die Daten gehören, sondern muss auch jeden einzelnen Messwert eindeutig dem richtigen Sensor zuordnen können. Auch die Reihenfolge der Daten muss beständig sein und möglichst zeitnah bzw. im besten Fall sogar in Echtzeit vorliegen.

Allgemein wird in dem Ziel der Arbeit auch definiert ausschließlich auf gut verfügbare, kostengünstige und einfachste Komponenten, Bauteile und Materialien zurückzugreifen. So gibt es für viele der Ziele und Lösungen der bereits genannten Problemstellungen Lösungen, diese sind aber in der Regel mit teuren Hightech Teilen umgesetzt. Deshalb stellen auch diese genannten Ziele eine Palette an weiteren zu lösenden Problemen dar.

Diese drei Punkte gilt es nun nachfolgend zu analysieren und später mithilfe eines Konzepts zu lösen, um anschließend eine brauchbare Umsetzung zu erhalten.

3.5 Taktiler Sensor

Wie die verwandten Arbeiten [46] [58] [79] [81] [82] und [83], gezeigt haben, bauen viele Sensorsysteme auf ein Array aus mehreren Taxeln auf. Da sowohl in den Arbeiten [13] und [15]

Erfahrungen gesammelt wurden, als auch alle gesetzten Ziele dieser Arbeit in dieses Konzept passen, wird an dieser Stelle der Fokus besonders auf taktile Sensor Arrays gelegt. Allerdings wird aufgrund der Tatsache, dass in diesem Bereich der taktilen Sensorik schon ein großes Angebot und ausführliche Analysen zur Verfügung stehen, wird dieser Teil, im Vergleich zu anderen Abschnitten der Analyse, relative kurz gehalten. Für eine detailliertere Analyse mit allen bekannten Verfahren und Methoden wird in [25] ausführlich eingegangen.

Großen Einsatz finden nicht nur piezoelektrische sonder auch piezoresistive Arrays (zum Beispiel in [46], [58] und [83]), die durch ihre Elastizität und Flexibilität in allen nur denkbaren Positionen anwendbar sind. Der Piezoelektrische Effekt, der hier ausgenutzt wird, erzeugt durch Verformung des Festkörpers eine Änderung der elektrischen Polarisierung und erzeugt damit elektrische Spannung, die einfach messbar ist. Somit ist der Sensor nicht nur durch eine leichte Schaltung anwendbar sondern sticht auch durch seine einfache Herstellung und variabler Form hervor. Als Nachteil ist zu erwähnen, dass diese Sensoren sehr fragil und temperaturempfindlich sind. Des Weiteren sind die Signale, in diesem Fall die Spannungswerte, nicht statisch.

Ähnlich wie die piezoresistive Arrays, verhalten sich die kapazitiven Arrays (zum Beispiel in [37]), wo die Messung der Kraft Zu- und Abnahme durch die Kapazität bestimmt wird. Dabei lassen sich im Gegensatz zu den piezoresistiven, sehr empfindlich statische Werte messen. Die etwas komplexere Schaltung macht den Einsatz allerdings nicht in allen Lagen möglich.

Ein weiteres Verfahren verwendet die Induktivität welches zum Beispiel in [31] beschrieben wird. Dieses eignet sich laut [71] nicht gut für die taktile Wahrnehmung eines Greifers an einem Roboter und finden in dieser Arbeit keine weitere Betrachtung.

An dieser Stelle wird auch noch keine endgültige Entscheidung getroffen, welche Sensoren in der Arbeit eingesetzt werden. Allerdings konnte, durch die kurze Analyse und besonders durch den Abschnitt der vergleichbaren Arbeiten (siehe 3.1), von verschiedensten Umsetzungen diverse Anforderungen an die Sensoren gestellt werden. Zum einen sollen die eingesetzten taktilen Sensoren in die Oberfläche des Greifers integriert werden und den Ort der Krafteinwirkung lokalisieren können. Die Sensoren und deren Außenhaut sollten außerdem robust genug sein und eine gewisse Temperaturbeständigkeit vorweisen. Um eine hohe Dichte, besonders an Stellen wie den Fingerspitzen zu gewährleisten, sollten sie klein und leicht sein (bei Menschen befinden sich dort ca. 150 Rezeptoren $/cm^2$ [80]). Weitere Anforderungen werden im Laufe der Analyse erarbeitet und im Abschnitt der Anforderungen detailliert beschreiben.

3.6 Datenübertragung

Die Datenübertragung bezieht sich auf den Transfer der Sensordaten. Dabei wird auf die Geschwindigkeit bzw. Datenübertragungsrate, das Übertragungsverfahren und die Struktur des Übertragungssystem, eingegangen. Der Fokus bleibt auf einem System, das in der Lage sein

soll, Schlupf zu erkennen. Die drei eben erwähnten Punkte werden nachfolgend ausführlich analysiert.

3.6.1 Geschwindigkeit

Die Übertragungsgeschwindigkeit hängt von mehreren Faktoren ab. Beginnend ist zu analysieren, wie schnell der Sensor reagieren muss, um das rutschen eines zu greifenden Gegenstands zu erkennen. Zur ersten Einordnung können die Frequenzbereiche der menschlichen Rezeptoren (siehe Tabelle 3.1) genutzt werden [26]. Besonders Interessant sind hier die *Merkel* und *Ruffini* Rezeptoren, da diese hauptsächlich für das Druck- und Berührungsempfinden zuständig sind. Des Weiteren zeigt [35] eine grobe Übersicht, diverse Zeitanforderungen für taktile Sensoren, u. a. auch dass eine Reaktionszeit, zur Erkennung eines rutschenden Objekts, von mindestens $10ms$ erforderlich ist. Es wird aber auch darauf hingewiesen, dass eine wünschenswerte Grenze in der Größenordnung von $1kHz$ liegen sollte. Auch, unter anderem von Dahiya, wird in [26] eine Aussage zu der erforderlichen Reaktionszeit getätigt. So wird auch dort eine Zeit von $1ms$, was einer Abtastrate von $1kHz$ entspricht, für die saubere Erkennung von Rutsch definiert. Ferner weißt er darauf hin, dass auch in einem Sensor-Array jeder Taxel diese Reaktionszeit aufweisen sollte. In [82] wird sogar eine Abtastrate von bis zu $1.9kHz$ (Bei der Verwendung eines $16 * 16$ Taxel Moduls) erreicht, um das rutschen eines zu greifenden Objektes zu detektieren. Dabei wird aber keine Aussage getätigt, ob diese hohe Taktrate notwendig ist.

Ein weiterer Faktor ist die Auflösung der Taxel und die daraus resultierende Anzahl der verbauten Sensoren. Des Weiteren spielt auch die Größe der gesamten Datenmenge, die übertragen werden muss, eine Rolle. Diese setzt sich zum einen aus der Anzahl der Taxel und zum anderen aus der Datengröße eines einzelnen Taxel Wertes zusammen. In vergleichbaren Arbeiten liegt die Anzahl der Taxel pro Sensor Array zwischen $100 - 300$ Stück, so sind es z.B. bei [79], [81] und [88] Arrays mit $16x16$ Taxel pro Sensor Modul. Auch hier spiegelt sich die Zahl annähernd der des Menschen wieder (vgl. 3.1). Bei der sogenannten Ortsauflösung[9], wird in [26] empfohlen, $1mm$ bei empfindlichen Stellen, wie z.B. den Fingern bzw. Fingerspitzen, zu verwenden. Hingegen sind bei weniger empfindlichen Stellen $5mm$ ausreichend. Außerdem wird bei einer Matrix-Anordnung der Taxel, zu einer Anzahl dieser in dem Größenbereich von $10 * 15$ geraten. Sinnvolle Datengrößen einzelner Taxel Größen sind im Bereich von 8-12bit (256-4096 Zustände) als ausreichend anzunehmen, was sich auch in Arbeiten wie [82], [79] und [81] wieder spiegelt.

[9] Räumlicher Abstand zwischen den einzelnen Taxel

3.6.2 Übertragungsverfahren

Für die Analyse einer geeigneten Übertragungstechnik sind einige Annahmen, auch wenn diese nicht endgültig sein müssen, zu treffen. Dieses hilft eventuell Möglichkeiten im Vorfeld und ohne detaillierte Betrachtung auszusortieren. Hierzu muss zum einen die Menge der Daten und die erforderliche Datenrate annähernd bekannt sein. Auch die Länge der Datenwege und die möglichen Übertragungsmedien sollten vorab grob bekannt sein. Diese Kriterien können dabei aus den vorherigen Sektionen der Analyse, besonders durch die verwandten Arbeiten, im gewissen Maße bereits entnommen werden. Dazu wurde in den Sektionen 3.1, 3.2 und 3.6.1 nötige Abtastraten der Datenerfassung und erforderliche Reaktionszeiten, zur Lösung der gesetzten Ziele aus 1.2, analysiert. Aus diesen Erkenntnissen werden nun nachfolgend einige Anforderungen definiert, die wie erwähnt bei der weiteren Analyse helfen sollen, jedoch für das später folgende Konzept nicht endgültig und verbindlich sein müssen.

Die Daten der Sensoren sollen mindestens alle $1,25ms$ am *Host* Computer aktualisiert werden. Das entspricht einer Frequenz von $800Hz$. Somit darf die Summe der Taxel aller Module auf dem Weg vom Sensor zum Endsystem die definierte Zeit nicht überschreiten. Um die ungefähr nötige Datenrate der einzelnen Module zu bestimmen, ist die Anzahl der Sensoren von essentieller Bedeutung. Auch hierzu wird anhand der Analyse aus den vorhergegangenen Sektionen eine Anzahl an Sensoren angenommen. Davon ausgegangen, dass es sich in diesem Fall um einen Zwei-Finger-Greifer handelt, wird eine Sensoranzahl pro Finger von 160 definiert. Zusätzlich wird davon ausgegangen, dass der einzelne Sensor einen 10 Bit Datenwert liefert. Anhand dieser Informationen ist nun die Datengröße eines einzelnen „Fingers“ bekannt:

$$D_{Finger} = 160Sensoren * 10bit = 1600bit \tag{3.1}$$

Da eine Frequenz von $800Hz$ festgelegt ist, müssen diese $1600bit$ also 800 mal pro Sekunde Übertragen werden. Aus diesen beiden Werten setzt sich die erforderliche Datenrate zusammen:

$$Datenrate = \frac{D_{Finger}}{Zeit} = \frac{1600bit}{0,00125s} = 1.280.000bit/s = 1600kbit/s = 1,6Mbit/s \tag{3.2}$$

Hierbei ist zu beachten, dass es sich ausschließlich um die Übertragung der Daten eines Moduls handelt. Da es sich bei diesem Wert allerdings nur um einen Richtwert handelt, der zur besseren Analyse der verschiedensten Übertragungstechniken dienen soll, reicht dieses erst einmal aus. Genauer und detaillierter wird hier in den Kapiteln Grundkonzept des Sensorsystems und der nachfolgenden Evaluierung der einzelnen Teilkomponenten eingegangen. Nachfolgend werden nun verschiedenste Übertragungstechniken auf ihre Eignung zur Lösung der

angestrebten Ziele begutachtet. Folgende Schnittstellenparameter und Eigenschaften sind dabei zu berücksichtigen:

1. Übertragungsrate, gemessen in Bits
2. Verdrahtungsaufwand
3. Störsicherheit bzw. Abschirmaufwand
4. Implementierungs- und Konstruktionsaufwand

Ein weit verbreitetes, serielles und einfaches Verfahren ist **UART** (*Universal Asynchronous Receiver Transmitter*). Die Daten werden als serieller Datenstrom mit einem festen Rahmen übertragen. Dieser besteht neben den reinen Datenbits (fünf bis neun) noch aus einem Start- und *Stop-Bit*, sowie einem optionalen *Parity-Bit* zur Erkennung von Übertragungsfehlern. Die Übertragungsgeschwindigkeit wird durch die Baudrate bestimmt, welche in vielen Fällen im Breich zwischen $9.600 bit/s - 115.200 bit/s$ liegt und maximal bis zu $500.000 bit/s$ betragen kann. Da selbst bei der maximale Bitrate, welche zugleich durch seine asymetrische Signalübertragung sehr fehleranfällig ist, die geforderte Frequenz von über $1,6 Mbit/s$ nicht erfüllt werden kann, ist diese Schnittstelle weniger geeignet. Des Weiteren würde es sich für eine Bus Struktur nur bedingt eignen, welches einem modularem Aufbau im Weg stehen könnte. Viele Arbeiten, wie zum Beispiel [6], [43] und [96], greifen dennoch auf dieses sehr einfache Verfahren zurück. Weitere einfache Schnittstellen, die Aufgrund ihrer zu geringen Bandbreite hier nicht in Frage kommen, sind unter anderem **I²C** und **Tow Wire**.

Eine differentiale (symmetrische) Übertragung wäre über den Schnittstellen-Standard **LVDS** (*Low Voltage Differential Signaling*) möglich. Dieser bietet neben seiner sehr hohen Datenrate, über $200 Mbit/s$, nicht nur ausreichend Durchsatz, sondern ist auch sehr robust. Auch hier liegt die typische Kabellänge im einstelligem Meterbereich, allerdings würde nur ein Bruchteil der maximalen Datenrate benötigt werden, womit die Störanfälligkeit auf längeren Wegen singt. Zwar handelt es sich hier um eine Punkt-zu-Punkt Verbindung, allerdings gibt es mehrere Standards (z.B. *Bus LVDS* oder *Multipoint LVDS*) die ein Bus-System ermöglichen und somit auch ein modulares System ermöglichen könnten. Dabei erfolgt der Leitungsaufbau immer als eine Linie und niemals als Stern mit etwaigen Stichleitungen. Da dieser Standard hauptsächlich in der Displayansteuerung zum Einsatz kommt, gibt es auf kleinen Mikrocontrollern kaum eine Möglichkeit diese Übertragungstechnik ohne großen Aufwand und Kosten zu betreiben. Trotzdem wird dieses Verfahren u. a. von [41] genutzt, um bei einem taktilen Greifer zur Unterwasser-Manipulation, die Kommunikation der einzelnen Verarbeitungsmodule zu ermöglichen. Dabei entsteht eine Datenrate von $211.72 Mbit/s$. Allerdings trägt der Anteil der Drucksensoren hierbei nur weniger als $800 kbit/s$ zum gesamten Datenvolumen bei. Ein weiteren Einsatz findet sich in [1] wieder, in der u. a. *M-LVDS* zur Kommunikation und Datenübertragung genutzt wird.

Ein bekannter und gängiger Standard für die differentielle und serielle Datenübertragung ist **RS-485**. Auch dieser besitzt die Eigenschaften, sehr Tolerant gegenüber elektromagnetischen

Störungen zu sein und verfügt über die Möglichkeit Daten in sehr hoher Datenrate, bis zu $12Mbit/s$, zu übertragen. Dabei sind Kabellänge von bis zu über einem Kilometer möglich. Des Weiteren erlaubt *RS-485* auch das bidirektionale Übertragen von mehreren Sendern und Empfängern (32 Teilnehmern) über nur einem Adernpaar. Da Mikrocontroller in der Regel keinen *RS-485* Baustein mitliefern ist man auch hier auf ein externes Modul angewiesen. Davon sind diverse Ausführungen von verschiedenen Herstellern zu beschaffen (z.B *MAX485*, *SN75176* oder *ADM483*), jedoch werden diese in der Regel über die *UART* Schnittstelle angeschlossen, was zwar bis auf die Geschwindigkeit alle Vorteile dieser Schnittstelle beibehält, jedoch eines der essentiellen verliert. Des Weiteren ist auch hier zu beachte, dass es sich jeweils um eine zusammenhängende Leitung in einer Linie handeln muss, was in gewissen Situationen zu Problemen führen kann. Zum Einsatz kommt dieses Verfahren in der Arbeit [6] von Banks. Hier wird das externe *RS-485* Modul über einen *RS232-485 Serial Port Converter* angesteuert und damit, so wie schon angesprochen, den Geschwindigkeitsvorteil verliert. Der Ausschlag gebende Grund für die Nutzung der Schnittstelle in dieser Arbeit war es, dass diese im System bereits Teil der Infrastruktur sind und zur Ansteuerung diverser Geräte dienen.

Ein schnelleres System, welches auch auf vielen Mikrocontrollern standardmäßig vorhanden ist, ist das **SPI** (*Serial Peripheral Interface*). Hier sind auch Datenraten im hohen zweistelligen Bereich möglich. Bei Mikrocontrollern setzt sich die Übertragungsrate dabei meist aus folgender Rechnung zusammen: $\frac{f_{CLK}}{4}$. Somit ist ein Viertel der maximalen Taktrate f_{CLK} des Mikrocontrollers, für die SPI Übertragung, möglich. Da übliche Taktraten eines Controllers zwischen $8MHz$ und $100MHz$ liegen, ergibt sich eine Datenrate von $2MHz$ bis $25MHz$. Außerdem handelt es sich hier um einen synchronen und seriellen Datenbus, welcher es einfach ermöglicht weitere Teilnehmer (zum Beispiel weitere Finger bzw. Sensormodule) an den Bus und somit in das Übertragungssystem zu integrieren. Neben den zwei Leitungen für die Übertragung und einer für die Leitung der Taktrate, wird für jeden zusätzlichen Teilnehmer eine weitere Leitung benötigt. Wie man aus dem Name bereits schließen kann, dient diese Schnittstelle oft zum anschließen externer Geräte, wie z.B. AD-Wandlern, diversen Sensoren oder auch weiteren Mikrocontrollern. Zum Einsatz im Bereich taktiler Sensoren kommt diese z.B. in [82], [79] und [81].

Eine Weitere serielle Schnittstelle ist **USB** (*Universal Serial Bus*). Diese ist mittlerweile nicht nur die wohl verbreitetste Verbindung zu externen Geräten, sondern besitzt Dank der symmetrischen Signalübertragung auch ausreichend Bandbreite (USB 2.0 (*Hi-Speed*) $480Mbit/s$). Sie wird von vielen Mikrocontrollern unterstützt und liefert im Standard eine Spannungsversorgung von $5V$. Im Gegensatz zu vielen vorherigen vorgestellten Schnittstellen, muss hier das USB-Protokoll eingehalten werden. Diese Schnittstelle bietet sich besonders gut als Verbindung zwischen dem gesamtem Sensorsystem und der Komponente, z.B. das System eines Roboter, die das System nutzt, an. Besonders durch die hohe Datenrate, die einfache Verkabelung und die Möglichkeit zum *Hot Plugging*. So wird dieses auch z.B. in [79] verwendet um das Sensorsystem mit dem System des Roboterarmes zu verbinden.

Ein Weiteres sehr robustes Bus-System, welches auch größere Entfernungen zurücklegen kann (bis zu $2km$), ist **CAN** (*Controller Area Network*). Dieses erreicht eine Übertragungsrate von bis zu $1Mbit/s$. Eingesetzt wird es in der Arbeit [18], wo mithilfe von *CAN* die einzelnen Sensormodule ausgelesen werden. Dabei besteht ein Modul aus 16 Taxel und wird mit einer Frequenz von $50Hz$ abgefragt. Dennoch spielt diese in der vorliegenden Arbeit keine Rolle.

Möglich sind auch IP-basierte Datenübertragungen. Zu den bekanntestem Verfahren hierbei zählen **Ethernet** und **Wireless Lan**. Durch den großen Protokoll Stapel, der hierbei verwendet wird, steigt die Anzahl der zu übertragenden Daten stark an, welcher aber durch die hohe Bandbreite, von bis zu über $1Gbit/s$, keine Rolle spielt. Auch die hohe Reichweite bzw. Kabellänge ist besonders in höheren Hierarchien der Übertragung sehr vom Vorteil. Des Weiteren kann mit Protokollen sichergestellt werden, dass die Daten korrekt und in der richtigen Reihenfolge ankommen. Probleme die sowohl bei kabelgebundenen als auch bei kabellosen Übertragungen entstehen ist die zeitliche Konstante. Daten können nur sehr schwer zeitlich garantiert werden. Ansätze zur Lösung dieses Problems sind in [68] aufgeführt, benötigen allerdings einen großen Hard- sowie Software Aufwand.

Kabellosen Datenübertragungen, wie zum Beispiel **Bluetooth** oder **Zigbee**, sind ideale Verfahren um die Verdrahtung sehr gering zu halten. Daraus ergibt sich eine leichte Erweiterbarkeit und Flexibilität des Systems. Auch deren mittlerweile sehr hohen Datenraten (mit deutlich über $10Mbit/s$) sprechen für diese Verfahren. Allerdings gib es bei allen drahtlosen Verbindungen einen großen Störfaktor, der gerade in häuslicher Umgebung vermehrt auftreten kann und eignet sich deshalb nur bedingt in sicherheitskritischen Anwendungen. In [84] und [85] werden einige Arbeiten vorgestellt, die dennoch darauf zurückgreifen.

3.6.3 Kommunikationsstruktur

Die Kommunikationsstruktur bezeichnet die Verbindungen zur Kommunikation und Datenübertragung zwischen den einzelnen Sensoren und dem Endsystem. Dafür gibt es wie nachfolgend dargestellt, mehrere Möglichkeiten, von einfachen direkten Verbindungen bis hin zu komplexen hierarchischen Strukturen die mehrere Ebenen durchlaufen. Dabei stehen Anforderung, wie ein geringer Verkabelungsaufwand, Kostengünstig, Zuverlässigkeit, Flexibilität und Modularität oft an erster Stelle.

Eine Möglichkeit ist es, für jeden Sensor eine Leitung bereitzustellen. So kann jeder Sensor über diese von dem Empfänger aufgefordert werden, seine Daten über diese Leitung zu senden (siehe Abbildung 3.5). Ausschließlich die Taktleitung (*CLK*) kann dabei geteilt werden. Das offensichtliche Problem ist hier der Verdrahtungsaufwand, welcher durch jeden weiteren Sensor stark ansteigt. Ein weiterer Nachteil ist die Limitierung der Sensoren, da jeder Sensor einen Eingang am Empfänger benötigt und z.B. Mikrocontroller in vielen Fällen nur 8 bis 16 Ports zum ansteuern besitzen [vgl. 36, S.39]. Hier gäbe es zwar die Möglichkeit durch so

genannte *Demultiplexer* die Anzahl der Ports zu steigern, womit aber das Problem der hohen Anzahl an Leitungen nicht beseitigt werden kann und die Schaltung noch komplexer gestalten würde. So bietet sich dieses Verfahren der direkten Verbindung nur bei sehr geringer Anzahl von Sensoren an.

Die Bus-Struktur, auf der rechten Seite der Abbildung 3.5 dargestellt, wirkt den eben genannten Nachteilen entgegen. Es gibt hier zwei gemeinsame Leitungspaare, wieder eine gemeinsame CLK Leitung, aber auch nur eine gemeinsame Daten-Leitung. Der Sender adressiert die Daten für den Empfänger, damit dieser Bescheid weiß, dass die Daten auf dem Datenbus für ihn bestimmt sind. Die Verbindung zwischen Sender und Empfänger ist, nachdem der Empfänger vom Sender angesprochen wurde, bidirektional. Durch die Nutzung der gemeinsamen *CLK* sowie Datenleitung wird der Aufwand für die Verkabelung deutlich reduziert. Des Weiteren ist das System viel flexibler, so dass das hinzufügen weiterer Sensoren oder Module keinen großen Aufwand benötigt und das System in seiner Struktur nicht beeinflusst. Eine bekannte Bus-Schnittstelle, die mit diesem Verfahren arbeitet, ist I²C vgl. [54].

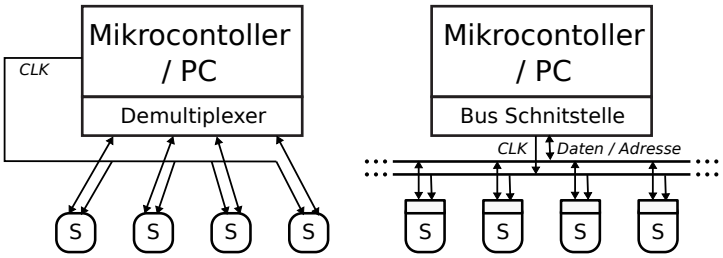


Abbildung 3.5: *Stern- und Bus-Architektur*

Ein weiteres Verfahren, welches die beiden aus Abbildung 3.5 vereint, ist die Struktur die auch bei *SPI* zum Einsatz kommt. Dieses ist in 3.6 dargestellt und arbeitet nach dem *Master-Slave-Prinzip*. Für die Kommunikation stehen drei gemeinsame Leitungen, an denen jeder Teilnehmer angeschlossen ist, zur Verfügung. Zwei separate Leitungen für den bidirektionalen Datenaustausch und eine vom Master zur Synchronisation ausgegeben *CLK* Leitung. Des Weiteren wird für jeden *Slave* eine *Select*-Leitung benötigt, über die er von Master als Kommunikationspartner ausgewählt werden kann. Durch das einfache Selektieren der einzelnen Sensoren, wird sowohl die Hardware als auch Software, z.B. im Vergleich zur *I²C* Schnittstelle, deutlich vereinfacht und lässt so eine schnellere Bandbreite zu. Dennoch ist als Nachteil zu erwähnen, dass auch hier die Anzahl der Leitungen mit jedem weiterem Sensor um eine Leitung steigt.

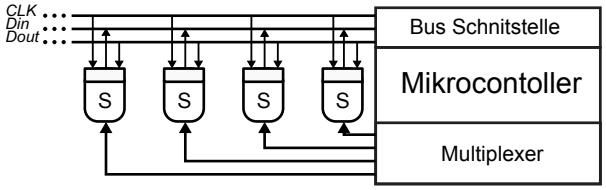


Abbildung 3.6: *SPI-Architektur*

In vielen Fällen, wie auch in dieser Arbeit, reicht es meistens nicht aus, ausschließlich mit einer einfachen Struktur zu arbeiten, um den Anforderungen an das System gerecht zu werden und Konflikte bei der Erfüllung dieser zu vermeiden. So sind die Anforderungen bei der Datenerfassung auf unterster Ebene meist andere, als bei der Übertragung an das Zielsystem. Um dafür eine Lösung zu finden, kann die gesamte Übertragung in einzelne Teilstrecken, durch die Verwendung von Modulen und Submodulen, aufgeteilt werden. Nun kann die gesamte Übertragung über mehrere hierarchische Ebenen strukturiert werden und für jeden Teil der Übertragungsstrecke, mit dem jeweiligen Datenvolumen, das beste Ergebnis für die jeweilige Teilstrecke erzielt werden. Zusätzlich lässt sich so die Verkabelung geringer halten und das gesamte System gewinnt an Modularität, da in der Regal immer nur das Teilsystem bei einer Erweiterung betroffen ist und nicht das gesamte Übertragungssystem. So lassen sich einzelne Komponente leichter austauschen und die Flexibilität und Zuverlässigkeit steigt. Besonders bei einer hohe Anzahl an Sensoren, über eine längere Strecke ($> 10cm$), ist eine direkte Ansteuerung (**Point-to-Point**), wie in Abbildung 3.7 dargestellt, nicht Sinnvoll. So müsste für jeden Sensor die Verkabelung durch das komplette System gelegt werden.

Eine Alternative die den eben beschriebenem Nachteil entgegenwirkt und auch in vielen Arbeiten Anwendung findet (z.B. in [82] und [79]), ist in Abbildung 3.8 dargestellt. Hier werden zusammenhängende oder räumlich nah positionierte Sensoren in Gruppen zusammengefasst und über einen Controller erfasst, oft bezeichnet als *Embedded Local Data Prossessing* [98, vgl.]. Dieser Controller kann so sehr nah an den Sensoren liegen und spart damit lange Verbindungswege und ermöglicht hohe Datenraten bei geringer Fehleranfälligkeit. Die erfassten Daten der Controller können dann z.B. über ein Bus-System, wie *SPI* oder *I²C*, auch über längere Stecken, einfacher, schneller und sicherer übertragen werden. Dieses lässt sich natürlich beliebig skalieren und mit unterschiedlichsten Übertragungstechniken kombinieren. Durch die in der Regel steigenden Datenmengen in den höheren Hierarchien, wird dort eine höhere Bandbreite benötigt, als auf den unteren Ebenen. Auch die Komplexität der Übertragungsprotokolle und die Leitungslängen nehmen in der Regel in höheren Ebenen deutlich zu [26, vgl.].

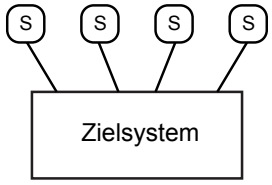


Abbildung 3.7: *Point-to-Point Architektur*

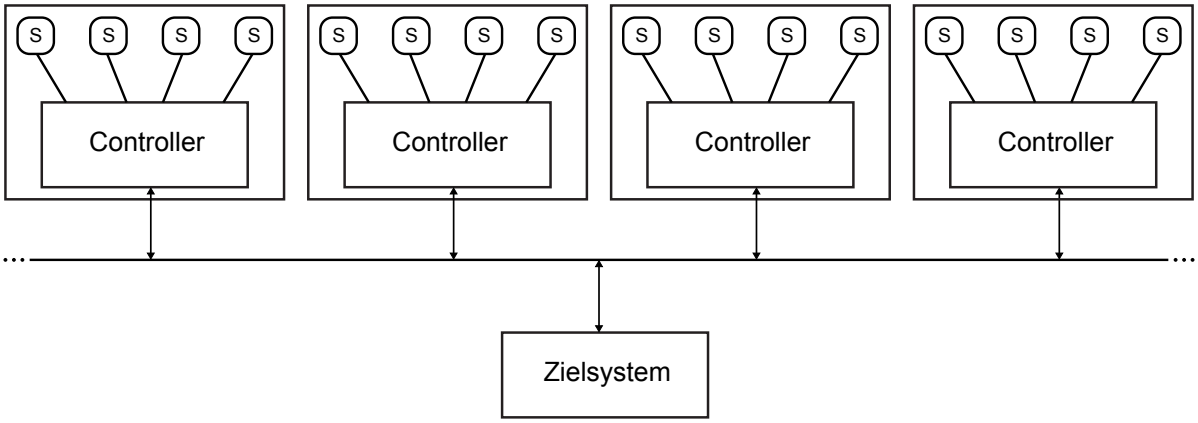


Abbildung 3.8: *Einfache hierarchische und modulare Bus-Architektur.*

3.7 Anforderungen

Anhand der vorherigen Analyse und Betrachtung verschiedenster Arbeiten aus dem selben Forschungsbereich und Themengebiet, werden in diesem Abschnitt die daraus gewonnen Erkenntnisse genutzt, um die Anforderungen für diese Arbeit zu definieren.

Der Kernpunkt dieser Arbeit ist es ein Konzept für ein taktiles Sensorsystem zu erstellen, mit dem es möglich ist schlupfsicheres Greifen zu ermöglichen. Dieser Kernpunkt soll zwar die höchste Priorität bekommen, dennoch wird bei dem System darauf geachtet, grundlegende Greifanforderungen, welche teilweise von [65] definiert wurde, zu erfüllen:

- **Positionskontrolle** um die Öffnung des Greifers auf die richtige Breite einzustellen und die Position des zu greifenden Objektes zu kennen.
- **Kraftkontrolle** zum greifen von Objekten mit verschiedenen Gewichtskräften.
- **Berührungspunkt Erkennung** für das schließen des Greifers bis ein Kontakt zu einem Objekt hergestellt wurde, um je nach Benutzerkonfiguration abrupt das Greifen zu stoppen.

- **Schlupferkennung** für das Halten eines Objektes mit Kraftkontrolle und zusätzliches überwachen des Schlupfs. So kann der Controller automatisch die Kraft erhöhen, bis das rutschen verhindert wurde.
- **Eventerkennung** zur Erkennung verschiedenster Events an dem Greifer, um daraufhin benutzerspezifische Aktionen durchzuführen.

Für die Erfüllung der Ziele, auch unter der Berücksichtigung der eben erwähnten Greifanforderungen, wird ein schnelles, flexibles und robustes taktiles Sensorsystem benötigt. Dazu muss nicht nur ein gescheites Sensorprinzip konzipiert werden, sondern es gilt auch ein Übertragungssystem zu erarbeiten, welches in der Lage ist, eine Vielzahl von Sensordaten zu erfassen und an das Zielsystem, dem Roboter, zu übermitteln. Ein wichtiger Punkt dabei ist, dass dieses System in der Komplexibilität beschränkt ist und die Kosten in einem angemessenem Rahmen liegen.

Durch den Einsatz im menschlichen Umfeld, sind eine enorme Feinfühligkeit und das Tragen von schweren Lasten keine Pflichtkriterien. Des Weiteren, wird versucht die Aufgabe mit möglichst wenigen Sensorsystemen zu bewältigen. Hier haben Ansätze wie [82] gezeigt, dass es allein mit einem Array aus Sensoren eines Typs möglich ist. So soll auch für diese Arbeit, die Umsetzung durch ein solches Sensorarray eines Sensortyps konzipiert und erstellt werden, welches die Anforderungen erfüllt. Ein zusätzlicher Punkt, der das Sensorarray kräftigt, ist die positive Erfahrung die bereits in Arbeiten wie [13], [15] und [16] gesammelt wurde und kann stark in diese Arbeit mit einfließen.

Eine der entscheidendsten Anforderung, welche dadurch einen großem Einfluss auf das komplette System nimmt, ist die Reaktionszeit, bezogen auf die Erfassung der Kräfte an den einzelnen Sensoren bis hin zu der gesamten Datenübertragung. Die Arbeiten [26], [82], [87], [91] und [35] haben gezeigt, dass sich die Abtastrate für die Erkennung von Schlupf im Bereich von $100Hz - 1000Hz$ befindet. Daher wird hier definiert, dass das zu entwickelnde System eine Frequenz von $800Hz$ erreichen muss. Dieses entspricht einer Reaktionszeit von $1,25ms$ und wurde auch schon unter 3.6.2 zur Analyse vorweg angenommen. In dieser Geschwindigkeit müssen nicht nur die Daten von den Sensoren an das Zielsystem, den Roboter, übertragen werden, sonder auch die Krafterfassung durch den Sensor ist darin beinhaltet. Hier spielt auch das Material und Verfahren des Sensors eine wichtige Rolle, denn diese müssen auch in der Lage sein in der hohen Geschwindigkeit, die Kraftwerte in einer guten Präzision aufzunehmen. Des Weiteren müssen die Daten zuverlässig übertragen werden. Fehlerhafte Daten können, besonders in einem menschlichen Umfeld, verheerende Folgen mit sich führen. Ein wichtiger Punkt in diesem Zusammenhang ist noch die Aktualität der Daten. Es wird kein Echtzeitsystem angestrebt, dennoch sollten die Daten so aktuell und zeitnah wie möglich dem Endsystem bereit gestellt werden.

Ein weiterer wichtiger Kernpunkt ist die Auflösung des Sensors. Vergleichbare Arbeiten, wie [82], [79], [81] und [88] zeigen, dass bei Array basierten Systemen die Anzahl der Taxel über

100 liegen sollte. Auch in [26] wird ein Array von $10 * 15$ empfohlen. Des Weiteren spiegelt sich diese Zahl auch bei dem Menschen wieder (vgl. 3.2). Deshalb sollte in dieser Arbeit, das zu entwickelnde Sensorarray die Anzahl von 150 Taxel als Richtwert ansehen. Da dieses System später für den Greifer eines Zwei-Finger-Greifer eingesetzt werden soll, sollten mindestens zwei dieser Sensorarrays mit der vorher definierten Abtastrate betrieben werden können. Dieses bedeutet die Erfassung und Übertragung von ca. 300 Taxel Daten innerhalb von $1,25ms$. Dabei ist die Datengröße eines Taxel mit mindestens $8bit$ zu berechnen.

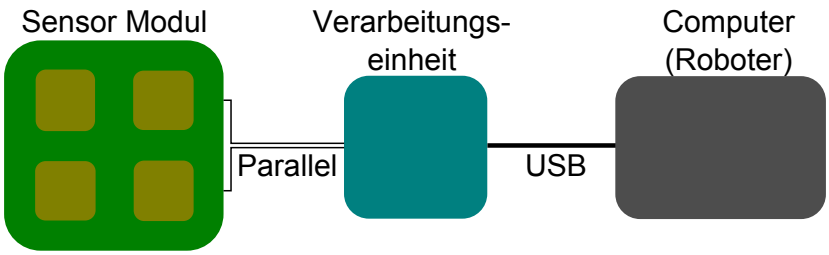
Um die freie Bewegung des Greifers zu gewährleisten, soll die Übertragung von dem Greifersystem zum Robotersystem über ein Verbindungskabel mit maximal $5mm$ Durchmesser erfolgen. Außerdem darf auch das Sensorsystem den Greifer nicht groß beeinträchtigen. Auch der Sensor selbst sollte möglichst klein gehalten werden und das Material dem menschlichen Umfeld angepasst werden.

Weitere Anforderungen an das System sind die Skalierbarkeit und Flexibilität. Das System soll einfach zu erweitern sein und die Sensoren müssen einfach dupliziert werden können, ohne in das komplette System einzugreifen. Ein weiterer Punkt ist es, die Herstellung einfach und besonders kostengünstig zu halten. Man muss in der Lage sein, alle Bauteile gut beschaffen zu können und mit einfachen Mitteln diese zu dem gesamten System zusammen zu bauen. So soll ausschließlich auf Standard Interfaces zurückgegriffen werden. Besonders durch die letzteren genannten Anforderungen, setzt sich das System dieser Arbeit besonders von vergleichbaren Arbeiten ab und ermöglicht einen massentauglichen Einsatz auch in privaten Haushalten. Des Weiteren fördert es die Entwicklung, da jeder in der Lage ist, sich dieses System für seine Zwecke der Forschung zu erstellen und damit die Entwicklung weiter vorantreiben kann.

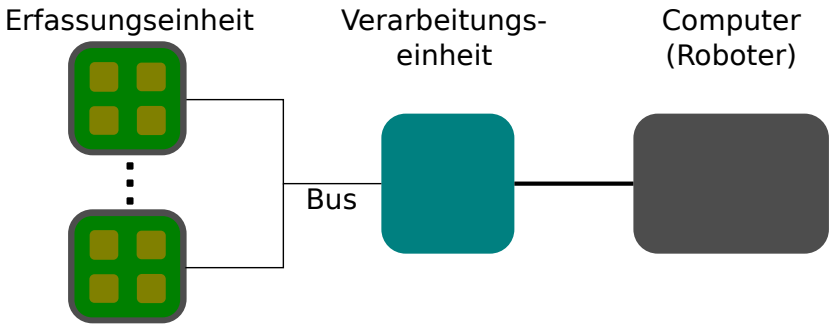
Alle diese Anforderungen gilt es nun im nachfolgendem Konzept zu berücksichtigen und am Ende der Arbeit in dem Sensorsystem umgesetzt sein.

4 Grundkonzept des Sensorsystems

In diesem Kapitel wird das Konzept der Sensorik, der Erfassung der Sensordaten und deren Übertragung an das Zielsystem, den Roboter, beschrieben. Ein grober Systementwurf ist bereits in der Arbeit [15] entstanden, welcher in Abbildung 4.1(a) dargestellt ist. An diesem Entwurf wird auch in dieser Arbeit weiter festgehalten, allerdings mit einigen Änderungen. Diese werden, nach dem Konzept des einzelnen Sensors beschrieben und geben einen Einblick in die Konzeption und die Evaluierung des gesamten Sensorsystems. Da es sich um ein sehr komplexes System handelt, welches zugleich sehr modular aufgebaut ist, wird die weiterführende Konzeption und Evaluierung der einzelnen Teilkomponenten in separate Kapitel aufgeteilt. Bei diesen Kapiteln handelt es sich um die Evaluierung der Erfassungseinheit, der Verarbeitungseinheit und des Sensorsystems.



(a) Jeder Sensor des Arrays ist mit der Verarbeitungseinheit verbunden, welche die Daten an den Computer via USB gebündelt überträgt.



(b) Jedes der N Erfassungseinheiten erfassen die Daten aus seinem Sensorarray und übertragt dieses über einen Bus an die Verarbeitungseinheit, welche diese Daten wiederum an den Computer via eines Bus weiterreicht.

Abbildung 4.1: *Übersicht des Konzepts des Sensorsystems aus [15] (a) und dieser Arbeit (b).*

4.1 Konzept des Sensors

Der Sensor bezeichnet in diesem Zusammenhang die unterste Ebene der Druckkrafterfassung. Diese sind in der Erfassungseinheit untergebracht. Dabei handelt es sich auch wie in den vorherigen Arbeiten [13] und [15], um ein taktiles Sensorarray, basierend auf Elektroden und einem elastischen und leitfähigen Polymers. Dieses Array wird dabei wie in der Abbildung 4.1(b) dargestellt, als Matrix-Anordnung umgesetzt. Dieses Sensorarray bietet durch seine Vielseitigkeit nicht nur die Möglichkeit den Schlupf zu erkennen, was z.B. die Arbeit [79] deutlich bewiesen hat, sondern kann auch diverse andere Aufgaben lösen. Des Weiteren handelt es sich dabei um ein einfaches und kostengünstiges Prinzip der Druckkrafterfassung, welches zugleich eine hohe Datenerfassungrate erlaubt. Eine ausführliche Beschreibung, Evaluation und Auswertung ist in den Kapiteln 5.1 und 5.2 zu finden.

4.2 Konzept des Sensorsystems

Der neue und abgeänderte Entwurf aus Abbildung 4.1(b) basiert auf den Erkenntnissen, die in der Analyse entstanden sind und hält sich an den daraus entstanden Anforderungen. Es handelt sich um 1 bis N Sensormodule, die für die Erfassung der Taxel Daten zuständig sind. Konkret wird in dieser Arbeit von zwei Modulen ausgegangen, da es sich bei dem Greifer des Roboters der als Zielsystem gedacht ist, um einen Zwei-Finger-Greifer handelt. So werden alle Taxel eines Moduls ausgelesen und als ein gesamtes Datenpaket, über ein Bussystem, an die Verarbeitungseinheit übertragen. Die Verarbeitungseinheit soll dabei möglichst in räumlicher nähe zu den Sensormodulen liegen, um Datenwege und die damit verbundenen Leitungen möglichst kurz zu halten. Dieses ist zum einen wichtig, um den Roboter durch eine Vielzahl an Leitungen nicht in seiner Arbeit einzuschränken und den Bewegungsablauf zu behindern. Zum Anderen stehen auf kürzeren Distanzen (siehe 3.6) weitaus mehr schnelle und einfache Datenübertragungssysteme zur Verfügung.

Die Verarbeitungseinheit dient in diesem System als Datenknoten, welcher die Daten der Module über eine Bus Verbindung an das Endsystem, den Roboter bzw. dessen Computer, weiterleitet.

Auf den Computer des Roboters wird in dieser Arbeit nicht eingegangen, da dieser nicht Teil des Sensorsystems ist und durch seine Verwundung in einer Vielzahl von anderen Arbeiten und Projekten nicht abgeändert werden kann. Somit wird immer von dem Ist-Zustand, welcher in den Grundlagen in 2.4 beschrieben wurde, ausgegangen.

Der Entwurf, abgebildet in 4.1(b), wird infolge der nächsten drei Kapiteln, Evaluierung der Erfassungseinheit, Evaluierung der Verarbeitungseinheit und Evaluierung des Sensorsystem, weiter erarbeitet, detaillierter beschrieben und evaluiert. Die genaue Vorgehensweise ist dabei zuvor im nächsten Abschnitt erläutert.

4.3 Konzeption und Evakuierung der einzelnen Komponenten

Bei dem dargestellten Konzept handelt es sich um ein System, welches aus den beschriebenen und dargestellten Teilsystemen besteht. Diese Teilsysteme gliedern sich in den einzelnen Systemen noch weiter in kleinere Teilsysteme auf. Dieses Zusammenspiel der diversen einzelnen Teilkomponenten gilt es sowohl einzeln, als auch im Zusammenhang zu betrachten. So werden zuerst die einzelnen Teilsysteme erarbeitet, analysiert und ausgewertet, um anschließend diese Ergebnisse für Teilkomponenten höhere Abstraktionsebenen mit einfließen zu lassen. Durch dieses Verfahren kann am Ende ein gesamt Konzept, welches den Anforderungen schon in vielen Teilen entspricht, erstellt werden. Damit wird schon früh die Aussicht auf ein erfolgreiches Endsystem gesteigert und bietet zugleich eine gute und nachvollziehbare Struktur der einzelnen Teilsysteme und deren Komponenten.

Die nachfolgenden Kapitel befassen sich ausführlich mit dem zuvor vorgestellten Konzept. Das Kapitel Evaluierung der Erfassungseinheit beschreibt die Aufnahme der einzelnen Kräfte an den Taxel, über deren Verarbeitung und anschließend die Übertragung der Daten an die Verarbeitungseinheit des Systems. Es beinhaltet damit die gesamte Datenerfassung. Darauf folgt das Kapitel Evaluierung der Verarbeitungseinheit, in dem die Weiterleitung der zuvor erfassen Daten beschrieben, erarbeitet und anschließend ausgewertet wird. Im letzten dieser drei Kapitel wird in Kapitel 7 das Zusammenspiel aller Komponenten veranschaulicht, zusammengefasst und erläutert.

Zur Beschreibung des Systems in den nachfolgenden Kapiteln, wird der *Bottom-up* Ansatz gewählt. Es wird auf der untersten Ebene, in diesem Fall bei der Erfassung der Druckkräfte an den Taxeln, begonnen und endet mit der Übertragung der Daten an den Computer.

4.3.1 Evaluierungsverfahren

Für die anschließende Konzeption und Evaluierung der einzelnen Teilsysteme, wird stets mit der selben Testumgebung, unter möglichst gleichen Bedingungen, gearbeitet. Bei Messungen die von einer nicht zum System gehörenden Komponente getätigt werden, wird darauf geachtet, dass diese Komponente den Leistungen des gesamten Systems angemessen ist. So werden alle Arbeiten und Tests die mit und auf einem PC laufen, auf dem selbem Betriebssystem wie dass des Roboters und mit ähnlichen Eigenschaften getätigt.

Da es sich bei der Evaluierung um verschiedene Komponenten von diversen Herstellern handelt, wird es besonders auf der *LOW-Level* Ebene nicht Möglich sein, eine einheitliche Software zu schreiben. Denn jeder Mikrocontroller und jedes Bauteil muss anders Konfiguriert werden und bietet durch beigefügte Bibliotheken verschiedenste Funktionen und Umsetzungen dieser an. Um dennoch die Software einheitlich zu gestalten und möglichst wenig redundanten Co-

de zu generieren, werden für alle Teilkomponenten einzelne sogenannte *Wrapper* Funktionen geschrieben, die dafür sorgen, dass sich alle diese Funktionen von außen her gleich verhalten. Diese Funktionen werden dann mit bedingten Definitionen(`#ifdef A`) versehen, damit der Präprozessor dieses je nach gewünschter Plattform verwendet oder aussortiert. So muss zwar jede Komponente Konfiguriert werden und dafür gesorgt werden, dass die *Wrapper-Funktionen* sich wie gewünscht verhalten, aber der ganze Code darüber muss nicht angefasst werden.

Bei der Kompilierung der Software, wird versucht auf freie Compiler zurückzugreifen. So wird in dieser Arbeit der bekannte *GCC* (GNU Compiler Collection) vorgezogen. Da es sich selbst bei dem *GCC* um verschiedene Compiler handelt, wird auch hier, wie schon im vorherigen Abschnitt, mit bedingten Definitionen gearbeitet. Das System besitzt ein *Makefile* in das alle Teilkomponenten Konfigurationen vorgenommen werde, unter andern auch die Wahl des richtigen Compilers. Durch die Bedingung in diesem *Makefile* ist es nur nötig ihm das gewünschte zu kompilierende System anzugeben und es sucht sich selbst den richtigen Compiler und die dazugehörige Konfiguration, wie zum Beispiel den Systemtakt. Dieses macht das bauen der Software nicht nur deutlich einfacher, sonder stellt auch sicher, dass immer mit derselben Konfiguration gebaut wird und es nicht zu ungewollten Veränderung dieser kommt.

5 Evaluierung der Erfassungseinheit

In diesem Kapitel wird die zuvor im Konzept angesprochene Erfassungseinheit erarbeitet. Dabei werden in einigen Teilen mehrere Ansätze und verschiedene Methoden entwickelt, umgesetzt, evaluiert und die Ergebnisse zusammengefasst. Wie schon zuvor beschrieben, dient diese Einheit zur Erfassung der Druckkraft der einzelnen Taxel und deren Übertragung an die nächst höhere Instanz in dem System (vgl. 4).

Beginnend wird das Prinzip der Kraft- und Druckaufnahme auf unterster Ebene beschrieben. Hier steht hauptsächlich der einzelne Sensor bzw. Taxel im Fokus. Dabei werden neben verschiedenster Taxelformen auch mehrere Materialien getestet, um das für diese Arbeit bestmögliche Ergebnis zu erzielen.

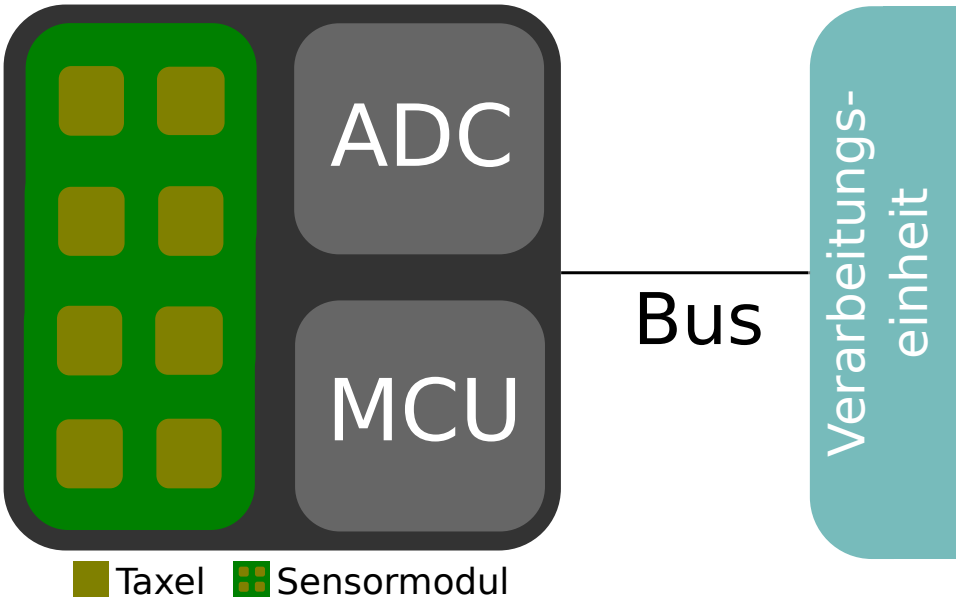


Abbildung 5.1: *Konzept der Erfassungseinheit.*

Frei nach dem *Bottom-Up* Prinzip, wird anschließend die nächst höhere Instanz, das Sensormodul, beschrieben. Hierbei werden die zuvor entwickelten Sensoren / Taxel mit Hilfe eines Mikrocontrollers ausgelesen und die erfassten Daten zur Weiterverarbeitung bereitgestellt. Anschließend werden zwei mögliche Übertragungstechniken, die zuvor im Kapitel Analyse, als mögliche Methoden zur Übertragung infrage kamen, evaluiert. Dabei handelt es sich um die

Schnittstellen *USB* und *SPI*, die die erfassten Sensordaten durch das Sensormodul, an die nächst höhere Ebene übertragen sollen.

Abschließend werden alle gewonnen Erkenntnisse und Ergebnisse zusammengetragen und bewertet. Dieses stellt dann die Grundlage für die Entwicklung des Systems in Kapitel 8 dar.

5.1 Sensor

Dieser Abschnitt der Erfassungseinheit befasst sich mit dem einzelnen Sensor oder hier auch als Taxel bezeichnet. Dabei wird das Prinzip dieses Sensors erläutert, welcher anschließend in einem Testsystem ausgewertet wird. Abschließend werden die gewonnen Ergebnisse zusammengefasst.

5.1.1 Sensor Prinzip

Der über die Sensoren gemessene Druck wird ähnlich wie in [13] und [15], mithilfe von A/D Wandlern, gemessen. Diese können durch eine unbelastete Spannungsteiler Schaltung, die sich durch Druck ändernde Spannung, messen. Die Abbildung 5.2 zeigt dieses Verfahren. Es besteht zum einen aus dem Vorwiderstand R_1 und dem Sensor Widerstand R_y. Der Sensor Widerstand besteht aus einem Material, welches seine Leitfähigkeit durch Ausübung von Druck, deutlich verstärkt. Auf dieses Material wird in dem Abschnitt 5.1.1 weiter eingegangen. Der angesprochene Vorwiderstand bildet den Eingangswiderstand, mit der Größe von $20k\Omega$. Die Verwendung eines $20k\Omega$ Widerstandswertes, wird aus Erfahrungen aus der Arbeit [13] und [15] gewählt und muss abhängig von den verwendeten Elektroden und dem Polymer gegebenenfalls, anhand von Messergebnissen, angepasst werden. Mithilfe der zwei Widerständen kann die Gesamtspannung U_{ges} von der Ausgangsspannung V_{cc}, abhängig von der Größe des Sensor Widerstandes, in zwei Teilspannungen aufgeteilt werden. Ist der Widerstand an dem Sensor sehr hoch $> 1M\Omega$, fällt fast die gesamte Spannung an diesem ab. Dadurch wird so gut wie keine Spannung mehr von dem A/D Wandler gemessen. Wird nun Kraft auf den Sensor ausgeübt, verkleinert sich der Widerstand des Sensors und die Spannung sinkt, was zur Folge hat, dass die Spannung am Vorwiderstand R_1 steigt. Diese Spannungsänderung wird mit dem A/D Wandler gemessen und somit lässt sich den Spannungswerten eine Kraft zuordnen. Durch die Digitalisierung der Messwerte mit Hilfe der A/D-Wandler lassen sich die Werte zusätzlich leichter weiterverarbeiten und übertragen.

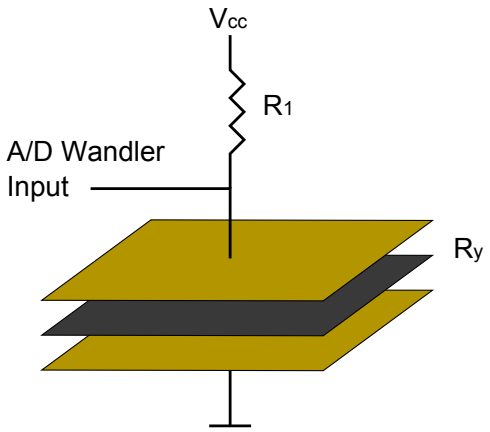


Abbildung 5.2: *Spannungsteiler-Messung eines Taxels mithilfe von einem A/D Wandler.*

Leitfähiges Polymer

Das Prinzip der leitfähigen Polymer wurde im den Grundlagen in Kapitel 2.1.2, ausführlich beschrieben. Diese Polymers eignen sich bestens für diese Art von Sensorsystem. Sie sind flexibel und lassen sich einfach, selbst in kleine Greifer, integrieren. Die Oberfläche ist in den meisten Fällen sehr robust und unbeeinflusst von unerwünschten Fremdeinwirkungen. Des weiteren lassen sich diese Polymers sehr günstig beschaffen.

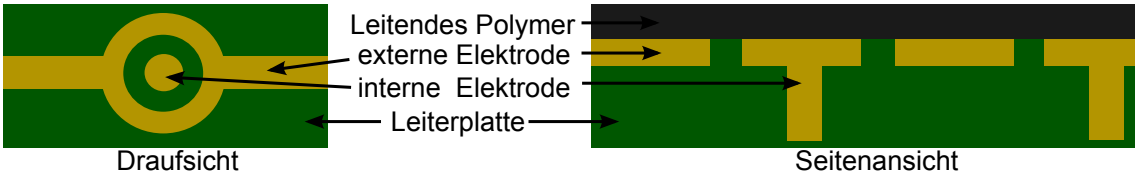
In diesem System wird zum einen das piezoelektrische Polymer der Firma *Interlink Electronics*® eingesetzt. Zum anderen wird auch das piezoresistive Polymer der Firma *Zoflex*® für nähere Tests verwendet.

Sensorarray aus Elektroden

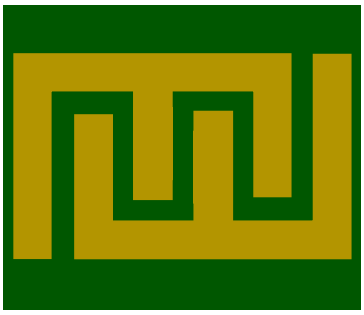
Um das zuvor beschriebene Prinzip umzusetzen, wird ähnlich wie in [19], mit Hilfe einer Leiterplatte, welche mit aufgedruckten Elektroden versehen und mit einem leitendem Polymer bedeckt ist, ein Hochauflösendes taktiles Sensorarray realisiert.

Die Abbildungen in 5.3 zeigen einige Varianten des eben angesprochenen Prinzips des Sensors. Dargestellt ist die Leiterplatte, mit den bedruckten Elektroden (interne und externe Elektrode). Durch das leitende Polymer, welches im unbelasteten Zustand einen hohen Widerstand von über $30M\Omega$ besitzt, liegt keine messbare Spannung zwischen den beiden Elektroden an, da sich diese nur über das Polymer physikalisch berühren. Durch Ausübung von Druck auf das Polymer, singt der Widerstand, abhängig vom Druck bis auf unter 0.1Ω. Durch den sich unter Druck ändernden Widerstand, kann mittels, wie auch schon in der Arbeit [13], mit einem einfachen Spannungsteiler, die Spannung gemessen werden. Die Abbildungen in 5.3 zeigen Beispiele von verschiedenen Anordnungen der Elektroden. Dabei handelt es sich um zwei

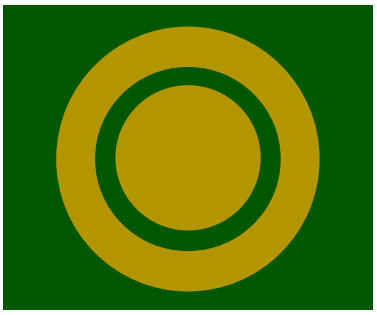
einfache Formen, wie 5.3(c) und 5.3(b) und eine durchaus komplexere in 5.3(d). Diese verschiedenen Anordnungen und Formen gilt es umzusetzen, um mithilfe von gescheiten Tests zu ermitteln, welches sich für die gesetzten Ziele und sich somit für das Ziel der Masterarbeit 1.2 eignet.



(a) Kreis in zwei Perspektiven



(b) Verzahnt



(c) Kreis

(d) Finger

Abbildung 5.3: *Taktile Sensoren auf einem PCB realisiert mit verschiedenen Anordnungen der Elektroden.*

Dieses taktile Sensorprinzip ist nicht nur einfach und kostengünstig herzustellen, sondern ermöglicht auch eine hohe Dichte an Sensoren und individuelle Formen des Sensorsystems. Somit eignet es sich gut für die in Kapitel 1.2 angesetzten Ziele.

5.1.2 System- und Testaufbau

Um die Machbarkeit und Funktionsweise des eben beschriebenen Sensorprinzip nachzuweisen, wird in diesem Abschnitt ein Testsystem aufgebaut. Hierbei wird das PCB[10], welches bereits in [15] entwickelt wurde, verwendet. Diese entworfene Platine ist in Abbildung 5.4 dargestellt. Sie zeigt die drei Formen, wobei der *Kreis* in zwei unterschiedlichen Größen vorhanden ist (3,5mm und 2,5mm Durchmesser). Bis auf die Form *Verzahnt* $(6x8)$, sind alle Elektroden in einer $8x8$ Matrix gedruckt. Diese Anzahl ist für die erforderlichen Tests und Analyse mehr als ausreichend und erleichtert die Verkabelung im Testaufbau.

[10] **PCB**: **P**rinted **C**ircuit **B**oard - Leiterplatte mit gedruckter Schaltung

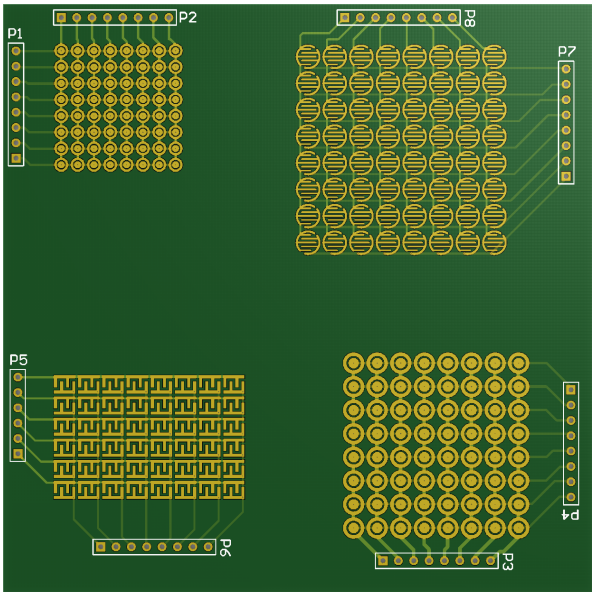


Abbildung 5.4: *Test Platine zur Analyse von verschiedenen Elektroden in einer Matrix-Anordnung.*

Da es sich hierbei ausschließlich um den Test jedes der vier Elektrodentypen handelt, wird pro Typ auch nur eine Elektrode des Arrays benötigt. Dabei werden bei jedem Test der Typen, diese jeweils mit beiden Polymers durchgeführt. So dass am Ende acht Messwerte vorhanden sind.

Um die einzelnen Sensordaten auszuwerten, wird eine Reihe der Sensor-Matrix, über die Kontaktpunkte $P(X)$ (siehe Abbildung 5.4) mit einer $5V$ Spannung versorgt. Wichtig ist, dass zur Zeit immer nur eine Spalte mit Spannung versorgt werden darf, da es sonst zu ungewolltem Verhalten bei der Messung der analogen Werte kommen kann. Die einzelnen Sensoren der aktiven Spalte können nun durch einen A/D Wandler und der zuvor beschriebenen Spannungsteiler Schaltung, ausgelesen werden.

Zur Aufnahme von Sensorkennlinien, wird mit Hilfe einer Waage die Kraft, die auf den Sensor einwirkt, bestimmt. Dazu wird die Sensorplatine mit dem leitendem Polymer auf diese gelegt und Druck ausgeübt. Der Massewert m (in Gramm), den die Waage anzeigt, wird in die Formel der Kraft $F = m * g$ eingesetzt und mit Hilfe der Fallbeschleunigung g der Erde, kann die Kraft F in Newton berechnet werden. Durch dieses Verfahren kann nun jedem beliebigen analogem Wert eine Kraft zugeordnet werden.

Der verwendete A/D Wandler besitzt ein 10-Bit ADC Register und liefert somit Messwerte im Bereich von 0 bis 1023. Liegt am Eingangskanal $0V$ an, so gibt der ADC den Wert 0 an. Hat die Spannung am Eingangskanal die Referenzspannung von $5V$ erreicht, so liefert der ADC einen Wert von 1023. Diese analogen Werte, werden über ein auf dem *Arduino MEGA2560* imple-

mentierten Programm, ständig per serieller Schnittstelle an den Computer übertragen. Dieses ermöglicht eine Auswertung und grafische Darstellung der Daten. Hierzu wird das Programm *MATLAB* benutzt, welches die seriellen Daten empfangen und direkt grafisch darstellen kann. Da zu diesem Zeitpunkt die Geschwindigkeit der Sensorerfassung noch keine Rolle spielt, wird kein Wert auf eine schnelle Übertragung gelegt, welche durch die Verwendung der seriellen Schnittstelle den Durchsatz der Messungen bremst [vgl. 13].

5.1.3 Auswertung Sensor

Dieses Kapitel beschreibt die Leistungsfähigkeit des entwickelten Sensorprinzips. Um die einzelnen Elektroden miteinander zu vergleichen, werden Messergebnisse in den Kategorien Präzision, Kraftaufnahme und Positionserfassung grafisch dargestellt.

Kraftkennlinie

Die Kraftkennlinie soll Aufschluss darauf geben, wie genau die Kraft des Sensors erfasst werden kann. Dabei sind nicht nur die Kräfte, an dem der Sensor anschlägt und wann dieser sein maximales Limit erreicht hat, sondern auch der Verlauf zwischen diesen Punkten, wichtig. Hierzu wurden Messwerte jedes Elektrodentyps, mit jeweils beiden Polymers, ermittelt und in dem Diagramm 5.5 dargestellt.

Bei dem Polymer der Firma *Zoflex*® in ■ dargestellt, zeigt sich das die Kraftkurve sehr schnell steigt. So war das Verhalten bei allen vier Elektroden auch nahezu identisch. Da es mit diesem Polymer nicht möglich ist die Kraft in mehrere Ebenen einzuteilen, kommt dieses für diese Arbeit nicht in Frage und wird nachfolgend nicht mehr betrachtet.

Die Ergebnisse des Polymers von *Interlink Electronics*® (■ ■ ■ ■) zeigen deutlich, dass es sich wie erwartet nicht um eine lineare Gerade, sondern um eine logarithmische Kennlinie handelt und spiegelt die Ergebnisse aus [13] und [16] wieder. Das Verhalten der verschiedenen vier Sensoren mit den unterschiedlichen Elektroden ist dabei sehr ähnlich. Kleine Unterschiede sind beim Ansprechverhalten zu erkennen, so schlagen die Finger-Elektrode und die Verzahnte-Elektrode schon etwas früher an. Eine weitere Differenz ist bei dem maximalen Werten vorhanden. Hier haben die Verzahnte-Elektrode und Kreis-Elektrode einen etwas höheren Wert gegenüber den anderen beiden Typen.

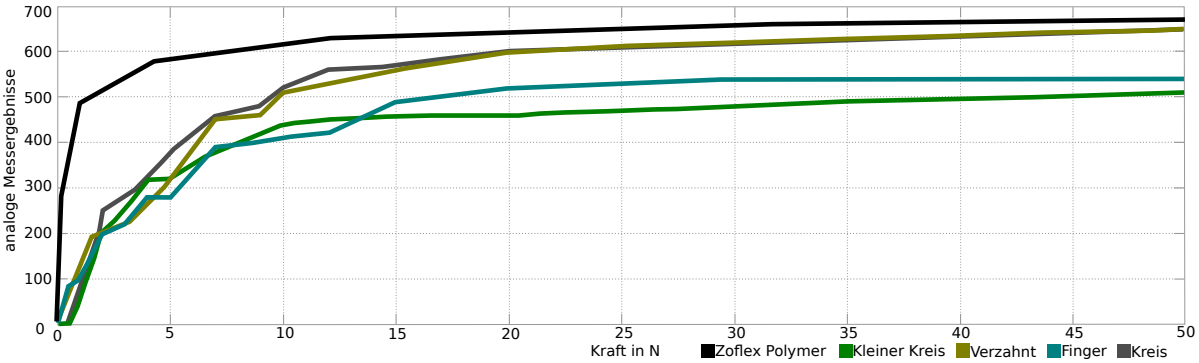


Abbildung 5.5: *Kraftkennlinien der einzelnen Sensoren mit den verschiedenen Elektroden (Kreis, kl. Kreis, Verzahnt, Finger) und Polymer.*

Präzision

Die Präzision beschreibt die Qualität eines Messverfahrens. Dazu wird eine Messung durch oftmaliges Wiederholen, unter gleichen Umständen, durchgeführt [vgl. 10]. Dabei lässt sich feststellen, wie weit die Werte durchschnittlich voneinander abweichen. In diesem Fall wurde an jedem Sensortyp mehrfach mit derselben Kraft und an derselben Position Druck ausgeübt. Das Messverfahren wurde jeweils mit vier verschiedenen Kräfte, mit jeweils 10 Messungen, durchgeführt. In der Abbildung 5.6 sind die Messergebnisse des Sensortypen mit der kreisförmigen Elektrode dargestellt.

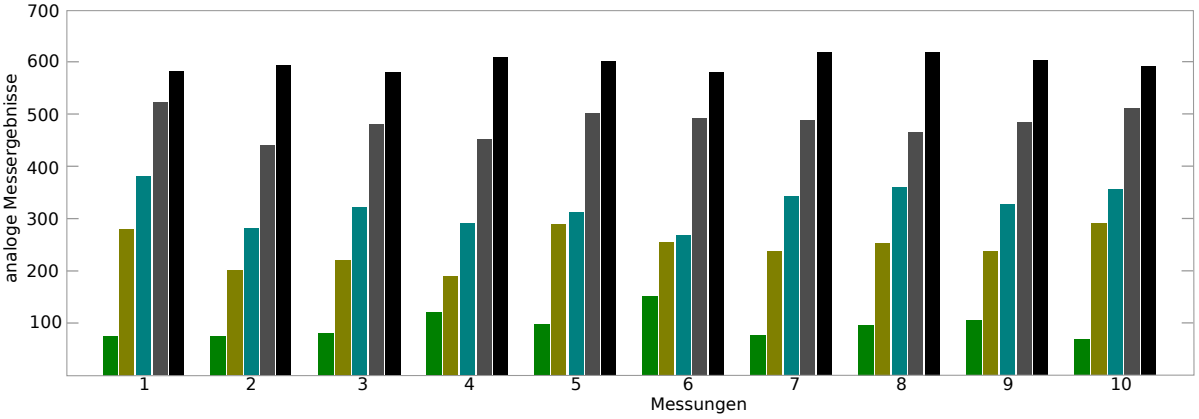


Abbildung 5.6: *Messungen für die Berechnung der Präzision des Sensors mit der kreisförmigen Elektrode.*

Zur Berechnung der Präzision, welche als Standardabweichung σ angegeben wird, wurde die Formel 5.1 aus [10] verwendet. Die Ergebnisse der Berechnung aller Typen ist in der Tabelle 5.1 abgebildet.

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{5.1}$$

x_i = Messwert; $\bar{x}$ = Mittelwert; n = Anzahl der Messwerte

Das Ergebnis spiegelt die erwartete Ungenauigkeit des Polymers wieder. Aber bezogen auf die Tatsache, dass es sich hierbei um einen taktilen Sensor eines Assistenzroboters handelt, sind die Ergebnisse der Messung ausreichend. Zwar ist die Standardabweichung Teilweise fast bei 30%, welches aber immer noch eine eindeutige Klassifizierung mehrerer Kraftstufen zulässt und so für die Erkennung des Schlupfs ausreicht. Des Weiteren liegt die Abweichung in einem Bereich, in dem beim Greifen, einer großen Anzahl an Objekten im alltäglichen Leben keine Beschädigungen dieser auftreten würde.

Kraft in N	x_{min}	x_{max}	$\bar{x}$	σ
1	55	150	90.8	27.3540
3	190	290	244.8	33.9476
5	250	380	319.4	40.1021
10	450	520	482.3	25.1285
50	580	620	597.0	16.7541

Tabelle 5.1: *Ergebnisse der Berechnung der Präzision für den Sensor mit der kreisförmigen Elektrode.*

Positionserfassung

In diesen Abschnitt soll geprüft werden wie das Sensorarray die Positionsänderung eines Objektes aufnimmt. Dazu wurde ein kleiner rechteckiger Gegenstand, innerhalb von ca. zwei Sekunden über das Sensor-Array gezogen. Ergebnis dieses Tests soll die Genauigkeit der Positionserfassung von Objekten prüfen und die Wahrnehmung von Bewegungen testen.

Abbildung 5.7 zeigt diesen Vorgang, an dem zu erkennen ist, dass sich das Objekt von oben nach unten bewegt hat. In der Darstellung ist jeweils zu sehen, wo sich das Objekt zurzeit befindet. Dieses Ergebnis zeigt, dass der Sensor ziemlich genau weiß, wo sich ein Objekt an seinem Sensorsystem befindet und auch die Bewegung dieses sehr gut wahrnehmen kann.

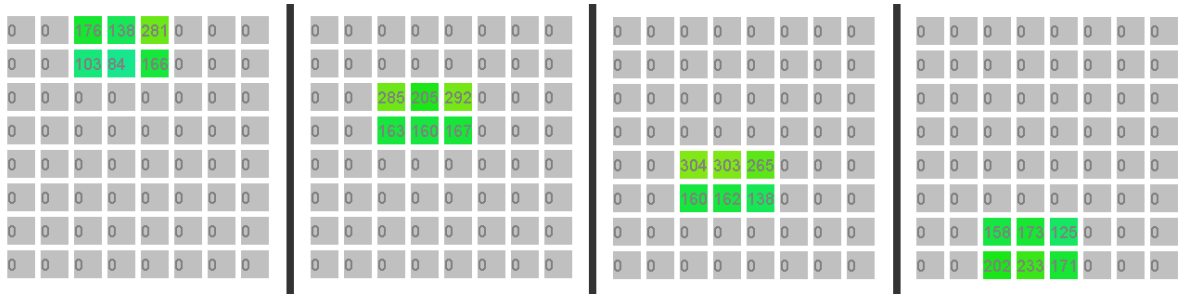


Abbildung 5.7: *Positionserfassung eines sich bewegenden rechteckigen Objektes.*

5.2 Sensormodul

Das Sensormodul besteht zum einen aus den Sensoren bzw. Taxel, dessen Prinzip zuvor ausführlich beschrieben wurde und zum anderen aus einer Prozessoreinheit, einem ADC, der die Daten der Sensoren erfasst und an die nächst höhere Instanz weiterleitet, in dem Fall an einen Mikrocontroller. Der Fokus in diesem Abschnitt liegt besonders auf der Erfassung der analogen Sensordaten. Die Übertragung dieser Daten wird in den beiden nachfolgenden Sektionen 5.4.1 und 5.4.2 der Übertragung ausführlich beschrieben.

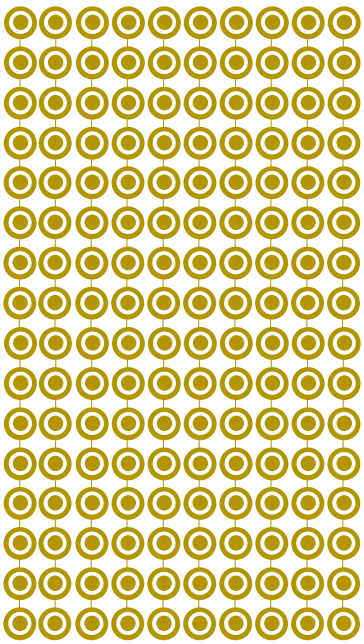


Abbildung 5.8: *Sensorarray eines Moduls mit* 160 *Taxel.*

Wie in den Anforderungen definiert, steht die gesamte Erfassung und Übertragung der Sensordaten stark unter dem zeitlichen Aspekt. Für die Übertragung aller Daten stehen in etwa $1 - 2ms$ zur Verfügung. Damit bleibt für ein einzelnes Sensormodul deutlich weniger Zeit als $1ms$. Aufgrund der zusätzlichen und notwendigen hohen Anzahl an Taxel, hier 160 pro Modul (vgl. 3.7), ist eine gesamte Abtastung nicht nur mit einem einzigen Analog-Digital-Wandler (ADC) durchzuführen. Dieses wäre zum einen problematisch, bezogen auf die Verdrahtung, zum anderen könnte es auch zu einem zeitkritischen Problem werden. Aus diesem Grund kommen eine Vielzahl an Wandlern zum Einsatz, auf welche später noch weiter eingegangen wird.

Bei dem Aufbau der Schaltung zwischen den einzelnen Taxel und der ADCs werden in dieser Arbeit verschiedene Verfahren in Betracht gezogen. Zum einen besteht die Möglichkeit, jedem Taxel seinen eigenen ADC zur Verfügung zu stellen. Dieses ist sowohl software- und hardwaretechnisch wohl die einfachste, wenn auch nicht die effizienteste Lösung und wird in dem Abschnitt „*Ein ADC Baustein pro Sensormodul*“ beschrieben und analysiert.

Eine weitere Methode ist es, ähnlich wie in [13] oder auch [9], dass sich mehrere Taxel eine Anzahl an ADCs teilen. In vorherigen Arbeiten und auch in vielen verwandten (vgl. 3.1), wird die sequentielle Abtastung vorgezogen. Die Ansteuerung eines Taxel wird dabei über das sogenannte *Demultiplexing* getätigt. Dieses hat laut [81] allerdings einen großen Geschwindigkeitsnachteil. Denn das Umschalten eines ADC Eingangs bis zu dem Zeitpunkt, an dem die Spannung stabil anliegt, kann bis zu $300\mu s$ dauern. Da hier die Anforderungen an die Geschwindigkeit sehr hoch ist, wird neben dem Verfahren auch dieses Verhalten in dem Abschnitt „*Lokales Teilen eines ADC Eingangs*“ ausführlich beschrieben und untersucht.

Vor der ausführlichen Beschreibung der zuvor vorgestellten Methoden, widmet sich dieses Kapitel noch der einzelnen Datenerfassung durch die Analog Digital Wandler.

5.2.1 Analog Digital Wandler

Um die einzelnen Taxel auszuwerten, benötigt es wie schon im Abschnitt 5.1 beschrieben einen Analog Digital Wandler (ADC). Dieser wandelt die sich ändernde Spannung der Taxel in ein digitales Signal um. Dieses Signal lässt sich dann leicht weiter Übertragen, Auswerten und Verarbeiten. Hinter einem ADC steht in der Regel ein Mikrocontroller, welcher sich um die Steuerung und Erfassung der analogen Daten kümmert. So ist es auch kein Zufall, dass eine Vielzahl an Mikrocontrollern bereits ein oder mehrere ADCs integriert haben. Da es sich in diesem Fall (siehe Anforderungen) um mindestens 160 zu erfassende analoge Werte handelt, wird die Verwendung von mitgelieferten ADCs an einem Mikrocontroller außer acht gelassen. Zusätzlich spielt dabei auch der Geschwindigkeitsfaktor eine entscheidende Rolle, da die integrierten ADCs in der Regel nicht an die Geschwindigkeit externer herankommen. Aus diesen Gründen wird auf externe ADCs zurückgegriffen.

Externe ADCs stehen in allen bekannten Bauformen und Größen zur Verfügung. Technisch liefern sie eine digitale Auflösung des analogen Signals von bis zu $24bit$, bei einer Abtastrate von über 1 GSPS[11] und bieten pro Baustein über 200 Pins. Außerdem werden viele bekannte, einfache, serielle und parallel Datenschnittstellen, wie z.B. *SPI*, und *I2C*, bereitgestellt. Bei der Zusammenfassung der vorhandenen technischen ADC Werten handelt es sich immer um das jeweilige Maximum, welche nicht nur in dieser Kombinationen nicht zur Verfügung stehen, sondern in deren *highend* Form auch sehr kostspielig sind. Des Weiteren stehen bei ADCs mit einer hohen Anzahl an Pins nur Bauformen (wie z.B BGA[12]) zur Verfügung, die sich zur Handbestückung nur sehr schwierig bis nicht verarbeiten lassen, was damit die Anforderungen an das System nicht erfüllen würde.

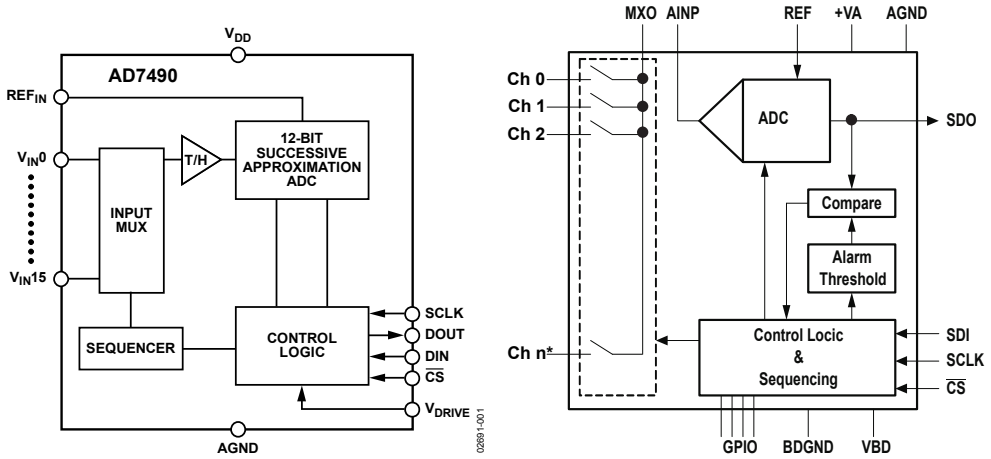
Bei der Auswahl eines passenden ADCs, sollte es sich um ein ADC mit Eigenschaften aus dem Mittelfeld handeln, welcher von allen bekannten Herstellern zur Verfügung steht und sowohl kostengünstig als auch leicht zu beschaffen ist. Für die Übertragen der zu erfassenden Daten stehen wie bereits erwähnt diverse Übertragungstechniken bereit. Allerdings muss dieser mindestens die *SPI* Schnittstelle besitzen, da nur über diese die erforderliche Geschwindigkeit, für die hohen Abtastrate, erzielen werden kann (vgl. 2.2.4). Die Abtastrate sollte damit im Bereich von 1 MSPS[13] liegen. Dadurch würde das Erfassen in der Theorie $160\mu s$ $(160Taxel * 1\mu s)$ dauern, welches deutlich im Rahmen der Anforderungen liegt. Da es sich bei dieser Arbeit um 160 Taxel pro Modul handelt, ist die Anzahl der Eingänge am ADC mit 16 vorzuziehen. Das hätte zur Folge, dass alle Eingänge der benötigten ADCs benutzt werden könnten. Im Fall der Methode aus 5.2.2, würden genau 10 ADC Bausteine benötigt, die dann über *SPI* angesprochen und ausgelesen werden können. Zusätzlich handelt es sich dabei um ADC Bausteine, die in der Regel $20 - 30$ Pins besitzen und sich somit gut verarbeiten lassen.

Zwei ADCs die diesen Anforderungen entsprechen, sind zum einen der *AD7490* der Firma *Analog Devices* [28] und der *ADS7957* der Firma *Texas Instrument* [38], welche in Form von Blockschaltbilder, in der Abbildung 5.9 dargestellt sind.

[11] **GSPS**: **G**iga **S**amples **P**er **S**econd

[12] **BGA**: **B**all **G**rid **A**rray, zu deutsch Kugelgitteranordnung, ist eine Gehäuseform bei der die Anschlüsse kompakt auf der Unterseite des Bauelements liegen.

[13] **MSPS**: **M**illion **S**amples **P**er **S**econd



(a) *AD7490*[28]. (b) *ADS79XX*[38].

Abbildung 5.9: *Blockschaltbilder der Analog Digital Wandler AD7490 und ADS79XX.*

Beide ADCs entsprechen, wie zuvor bereits erwähnt, den Anforderungen. Der ADC der Firma *Texas Instrument* stellt allerdings deutlich mehr Funktionen bereit, ist hingegen in seiner Konfiguration und Beschaltung etwas aufwendiger als der *AD7490*. Da sich ansonsten beide in der Theorie sehr identisch verhalten, wird nachfolgend nur noch der ADC der Firma *Analog Devices* betrachtet. Dieser besticht auch besonders durch seine einfache Inbetriebnahme hervor und entspricht damit auch in einem weiteren Punkt den gesetzten Anforderungen.

Analog Digital Wandler AD7490

Dieser ADC lässt sich einfach, über wenige Leitungen anschließen (siehe Abbildung 5.9(a)), ansprechen und in das System integrieren. Um die angestrebten $1MSPS$ zu erreichen, muss der ADC mit $5V$ Spannung versorgt werden. Des Weiteren bietet der ADC verschiedene Modi zum *Power Management*. Allerdings wird auch nur bei dem *Normal Mode* der höchste Datendurchsatz erreicht. Dieses hat den Grund, dass der ADC durchgehend mit der Versorgungsspannung betrieben wird und keine Zeit durch das *Power Management* verloren geht.

Durch das Fallen des $\overline{\text{CS}}$ -Signals beginnt die Wandlung und das so genannte *track-and-hold* begibt sich in den *hold-mode*. Dieses bedeutet, dass die Spannung für eine Gewissen Zeit gehalten wird um diese stabil zu messen. Nach 14 Takten kehrt der Modus wieder in den *track-mode* zurück. In dieser Phase muss der ADC mindestens $300ns$ verweilen, bevor er erneut in den *hold-mode* gehen kann. Nach 16 Takten steht das Ergebnis der Wandlung bereit. Dieses enthält $4Bit$ für die Port Zuweisung und $12Bit$ für den analog gewandelten Wert. Eine erneute Wandlung wird erneut initiiert, nach der Zeit t_{Ruhe} und durch erneutes ziehen der Leitung $\overline{\text{CS}}$ auf *LOW*. Die Länge der Zeit t_{Ruhe} ergibt sich abhängig aus der Taktrate, da die *hold* Phase

$300ns$ benötigt und erst nach dem 14 Takt in diese wechselt. Die schnellste $t_{Convert}$ Zeit wird vom Hersteller mit $800ns$ angegeben. Da es sich um 16 Takte handelt, werden pro Takt:

$$t_{Takt} = \frac{t_{Convert}}{t_{Takt}} = \frac{800ns}{16} = 50ns \tag{5.2}$$

benötigt. So befindet sich der ADC schon $100ns$ während der $t_{Convert}$ Zeit in der *track* Phase. Somit fehlen noch $200ns$, welche in der t_{Ruhe} bestritten werden müssen. Daraus ergibt sich eine Taktrate (SCKL) von $20Mhz$, da in der Zeit von $t_{Gesamt} = 1\mu s$ in 20 Takten absolviert werden müssen, um den Durchsatz von $1MSPS$ zu erreichen. Somit muss bei der Verwendung dieser maximalen Taktrate am Ende einer Wandlung noch mindestens $200ns$ (t_{Ruhe}) gewartet werden, bevor die $\overline{\text{CS}}$ erneut gezogen werden kann. Das gesamte Zeitverhalten ist dem Zeit Diagramm aus der Abbildung 5.10 zu entnehmen.

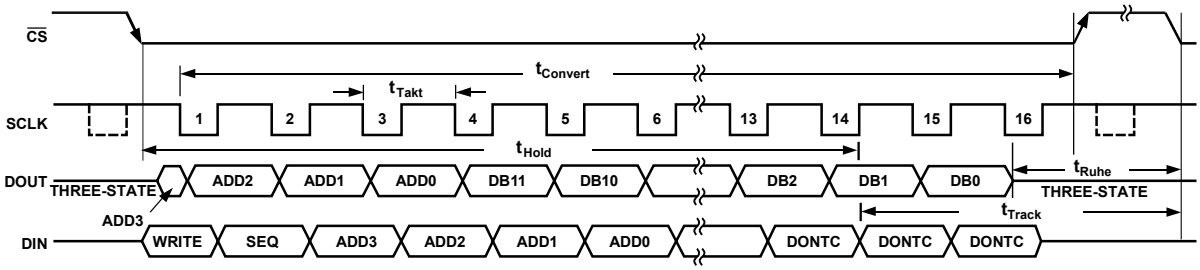


Abbildung 5.10: *AD7490 Zeit Diagramm der seriellen Kommunikation.*

Testaufbau

Um den *AD7490* zu verifizieren und die theoretische Datenrate der Datenerfassung zu bestätigen, wird ein Testsystem erstellt. Die gewonnen Informationen werden dann im nächsten Abschnitt ausgewertet und dienen im späteren Verlauf der Arbeit als Referenz, z.B. bei der Wahl eines Mikrocontrollers.

Wie schon in Teil 5.2, bietet sich auch in diesem Stadium der Arbeit ein Entwicklungsboard an, da in der Regel alle Schnittstellen des Mikrocontrollers herausgeführt sind und sich somit leicht ansteuern und verbinden lassen. Da der ADC allerdings besonders in seiner maximalen Geschwindigkeit betrieben bzw. getestet werden soll, kommt das System aus dem voran gegangen Testsystems nicht infrage. Denn der *Arduino MEGA2560* arbeitet nur mit einer Taktrate von $16MHz$ und stellt damit nur maximal $4MHz$ für die SCLK bereit. Dieses könnte in der finalen Umsetzung ausreichend sein, ist aber im Fall der Verifizierung nicht sinnvoll.

Um die Erfassung auf dieser hohen Geschwindigkeit von Seiten des *SPI Masters* möglichst unabhängig zu ermöglichen, wird hier der *UM232H-B*, ein *USB to Serial/Parallel Break-Out*

Module der Firma *FTDI*[14], eingesetzt. Mit diesem kann die *SPI* Kommunikation per USB erreicht werden. Dabei schafft dieser ohne Probleme die $20MHz$ und stellt damit eine optimales Testsystem für die ADC Evaluierung dar.

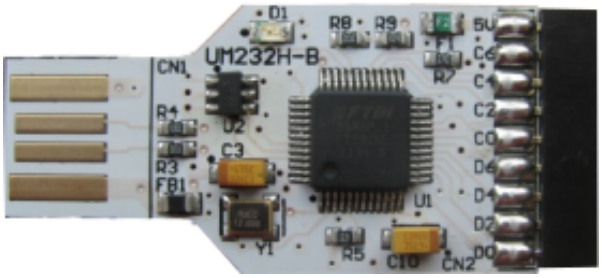


Abbildung 5.11: *UM232H-B USB to Serial/Parallel Break-Out Module*

Zur Erfassung der Daten der 16 Taxel über den ADC, wird die nötige Schaltung aufgebaut. Die Abbildung 5.12 zeigt alle nötigen Leitungen zur Ansteuerung eines *AD7490*. Der ADC und alle 16 Taxel werden mit $5V$ Spannung versorgt. Die Masse *AGND*[15] ist mit der des Systems verbunden. Zu beachte ist hier, dass in der Regel die Massen für die analogen und digitalen Schaltkreise getrennte werden und nur an einem Punkt zusammengeführt sind [vgl. 55]. Dieses wird in diesem Testsystem zunächst nicht berücksichtigt und deshalb eine einfache gemeinsame Masse verwendet. Der Pin V_{DRIVE} soll die Spannung zwischen dem ADC und dem seriellen Kommunikationspartner kontrollieren. So erlaubt der ADC eine einfache Verbindung zu $3V$ und $5V$ Prozessoren, welches z.B. bei einer $3,3V$ Betriebsspannung eines Mikrocontrollers, benötigt wird. An REF_{IN} ist eine externe $2,5V$ Referenz Spannungsquelle, *AD780*, angeschlossen. Da es sich bei den zu messenden Spannungen an den Taxel um einen Bereich von $0-5V$ handelt, muss das *RANGE Bit* (siehe Tabelle 5.2) auf Null konfiguriert sein. So wird die Referenz Spannung von dem *AD7490* verdoppelt. Die Ergebnisse der Analog-Digital-Wandlung werden in $16bit$ ausgegeben. Dieser $16bit$ Datenstrom besteht aus 4 Adressbits, zur Zuweisung des Wertes zu einem Kanal, gefolgt von 12 Datenbits.

Für die Übertragung der Daten sind die für SPI erforderlichen Leitungen geschaltet. Dabei handelt es sich um die serielle Taktrate (*SCLK*), den Dateneingang (*DIN*), sowie Datenausgang (*DOUT*) und der *Chip Select* Leitung ($\overline{\mathrm{CS}}$). Die drei erst genannten Leitungen (*SCKL*, *DIN* und *DOUT*) werden von allen Teilnehmern auf dem Bus gemeinsam benutzt, im Gegensatz zu der *Chip Select* Leitung, die für jeden Teilnehmer separat vorhanden sein muss.

In diesem Setup ist der *UM232H-B* der Master des Datenbus, somit stellt dieser die Taktrate bereit und bestimmt durch die $\overline{\mathrm{CS}}$ Leitung, welcher Teilnehmer auf dem Bus aktiv werden darf. Da es sich bei dem *Chip Select* Port um ein *Low* Aktives Signal handelt, muss die Leitung im Ausgangszustand auf *High* ($5V$) stehen.

[14] **FTDI**: **F**uture **T**echnology **D**evices **I**nternational Ltd
[15] **AGND**: **A**nalog **G**rou**ND**

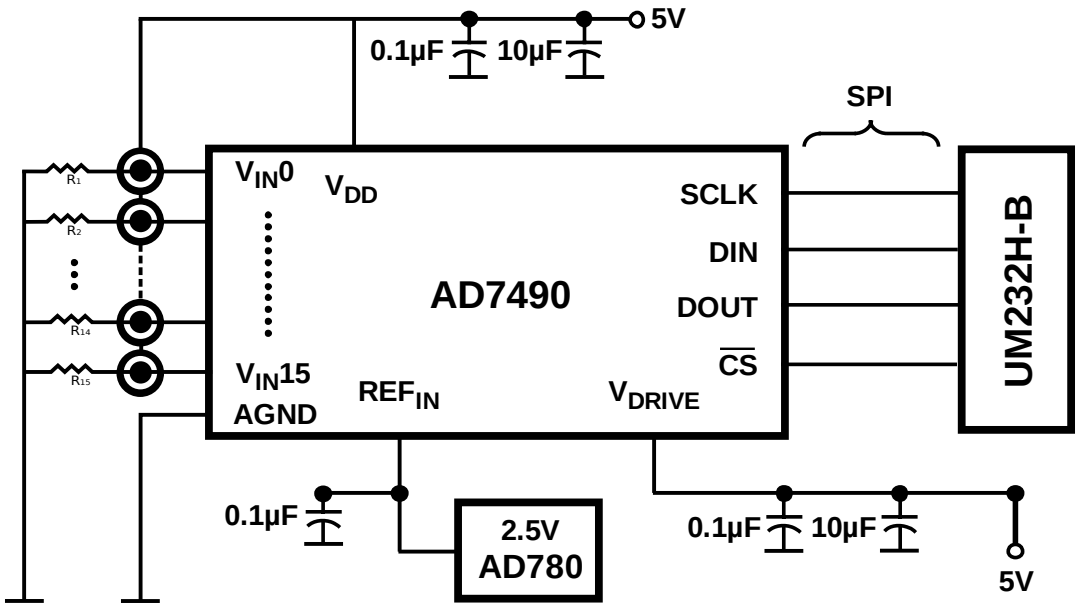


Abbildung 5.12: *Schaltplan eines AD7490 aus dem Testsystem der Erfassungseinheit mit angeschlossenem UM232H-B und der Referenzspannung AD780.*

Bevor der ADC Verwendet werden kann, muss dieser initial konfiguriert und eine der zur Verfügung stehenden Modi, zur Erfassung der 16 Ports, eingestellt werden. Diese gesamte Konfiguration wird durch das Schreiben in das $12Bit$ *Control Register* (siehe Tabelle 5.2) des ADCs getätigt. Da in diesem System jeder der 16 Ports eines ADCs gelesen werden muss, wird der sequentielle Modus verwendet, welcher der Reihe nach von Port 0 bis 15 iteriert und anschließend wieder bei 0 beginnt. Dazu wird das Register wie folgt zum ADC geschrieben:

Das *WRITE* Bit teilt dem ADC mit, dass es sich bei den Daten, um Daten für das Control Register handelt. Zum Auswählen des *Sequenz Modus* muss das *SEQ* und *SHAFOW* Bit gesetzt werden. Zusätzlich müssen die vier Bits *ADD0 - ADD3* gesetzt werden, damit durch alle 16 Ports iteriert wird. Die Bits *PM0* und *PM1* werden auch auf Eins gesetzt, was den *Power Mode* auf den normalen Betrieb stellt. Wie schon zuvor erwähnt, wird in diesem System ein Spannungsbereich von $0 - 5V$ benötigt, was das setzten des Bits *RANGE* erfordert. Die Bits *WEAK/TRI* und *CODING* werden außerdem auf Eins gesetzt. Da es sich bei der Übertragung immer um $16Bit$ Werte handelt, werden die letzten $4Bit$ mit Nullen gefüllt. Das komplette Register ist in der Tabelle 5.2 dargestellt.

11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WRITE	SEQ	ADD3	ADD2	ADD1	ADD0	PM1	PM0	SHADOW	WEAK/TRI	RANGE	CODING
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**

Tabelle 5.2: *ADC Control Register mit den gesetzten Bits für die Konfiguration.*

Zum testen der Genauigkeit bei der hohen Geschwindigkeit, werden vier definierte Spannungswerte ($0V$, $2,5V$, $3,5V$ und $5V$) jeweils an vier ADC Eingängen angeschlossen. Da es sich bei den Werten im späteren System um Spannungswerte im oberen Bereich von $5V$ handelt (vgl. 5.1), wurden diese Werte bewusst in diesem Bereich gewählt. So kann überprüft werden, ob jeweils alle 16 ADCs auch bei hohen Geschwindigkeiten die erwarteten Spannungen messen und ob die jeweils vier ADCs mit denselben Spannungen auch die exakt gleichen Ergebnisse liefern.

Zur Erfassung der Daten wird wie bereits beschrieben der *UM232H-B* benutzt. Dazu stellt *FTDI* einen Treiber zur Verfügung und eine handvoll Bibliotheken für diverse Programmiersprachen. In diesem Fall wird ein *Python*[16] Script erstellt, um die Daten zu erfassen und nachfolgend in der Auswertung darzustellen.

Auswertung

Die definierten Spannungswerte $0V$, $2,5V$, $3,5V$ und $5V$ wurden mithilfe des ADCs erfasst. Dieser lieferte die Daten bei einer Taktrate von $20MHz$, das entspricht in diesem Fall $1MSPS$. Die Daten wurden über den *UM232H-B* empfangen und konnten so ausgewertet werden. Die Ergbnisse von drei kompletten Messungen sind in der Abbildung 5.13 dargestellt. Dazu wurden die rohen analogen ADC Werte, zur besseren Veranschaulichung, in Spannungswerte umgewandelt.

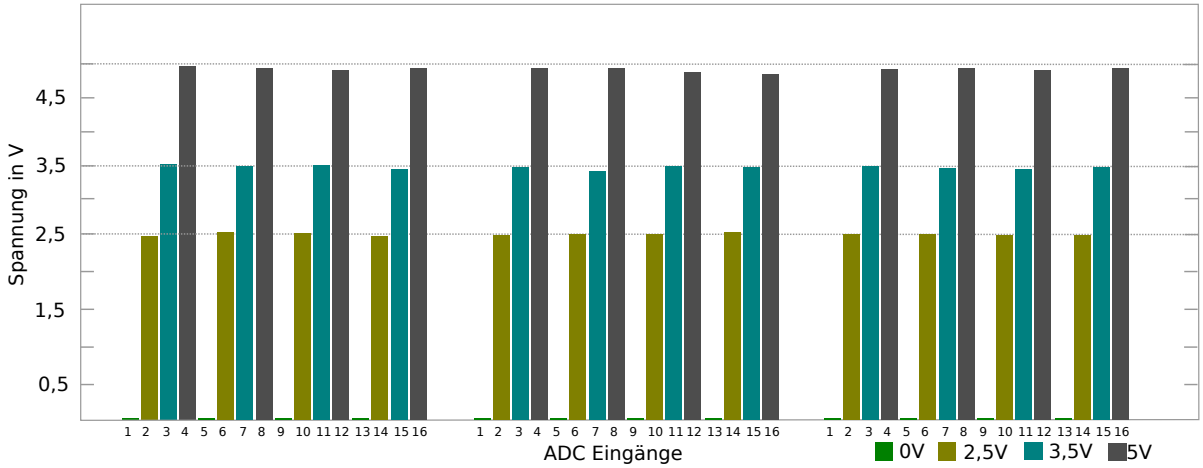


Abbildung 5.13: *Ergebnisse der ADC Wandlung bei $20MHz$ von definierten Spannungswerten: $0V$, $2,5V$, $3,5V$ und $5V$.*

Die Ergebnisse zeigen eindeutig, dass die Messungen auch bei maximaler Geschwindigkeit, selbst beim internen umschalten der Eingänge, stabil bleiben. Damit kann nachfolgend, soweit

[16] **Python** ist eine universelle, üblicherweise interpretierte höhere Programmiersprache

der *SPI Master* es zulässt, mit der vollen Geschwindigkeit kalkuliert werden. Des Weiteren sind diese Ergebnisse bei der Evaluierung der Taxel Erfassungsmethoden in dem nachfolgenden Abschnitt sehr wertvoll.

5.2.2 Taxel Erfassungsmethoden

In diesem Abschnitt wird beschrieben, wie die einzelnen Taxel ausgelesen werden. Das Verfahren der reinen Erfassung wurde in den vorherigen Abschnitten bereits ausführlich beschrieben. In diesem Teil steht der Fokus auf der Erfassung aller Taxel. Dabei spielen die Komplexität, Implementierung, Geschwindigkeit und Kosten der Schaltung eine primäre Rolle. Nachfolgend werden einige mögliche Methoden zu Datenerfassung aller 160 Taxel vorgestellt und anschließend ausgewertet.

Ein ADC Eingang pro Taxel

Bei diesem Verfahren, zur Erfassung aller 160 Taxel, steht auch die gleiche Anzahl an Analog-Digital-Wandler zur Verfügung. So kann jeder Taxel über seinen ADC *Port* direkt angesprochen werden und es benötigt keine komplexe Schalttechnik oder weitere *I/O Ports* für die Selektierung der Taxel. Die Erfassung der Daten wird von einem Mikrocontroller gesteuert, welcher die Daten der ADCs erfasst und weiterleitet. Dieses Modell wird nun nachfolgend ausführlicher beschrieben und zur besseren Veranschaulichung in einer Abbildung (siehe 5.14) dargestellt.

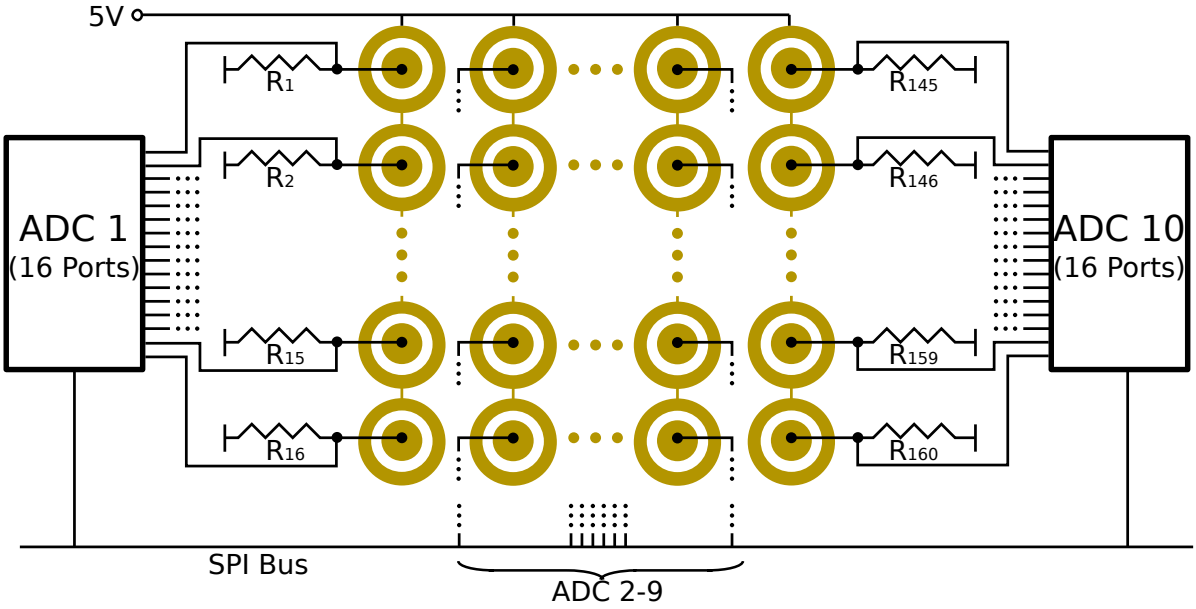


Abbildung 5.14: *Schaltung mit 10 ADCs mit jeweils 16 Eingängen zur Erfassung von 160 Taxel, mithilfe unbelasteter Spannungsteiler.*

Das Sensorarray eines Moduls besteht aus 160 Taxel die in 10 Reihen mit jeweils 16 Taxel aufgebaut sind (siehe Abbildung 5.8). So wird jedem ADC eine dieser Reihen zugeordnet und ein Taxel pro ADC Eingang angeschlossen. So ist jedem Taxel genau ein Eingang zugewiesen. Diese Schaltung ist in Abbildung 5.14 dargestellt. Eine ähnliche Schaltung wird auch in [81] und [79] verwendet und diente in Gewissermaßen als Vorbild.

Die 10 ADCs werden dabei über den SPI Bus ausgelesen. Dieser Bus wird von einem Mikrocontroller gesteuert, welcher zugleich die Aufgabe hat die Daten zu sammeln und diese für die nächsthöhere Instanz bereitzustellen.

Zur Erfassung der Taxel Daten über die 10 ADCs, wird die nötige Schaltung aufgebaut. Der Kern dieser Schaltung ist die Ansteuerung der ADCs. Jeder ADC hat seine eigene Teilschaltung, welche ähnlich wie in Abbildung 5.12, aufgebaut ist. Diese zeigt auch alle nötigen Leitungen zur Ansteuerung eines *AD7490*.

Die meisten Leitungen, werden von allen 10 ADCs gemeinsam genutzt. Die $\overline{\mathrm{CS}}$ Leitung sowie die Spannungsteilerschaltung jedes Taxels sind hingegen für jede der 10 Teilschaltungen extra zu veranschlagen. Somit werden am Mikrocontroller neben den 3 Leitungen zur Kommunikation noch 10 weitere für die Auswahl, durch die jeweilige $\overline{\mathrm{CS}}$ Leitung, des jeweils aktiven ADCs benötigt. Hier wäre ein Einsatz eines externen *Muliplexers* denkbar, wenn der Mikrocontroller die Anzahl der *I/O Ports* nicht bereitstellen kann. Des Weiteren wird für jeden Taxel ein Widerstand benötigt, was bei der Summer von 160 Taxel auch eine Anzahl von 160 Widerständen ergibt.

Ein ADC Baustein pro Sensormodul

Bei diesem Verfahren stehen zur Erfassung aller 160 Taxel nur 16 Analog-Digital-Wandler Eingänge zur Verfügung und somit wird nur ein ADC Baustein benötigt. So wird nicht jeder Taxel über seinen ADC Port direkt angesprochen, sondern teilt sich einen Port mit den neun weiteren Taxeln der gleichen Reihe. Diese Art von der Erfassung der Daten bezeichnet man auch als Matrix Schaltung. Diese Schaltung wird in vielen Arbeiten angewandt wie z.B. in [9] und [13]. Außerdem fand es, im Zuge eines Testaufbaus (vgl. 5.1), bereits in dieser Arbeit Einsatz. Dieses Modell wird nun nachfolgend ausführlicher beschrieben und deren Funktionsweise erläutert.

Um die einzelnen Sensordaten auszuwerten, werden die Reihen und Spalten der Sensor-Matrix mit dem Mikrocontroller verbunden. So kann jeder Sensor der Matrix über einen Zeilen und Spalten Wert aktiviert und ausgewertet werden. Zur Aktivierung wird durch die digitalen *Ports* des Mikrocontrollers eine Zeile auf Spannung gesetzt, in Abbildung 5.15 als *Demultiplexer* dargestellt. Hierzu wird an den Ausgängen eine „logische Eins“ durch eine Reihe von „logischen Nullen“ geschoben, so dass jeweils immer nur ein Ausgang und somit nur eine

Spalte zurzeit aktiviert ist. Wichtig ist hier, dass die Ports eine $5V$ Spannung liefern, ansonsten muss z.B. mithilfe eines Transistors diese von extern geschaltet werden. Die einzelnen Sensoren der aktiven Spalte werden nun iterativ durch den A/D Wandler, dessen Ports sich in den Reihen befinden, ausgelesen. Wurden alle Werte der Sensoren einer Spalte digitalisiert, wird die nächste Spalte, durch das weiter schieben der „Logische Eins“, aktiv geschaltet.

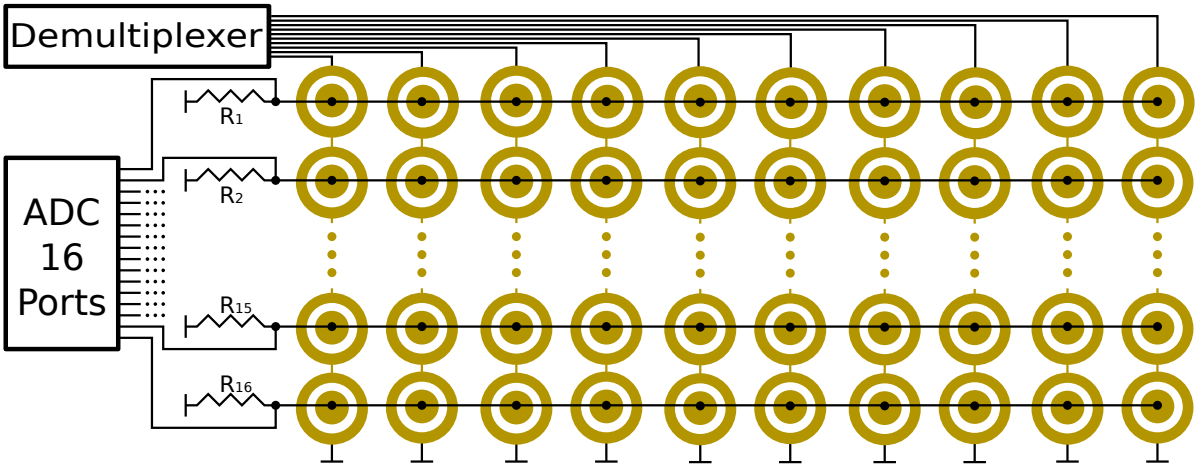


Abbildung 5.15: *Schaltung mit einem ADC (16 Eingängen) und einem Demultiplexer zur Datenerfassung, mithilfe eines unbelasteten Spannungsteilers, von 160 Taxel.*

Zur Erfassung der Taxel Daten über die den ADC, wird die nötige Schaltung aufgebaut. Die Schaltung ist dabei ähnlich wie in Abbildung 5.12 aufgebaut, welche alle nötigen Leitungen zur Ansteuerung des *AD7490* zeigt. Zusätzlich werden noch 10 Leitungen zur jeweiligen Selektierung der einzelnen Spalten der Sensor-Matrix benötigt. Auch hier wäre ein Einsatz eines externen *Demultiplexers* denkbar, wenn der Mikrocontroller die Anzahl der *I/O Ports* nicht bereitstellen kann.

Lokales Teilen eines ADC Eingangs

Eine weitere Methode zur Erfassung einer hohen Anzahl an Taxel, besteht aus einer Kombination der beiden zuvor vorgestellten Varianten. Taxel die möglichst lokal beisammen liegen, teilen sich einen ADC Eingang. In diesem Fall besitzen jeweils zwei Taxel-Spalten einen ADC Baustein. Somit teilen sich immer zwei Taxel einen ADC Eingang. Dieses ist in Abbildung 5.16 veranschaulicht.

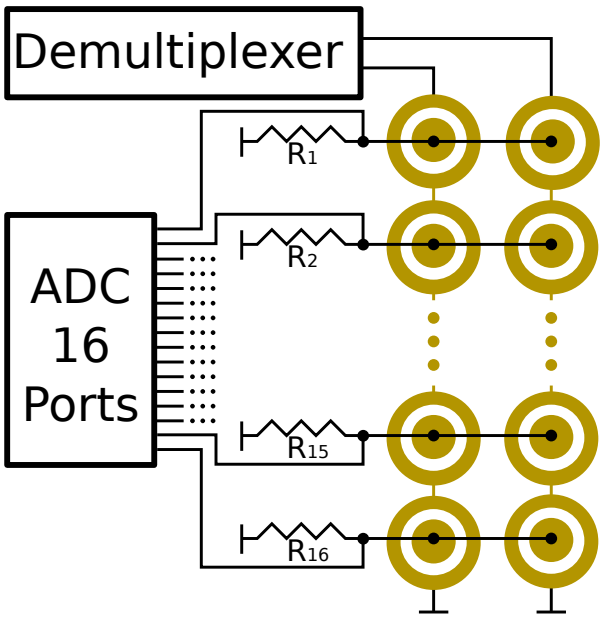


Abbildung 5.16: *Schaltung mit einem ADC (16 Eingängen) und einem Demultiplexer zur Datenerfassung, mithilfe eines unbelasteten Spannungsteilers, von 32 Taxel.*

Da es sich um 160 Taxel handelt und sich immer zwei Taxel einen Eingang teilen, werden hier fünf ADCs mit jeweils 16 Eingängen verwendet. Eine von fünf ADC Schaltungen ist in der Abbildung 5.16 dargestellt. Der *Demultiplexer* beschaltet dabei eine Spalte von 16 Taxel mit Spannung, so dass der ADC über die Spannungsteiler die analogen Werte digitalisieren kann. Nachdem alle 16 Taxel erfasst sind, wird die erste Reihe von der Versorgungsspannung getrennt und die zweite Reihe versorgt.

Jede Spaltengruppe, die durch die *Demultiplexer*-Anordnung in Untergruppen zu je zwei Spalten mit jeweils 16 Taxel zusammengefasst ist, ist eine übergeordnete ADC Schaltung. Dieser wird als ein einzelner elektronischer Baustein mit Ein- und Ausgängen für die Datenübertragung, einer Steuerleitung zum selektieren des *Demultiplexers* und einer Versorgungsleitungen (hier noch nicht aufgeführt) definiert. Jeder dieses Bausteins benötigt zusätzlich 16 Widerstände für die Spannungsteilerschaltung.

Die Steuerung der fünf ADC Schaltungen übernimmt dann der Mikrocontroller. Dieser selektiert über Standard Ein- und Ausgänge die einzelnen Spalten der Matrix und liest über den SPI Bus die Daten der ADCs ein. In der Abbildung 5.17 ist das System, um die ADC Datenerfassung dargestellt. Alle Komponenten, sowohl die Taxel als auch alle aufgeführten Bauteile zur Datenerfassung, befinden sich dabei auf der selben Platine.

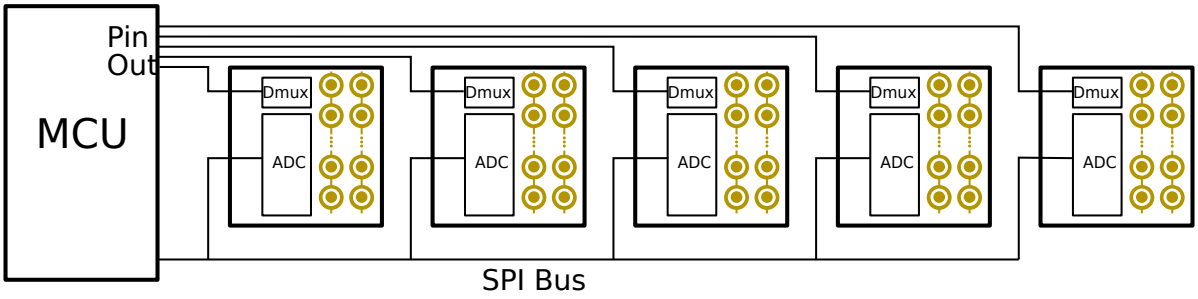


Abbildung 5.17: *Datenerfassungseinheit zur Erfassung von 160 Taxel mit Hilfe von fünf ADC Schaltungen.*

Dieser modulare Aufbau ermöglicht es ohne großen Aufwand die Anzahl der Sensoren zu erweitern. Zwar bedeutet die Erhöhung der Zahl der Taxel auch eine Geschwindigkeitseinbuße, ansonsten ist es aber durch die Verwendung der Chip-Select-Leitungen bei der SPI Schnittstelle, nahezu möglich beliebig viele ADC Schaltungen hinzuzufügen. Dabei sollte die Anzahl der Spalten immer im geraden Bereich (2,4,6,...) liegen und die Summe der Taxel der jeweiligen Spalten die gleiche Anzahl betragen.

Auswertung

Die Konzeption der verschiedenen Methoden zur Erfassung von einer großer Anzahl an Taxel haben gezeigt, dass es mehrere Möglichkeiten gibt dieses umzusetzen. Dabei bauten zwar alle drei Verfahren auf das selbe Grundkonzept auf, unterscheiden sich dennoch in Aufwand und Komplexität.

Bei der ersten Methode werden mit Abstand die meisten Bauteile, 160 Widerstände und 10 ADCs, benötigt. Dafür lässt sich diese Schaltung sehr leicht über den SPI Bus ansprechen und auslesen. Des Weiteren ist die Zuweisung des Taxel in der Matrix einfach, da jeder seinen festen ADC Eingang besitzt. Zusätzlich gibt die hohe Anzahl der ADCs eine gewissen Sicherheit. Der Ausfall eines ADCs hat zwar Auswirkungen auf die Ergebnisse, das gesamt System würde aber dennoch weiterhin, wenn auch eingeschränkt, funktionieren. Für die Schaltung werden neben den Leitungen für die Kommunikation über den SPI Bus, 10 $\overline{\mathrm{CS}}$ Leitungen zur Ansteuerung der einzelnen ADCs benötigt. Dieses hält sich alles im Rahmen der Anforderungen. Die hohe Anzahl der 160 Widerstände hingehen, bedeutet einen sehr hohen Aufwand bei der Erstellung des Moduls.

Bei der zweiten Methode, würde ein Ausfall eines Einganges oder sogar des gesamten Bausteines, das komplette System unbrauchbar machen. Dafür ist der große Vorteil dieses Modells, dass sehr wenige Bauteile benötigt werden, um eine hohe Anzahl an Taxel auszulesen. Dazu werden gerade mal ein ADC Baustein mit 16 Ports benötigt und zusätzlich 16 Widerstände für die jeweilige Spannungsteilerschaltung. Zwar reduziert sich durch die Verwendung nur eines

ADCs die $\overline{\mathrm{CS}}$ Leitungen auf nur eine, stattdessen werden aber 10 weitere zur Ansteuerung der einzelnen Spalten benötigt.

Die dritte und letzte hier vorgestellte Methode, ist eine Kombination der beiden zuvor vorgestellten Varianten. Damit besitzt sie ein Mittelmaß der jeweiligen Vor- und Nachteile dieser Verfahren. Hier wurde besonders die Modularität hervorgehoben, welche allerdings auch bei den beiden anderen möglich wäre. Der große Vorteil dieser Variante liegt ausschließlich in dem Mittelmaß. So kann mit einer deutlichen Verringerung der Bauteile eine nahezu gleiche Geschwindigkeit und Fehlersicherheit bei der Erfassung erlangt werden. Auf diesen Punkt wird an dieser Stelle nicht weiter eingegangen, da dieses in 5.3 weiterführend erläutert wird.

Welches Verfahren zum Einsatz kommt, muss individuell an die Anforderungen des Systems entschieden werden. So können auch besonders externe Bauteile oder die Implementierung der Software Auswirkungen auf das Konzept der Erfassung der Taxel nehmen, was besonders die Arbeit [79] gezeigt hat.

5.2.3 Auswertung Sensormodul

Die Konzeption und Evaluierung des Sensormoduls hat gezeigt, dass es eine Vielzahl an Möglichkeiten gibt, selbst hohe Anzahlen an Taxel auszulesen. Dabei gelingt dieses auch mit einfachen Verfahren und Komponenten und erreicht damit selbst hohe Geschwindigkeit und Präzision.

Nachfolgend wird nun auf den Mikrocontroller eingegangen, der sehr großen Einfluss auf das hier beschriebene Sensormodul nimmt.

5.3 Mikrocontroller

Wie im Konzept unter 4 beschrieben, wird die gesamte Erfassungseinheit von einem Mikrocontroller gesteuert. Dieser hat nicht nur die Aufgabe die Daten der ADCs zu lesen, sondern diese auch für den Weitertransport an eine höhere Instanz bereitzustellen und letztendlich zu versenden.

Die Anforderungen an den Mikrocontroller ergeben sich zu einem aus den getätigten Anforderungen unter 3.7 und zum anderen aus den bereits erarbeiteten Konzept. So sollte dieser in einer Gehäuseform vorliegen, die sich mit einfachen Mitteln verarbeiten lässt. Des Weiteren spielen generelle Faktoren wie Kosten, Komplexität und Verfügbarkeit des Chips eine wichtige Rolle. Aber auch Erfahrungen aus anderen Projekten fließen bei der Wahl mit ein. Da sich bereits konzeptionell auf eine Übertragung der Daten von dem ADC per *SPI* geeinigt wurde, ist diese Schnittstelle eine unabdingbare Voraussetzung. Daraus folgt auch, dass ein Controller mit einer System Taktrate unter $16MHz$ nicht in Frage kommen kann.

Zusätzlich wird für den Transfer der erfassten Daten eine weitere Schnittstelle, welche die Anforderungen an die Geschwindigkeit erfüllen muss, benötigt. Einige sind in der Tabelle 5.4 aufgeführt und werden im Kapitel 5.4 ausführlicher beschrieben.

Da es sich um keine sehr komplexen Aufgaben und keine großen Datenmengen handelt, wird die Größe des Speichers nicht so stark bewertet, sollte allerdings nicht vernachlässigt werden. Besonders bei der Verwendung der *USB* Schnittstelle, die durch den *USB Stack* schon einen großen Teil des Speicher beanspruchen kann (vgl.5.4.1).

Da es in einem System mehrere Erfassungseinheiten geben wird und die Anforderungen einen Modulen Ansatz anstrebt, sollte jede Einheit eindeutig zu identifizieren sein. Dieses kann über eine einfache Variable fest im Code geschehen, welche eine eindeutige Identifizierungen beinhaltet. So müsste diese aber bei jeder Änderung der Software individuell für jede Erfassungseinheit angepasst werden. Einfacher wäre es diesen Code auf dem Mikrocontroller persistent zu speichern. So kann eine neue Software eingespielt werden, ohne das eine neue Zuweisung nötig ist. Für so eine persistente Speicherung bieten viel Mikrocontroller einen EEPROM[17] Speicher an.

Als letzte wichtige Eigenschaft, muss er mindestens die Anzahl der I/O Schnittstellen bereitstellen die von den jeweiligen Konzepten aus 5.2 benötigt werden.

Wie auch schon im Bereich der ADCs stehen auch hier wieder eine Vielzahl verschiedenster Controller von diversen Herstellern zur Verfügung. Eine kleine Auswahl ist in der folgenden Tabelle 5.3 aufgelistet.

Hersteller	Bezeichnung	Taktrate	Schnittstellen	I/O	Sonstiges
Atmel	Atmega32U4	16MHz	I²C, SPI, UART, USB	26	2,7V - 5,5V, EEPROM, AVR
	ATXMEGA128A1U	32MHz	I²C, SPI, UART, USB	78	1,6V - 3,6V, EEPROM, AVR
Microchip	PIC32MX2XX	40MHz	I²C, PMP, SPI, UART, USB	19	2,3V - 3,6V, extern Oszillator, MIPS32® M4K™
NXP	LPC1768/69	100MHz	CAN, Ethernet, I²C, Microwire, SPI, UART, USB	70	2,4V - 3,6V, ARM® Cortex®-M3
Cypress	CY8C55	67MHz	I²C, SPI, UART, USB	36	2,7V - 5,5V, EEPROM, ARM® Cortex®-M3
Freescale	MK20DX256	72MHz (96MHz)	CAN, I²C, SPI, UART, USB	40	1,7V - 3,6V, EEPROM, ARM® Cortex®-M4

Tabelle 5.3: *Auflistung einiger Mikrocontroller die für den Einsatz in der Erfassungseinheit in Frage kommen.*

[17] EEPROM: **E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory

Für eine genauere Untersuchung werden nachfolgend einige der in der Tabelle 5.3 aufgeführten Mikrocontroller, genauer betrachtet. Bei der Auswahl sind die oben erwähnten Kriterien mit eingeflossen. Da im laufe der Arbeit bereits viel Erfahrung mit Controllern von *Atmel* gewonnen wurde, kommt dieser für einen möglichen Einsatz in Betracht. Das gleiche gilt für den *PIC32MX2XX* von *Microchip*. Beide sind kleine und einfache Mikrocontroller, die besonders im Hobbybereich starke Nachfrage besitzen. Bei den restlichen aufgeführten, handelt es sich schon um deutlich komplexere Einheiten. Aus diesem Grund wird zusätzlich nur noch auf den *MK20DX256* von *Freescale Semicondutor* eingegangen.

Eine weitere Voraussetzung ist es, dass zu den jeweiligen Mikrocontroller Entwicklerboards zur Verfügung stehen, was auch bei der Auswahl der Controller eine Rolle gespielt hat, um erste Tests ohne großen Aufwand durchführen zu können. Aus diesem Grund wird in den einzelnen Beschreibungen der Schaltungsaufbau vernachlässigt, da diese Boards über einfache Kabelverbindungen mit der jeweiligen Peripherie verbunden werden können. Zu beachten und bei der Auswertung zu berücksichtigen ist, dass steckbare Leitungen eine deutlich höhere Anfälligkeit an Fehlern besitzen. In der späteren Umsetzung wird dann angestrebt, die einzelnen Komponenten auf der Erfassungseinheit über Leiterbahnen zu verbinden. Dieses ist besonders bei hohen Taktraten empfehlenswert.

Obwohl die Geschwindigkeit bei der Erfassung der Daten an den ADCs eine primäre Rolle spielt, ist dieses nicht das einzige Kriterium. Es darf auch bei dieser Einheit der Blick auf das gesamte System nicht verloren gehen. So werden bei der genaueren Betrachtung auch Form, Ausstattung und Handhabung genauer beschrieben. Neben der detaillierteren Beschreibung der einzelnen Mikrocontrollern, werden die Daten der ADCs über die *SPI* Schnittstelle erfasst, um erste Leistungen zu verifizieren. Die Übertragung dieser erfassten Daten, an die nächst höhere Ebene, soll aber nicht Thema dieses Abschnittes sein und wird später in den Sektionen 5.4.1 und 5.4.2 ausführlich fortgeführt. Die Ergebnisse werden abschließend unter anderem Zusammengefasst und Ausgewertet.

5.3.1 Evaluierungsverfahren

Zur genaueren Evaluierung der drei Mikrocontroller, werden jeweils die ADC Werte ausgelesen. Dabei wird immer mit der maximal möglichen SPI Taktrate *SCK* des jeweiligen Gerätes gearbeitet. Bei der Software für das auslesen der Daten wird zuerst ein allgemeiner Ansatz gewählt, so dass die Software für alle drei Controller bis auf die unterste Abstraktionsschicht identisch ist (siehe 5.1). So kann über die Funktion `ad7490_read_all()`, unabhängig vom Controller, der ADC ausgelesen werden. Die Funktion `spi_txrx16()` enthält dann, je nach Konfiguration, den spezifische Code der einzelnen Controller.

Bei der Erfassung der Daten wird zu erst eine *Polling* Umsetzung gewählt. Bei dieser Methode steuert die Software, und somit die CPU, die komplette Transaktion. Sie schreibt die zu übertragenden Daten in das richtige Register, startet die Übertragung und wartet darauf (*Polling*),

dass die Übertragung abgeschlossen ist, um anschließend die Ergebnisse aus dem Register zu verarbeiten. Dieser *Polling mode* ist dabei einfach zu steuern und umzusetzen.

```
1  int ad7490_read_all(ad7490* device, uint16_t* values)
2  {
3      int i;
4      uint16_t dummy=0;
5
6      for(i=0;i<16;i++)
7      {
8          spi_chip_select(device->chip_select, LOW);
9          spi_txrx16(&dummy, &values[i]);
10         spi_chip_select(device->chip_select, HIGH);
11     }
12     return 0;
13 }
```

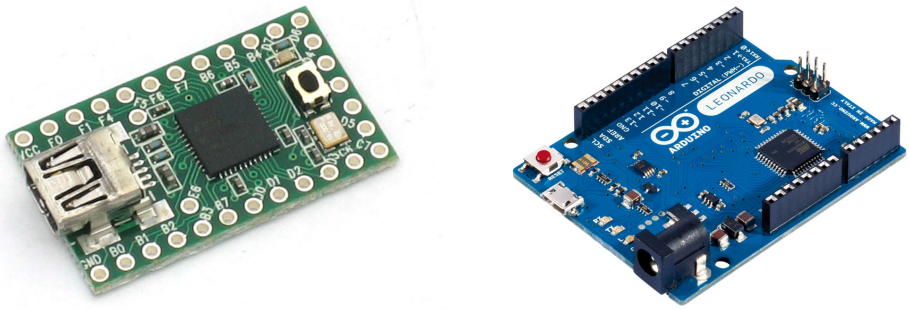
Listing 5.1: *Methode zum lesen aller Werte eines AD7490.*

Um bei der Erfassung weiter die Geschwindigkeit zu verbessern, wird neben der Optimierung des Codes bei dem *Polling mode* Ansatz, auch die *Direct Memory Access* (DMA) Methode implementiert und ausgewertet. Bei dem DMA Verfahren wird die Transaktion zwar von der CPU eingeleitet, nachfolgend kümmert sich aber der *Peripherie DMA Controller*(PDC) um die Übertragung. Der PDC erhält einen *Buffer* mit Daten die er senden soll und einen für die zu empfangenden Daten. Zusätzlich besteht die Möglichkeit dem PDC einen zweiten *Buffer* zu übergeben, in den er automatisch wechselt, wenn der erste übertragen bzw. gefüllt wurde. Je nachdem wie die SPI Schnittstelle konfiguriert wurde, bekommt die CPU Interrupts, wenn die Übertragung fertig ist. Dadurch wird die Zeit, in der die CPU benutzt wird, deutlich geringer. So kann neben der gewonnen freien CPU Zeit, in der andere Dinge getätigt werden können, auch mit einem Geschwindigkeitsvorteil gerechnet werden. Dieser DMA Controller steht allerdings nicht bei allen Mikrocontrollern zur Verfügung.

Um bei der Evaluierung auch den Hardware Aspekt nicht zu vernachlässigen, wird auch jede der vorgestellten Modelle zur Datenerfassung der Taxel aus 5.2.2, jeweils mit jedem Mikrocontroller durchlaufen, um so am Ende eine gute Gegenüberstellung der Mikrocontroller, im Bezug auf Hardwareaufwand, den elektronischen Aufbau und die Geschwindigkeit, zu erhalten. Diese Ergebnisse werden in der Auswertung 5.3.5 ausführlich erörtert.

5.3.2 Atmel Atmega32U4

Der *Atmega32U4* von *Atmel* ist ein $8Bit$ Mikrocontroller mit $32KByte$ *Flash* Speicher und einem *USB 2.0 Full-Speed Controller*. Des Weiteren besitzt er eine *SPI*, *UART* und *I2C* Schnittstelle. Er steht in einem TQFP-Gehäuse $(10x10mm)$ mit 44 Pins zur Verfügung und lässt sich damit auch mit einfachen Mitteln in eine Schaltung integrieren. Die maximale Geschwindigkeit des Systemtakts beträgt $16MHz$, welche nur bei einer Versorgungsspannung von mehr als $4,4V$ erreicht werden kann. Zur Generierung dieses Takts sorgt ein interner Oszillator.



(a) Teensy 2.0[56] (b) Arduino Leonardo

Abbildung 5.18: *Zwei komplett USB-basierter Mikrocontroller Boards zur Entwicklung auf dem ATMEGA32U4.*

Er stellt mit seiner $16MHz$ Taktrate, im Vergleich zu den anderen Modellen, zwar die geringste Taktrate zur Verfügung, erreicht damit aber die in der Anforderung definierten Geschwindigkeit. Des weiteren verfügt er über die notwendige SPI Schnittstelle. So kann der Bus mit einer Taktrate von $\frac{f_{CLK}}{4} = \frac{16MHz}{4} = 4MHz$ (vgl. 3.6.2) betrieben werden. Auch der interne RAM Speicher mit $256MB$ ist ausreichend groß, um die Messergebnisse für den Abruf der Daten bereitzustellen. Durch den *High Speed USB Port* bestehe auch die Möglichkeit, die gesammelten Daten an die nächst höhere Instanz weiterzuleiten. Möglich ist hier eine Übertragung mit bis zu $12Mbit/s$. Dabei handelt es sich allerdings um die Brutto Übertragung.

Die Wahl für einen *ATMEL* Mikrocontrollers wurde aufgrund der zuvor gewonnen Erfahrungen durch das *Arduino MEGA2560* gewählt, welcher diesem sehr ähnlich ist. Allerdings stellt er deutlich mehr Peripherie zur Verfügung, welche in dieser Arbeit nicht benötigt wird und damit die Komplexität erhöht. Zusätzlich fehlt ihm eine USB Schnittstelle, welche eine mögliche Kommunikation zur nächsten Instanz darstellen könnte.

Entwicklungsboards mit dem *ATmega32U4* stehen von vielen Herstellern zur Verfügung, bei denen alle Schnittstellen des Mikrocontrollers herausgeführt sind und sich somit leicht ansteuern und verbinden lassen. Auch hier bietet *Arduino*, mit seinem *Leonardo* (siehe Abbildung 5.18(b)), wieder ein mögliches Entwicklungsboard für den genannten Mikrocontroller an. In diesem Fall wird allerdings auf das *Teensy 2.0* Entwicklungsboard zurückgegriffen. Dieses unterscheidet sich besonders durch seine viel kleinere Bauform und könnte so im späteren Verlauf auch in Prototypen oder sogar in ein komplettes System, integriert werden.

Der Mikrocontroller auf dem Teensy Board kann mithilfe der von Arduino gestellten Entwicklungsumgebung programmiert werden. Diese steht dem Benutzer kostenlos zur Verfügung. Als Sprache dient eine Abwandlung von *Processing*[18]. Diese Sprache macht es auch Nutzern mit

[18] Processing ist eine objektorientierte, stark typisierte Programmiersprache

wenig Programmierkenntnissen möglich, kleine Programme zu schreiben. Da die Sprache wie eben erwähnt sehr einfach gehalten ist und viel dem Compiler überlassen wird, wird der Programmcode ausschließlich in *C* programmiert. Wie jeder AVR Kernprozessor lässt sich auch bei diesem der Programmcode mit dem *avr-gcc* Compiler bauen.

5.3.3 Microchip PIC32MX250

Der Controller *PIC32MX250* von *Microchip* ist ein $32Bit$ Mikrocontroller mit $128KByte$ *Flash* Speicher und einem *USB 2.0 Full-Speed Controller*. Er besitzt neben jeweils zwei *SPI*, *UART* und *I2C* Schnittstellen auch 19 I/O Ports. Als Gehäuse stehen 4 Bauformen zur Verfügung. Dabei handelt es sich auch, wie schon bei dem *Atmega32U4*, um Gehäuse die sich mit einfachen Mitteln in eine Schaltung integrieren lassen. Wie in Abbildung 5.19(a) zu sehen ist, besitzt der Controller nur 28 Pins. Der Großteil dieser Pins ist mit mehreren Funktionen belegt, die in Software geschaltet werden müssen. Dieses vereinfacht zwar hardwaretechnisch seine Handhabung, könnte aber aufgrund der geringen Anzahl schnell an seine Grenzen kommen, wenn es darum geht viele externe Komponenten zu steuern und zu erfassen. In dieser Arbeit reicht diese Anzahl dieser Pins allerdings aus, was durch die vorherigen Konzepte unter 5.2 verifiziert wurde. Auch hier steht ein *High-Speed USB Port* zur Verfügung, mit der möglichen Übertragung von bis zu $12Mbit/s$.

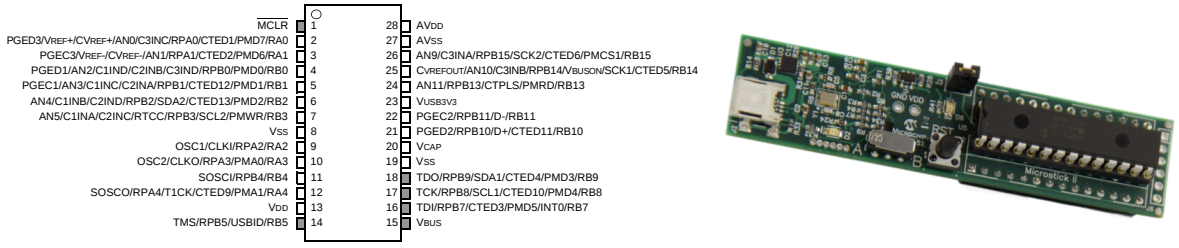


Abbildung 5.19: *Darstellung der Pin Konfiguration des PIC32MX2XX und des Entwicklungsboards Microstick 2.*

Er besitzt einen maximalen Systemtakt von $40MHz$. Diese kann sowohl durch den internen $8MHz$ FRC Oszillator erzeugt werden, als auch durch einen externen. Dieser externe Oszillator wird in dieser Arbeit bevorzugt, da nur durch diese Verwendung, die Taktrate für den USB Controller erzeugt werden kann (nicht in der Abbildung 5.20 dargestellt). Dazu muss der Oszillator wie mit ■ in Abbildung 5.20 dargestellt, konfiguriert werden, um die $40MHz$ zu erreichen. Zusätzlich wird auch die *SPI CKL* über diesen externen Oszillator gespeist und benötigt zusätzlich noch den in ■ dargestellten Pfad. Durch diesen Systemtakt lässt sich der ADC über eine maximale Geschwindigkeit von $10MHz$ betreiben: $\frac{f_{CLK}}{4} = \frac{40MHz}{4} = 10MHz$.

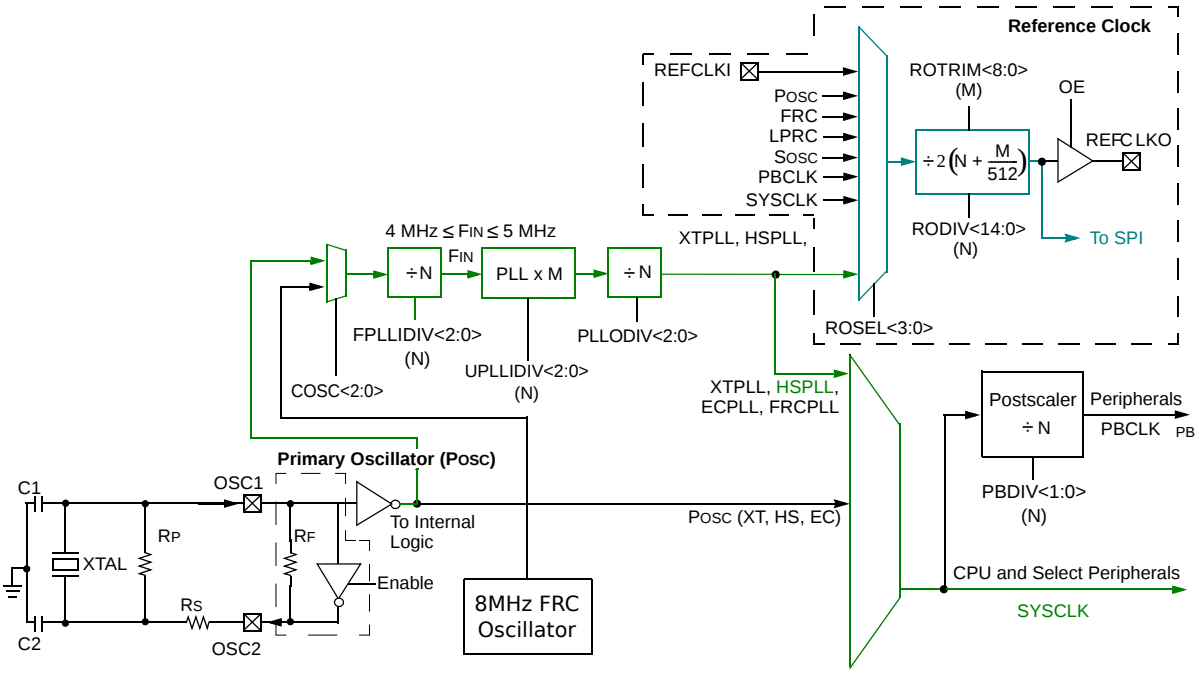


Abbildung 5.20: *Stark vereinfachtes Oszillator System Block Diagramm der PIC32 Familie aus [50], mit dem Fokus auf den Pfad der SPI CLK (in Grün und Blau dargestellt).*

Im Vergleich zum *ATmega32U4* stehen zwar auch Alternativen bei der Wahl eines Entwicklungsboards bereit, allerdings nicht in der Vielzahl. In diesem Fall wird der *Microstick 2* (siehe Abbildung 5.19(b)) von *Microchip* selbst verwendet. Auch hier sind alle Pins herausgeführt. Da es sich bei dem *Microstick 2* nicht ausschließlich um ein Entwicklungsboard für den *PIC32MX250* handelt, können auch viele andere Mikrocontroller mit der passenden Bauform auf das Board gesteckt werden. Ansteuern und Programmieren lässt sich auch dieses Board einfach über einen USB Anschluss.

Für die Erstellung der Software stellt *Microchip* eine eignende Entwicklungsumgebung bereit, die sich *MPLab* nennt. Auf diese ist man natürlich nicht angewiesen, anders als bei dem Compiler. Hier kann nicht auf den freien *GCC* zurückgegriffen werden, sondern es muss der Compiler *XC32* vom Hersteller benutzt werden. Dieser steht in der freien Version nur mit Einschränkungen zur Verfügung, ist für diesen Zweck der Arbeit aber ausreichend.

5.3.4 Freescale MK20DX256

Auch hier handelt es sich um einen $32bit$ Mikrocontroller mit einer *ARM* Prozessor Architektur. Dieser kann mit bis zu $96MHz$ betrieben werden und besitzt diverse Schnittstellen, z.B. SPI, USB und UART. Außerdem einen $256K$ *Flash Memory*, $64K$ *RAM*, sowie $2K$ *EEPROM*, zudem 34 digitale *I/O Pins*, welche über 5 Volt Toleranz verfügen. Des Weiteren steht ein DMA

Controller mit bis zu 16 Kanälen zur Verfügung, welcher sich auch mit der SPI Schnittstelle verbinden lässt.

Als Entwicklungsboard bietet sich auch hier eins von dem Drittanbieter *PJRC* an. Dieser stellt mit dem *Teensy 3.1* (dargestellt in 5.21) ein Board, bestückt mit diesem Mikrocontroller zur Verfügung. Angefangen bei der wirklich kleinen Dimensionierung und dem Gewicht- $(2,79g)$, Größe- $(3,6cmx1,8cm)$, Leistungsverhältnis findet er vor allem im Modellbau und bei MCU-Projekten mit geringen Platzverhältnissen viel Einsatz.

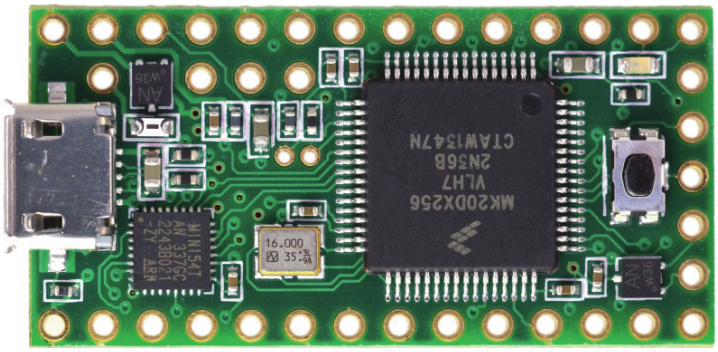


Abbildung 5.21: *Teensy 3.1 bestückt mit einem MK20DX256*

Die System Taktrate ist zwar offiziell mit $72MHz$ angegeben, kann aber auf $96MHz$ übertaktet werden. Aus dieser übertakteten Taktrate ergibt sich eine maximale Taktrate für die *SCK* von $\frac{96MHz}{4} = 24MHz$. Dieses ist allerdings für den ADC Baustein zu hoch. Da die Taktrate nicht frei wählbar ist und immer über Vorteiler generiert wird, ist die maximale Taktrate des ADC nicht zu erreichen. Die Taktrate für den Takt der SPI Schnittstelle wird bei diesem Mikrocontroller aus den folgenden drei Parametern und der Rechnung 5.3 generiert:

- **PBR** *Prescaler* Baudrate mit den Werten: $2,3,5,7$
- **DBR** Double Baudrate *Scaler* mit den Werten: $0,1$
- **BR** Baudrate *Scaler* mit den Werten: $2,4,6,8,16,32,\ldots,256$

$$SCK = \frac{F_{SYS}}{PBR} * \frac{1+DBR}{BR} = \frac{96MHz}{2} * \frac{1+0}{2} = 24MHz \tag{5.3}$$

Um nun der $20MHz$ möglichst nahe zu kommen, wird die Taktrate wie folgt berechnet:

$$SCK = \frac{96MHz}{5} * \frac{1+1}{2} = 19.2MHz \tag{5.4}$$

Als Entwicklungsumgebung kann auch hier die mit den nötigen Treibern upgedatet Arduino IDE verwendet werden. Aber wie schon bei dem *Atmega32U4*, wird der Programmcode ausschließlich in *C* und *C++* erstellt. Auch bei einem ARM Kernprozessor lässt sich der Programmcode

mit einem *GCC* Compiler (*arm-gcc*) bauen, welcher in diesem Fall auch für die Erstellung der Software eingesetzt wird.

5.3.5 Auswertung Mikrocontroller

Alle drei zuvor vorgestellten Mikrocontroller erfüllen die geforderten Anforderungen. So besitzt jeder mindestens eine SPI Schnittstelle und eine weitere Schnittstelle die sich für den Weitertransport der Daten eignet. Des Weiteren sind alle drei Modelle sowohl in ihrer Anschaffung als auch bei der Anwendung einfach und verfügbar. Besonders die beiden Mikrocontroller von *Freescale* und *Atmel*, bestückt auf den beiden Teensy Boards, sind in ihrer Abmessung extrem klein und es stehen sehr viele Code Beispiele zur Verfügung.

In Sachen Geschwindigkeit variieren die drei Controller allerdings sehr stark. Hier kann besonders der *MK20DX256* Punkten. Da dieser aber auch den höchsten Systemtakt hat, ist dieses auch das erwartete Ergebnis. Die unterschiedlichen Zeiten, für die Erfassung eines ADCs, sind zum Vergleich in Abbildung 5.22 dargestellt. Darin enthalten ist auch die jeweilig maximal theoretische Zeit abgebildet.

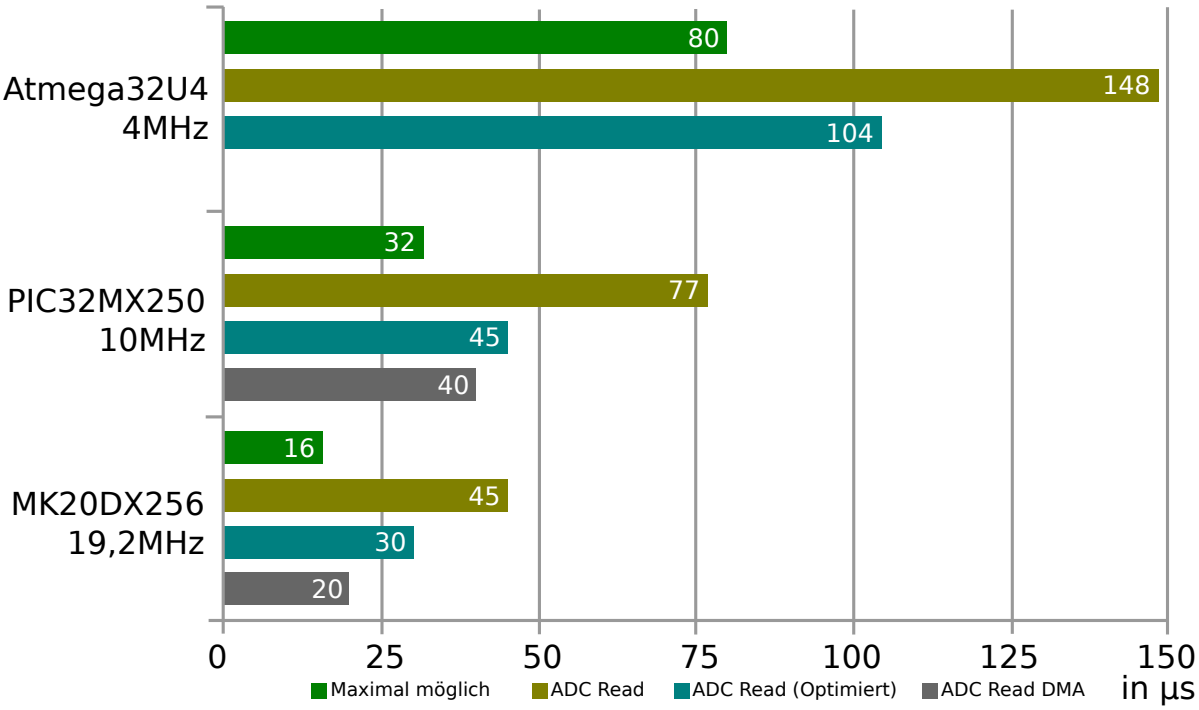


Abbildung 5.22: *SPI Benchmark eines ADC an den drei Mikrocontrollern.*

Für die tatsächlich benötigte Zeit, wurde in allen Fällen mit der maximalen *SPI SCK* des jeweiligen Controllern gemessen. Der ADC war zum Zeitpunkt der Messung bereits komplett konfi-

guriert und der sequentielle Modus eingestellt. Bei der ersten Messung der 16 Werte, wurde die allgemeine Funktion *ad7490_read_all()* benutzt, die je nach eingestelltem Mikrocontroller die gerätespezifische Methode zum lesen aufruft (vgl. 5.1).

Da durch diese sehr allgemein gehaltene Funktion, viel Zeit durch die Abstraktion verstreicht, wurde der gleiche Test noch einmal mit einer stark vereinfachten und somit optimiertem Code durchgeführt. Dazu wurde in der *ad7490_read_all()* Funktion direkt auf die Register zugegriffen (siehe 5.2).

```
1  int ad7490_read_all(ad7490* device, uint16_t* values)
2  {
3      int i;
4      uint16_t dummy=0;
5
6      for(i=0;i<16;i++)
7      {
8          digitalWrite(CS_PIN1, LOW);
9          SPI0_MCR |= SPI_MCR_CLR_RXF;
10         // Write to register
11         SPI4T3_WRITE_16(0, 0, 10);
12         SPI4T3_WAIT();
13         // Reade from register
14         values[i]=SPI0_POPR;
15         digitalWrite(CS_PIN1, HIGH);
16     }
17     return 0;
18 }
```

Listing 5.2: *Methode zum lesen aller Werte eines AD7490, am Beispiel des Mikrocontrollers MK20DX256, mit optimiertem Code.*

Die Ergebnisse zeigen, dass in allen Fällen die Zeit für die Erfassung dadurch fast halbiert wurde. Da durch diesen statischen Code, die Zeitersparnis so groß ist, wird nachfolgend versucht weiterhin mit möglichst wenig Schritten die Daten auszulesen.

Eine Weitere Verbesserung konnte durch die Verwendung eines DMA Controller erzielt werden. Dieser steht allerdings bei dem *Atmega32U4* nicht zur Verfügung.

Die eben gemessenen Werte bezogen sich alle nur auf die Erfassung von 16 ADC Werte und somit auf 16 Taxel. Um die Geschwindigkeit bei der Erfassung aller 160 Taxeldaten zu erhalten, wurden diese jeweils mithilfe der drei unter 5.2 erarbeiteten Modelle, ausgelesen. Dabei wurde immer der jeweils schnellste Ansatz zur Erfassung gewählt. Die Ergebnisse sind in Abbildung 5.23 dargestellt.

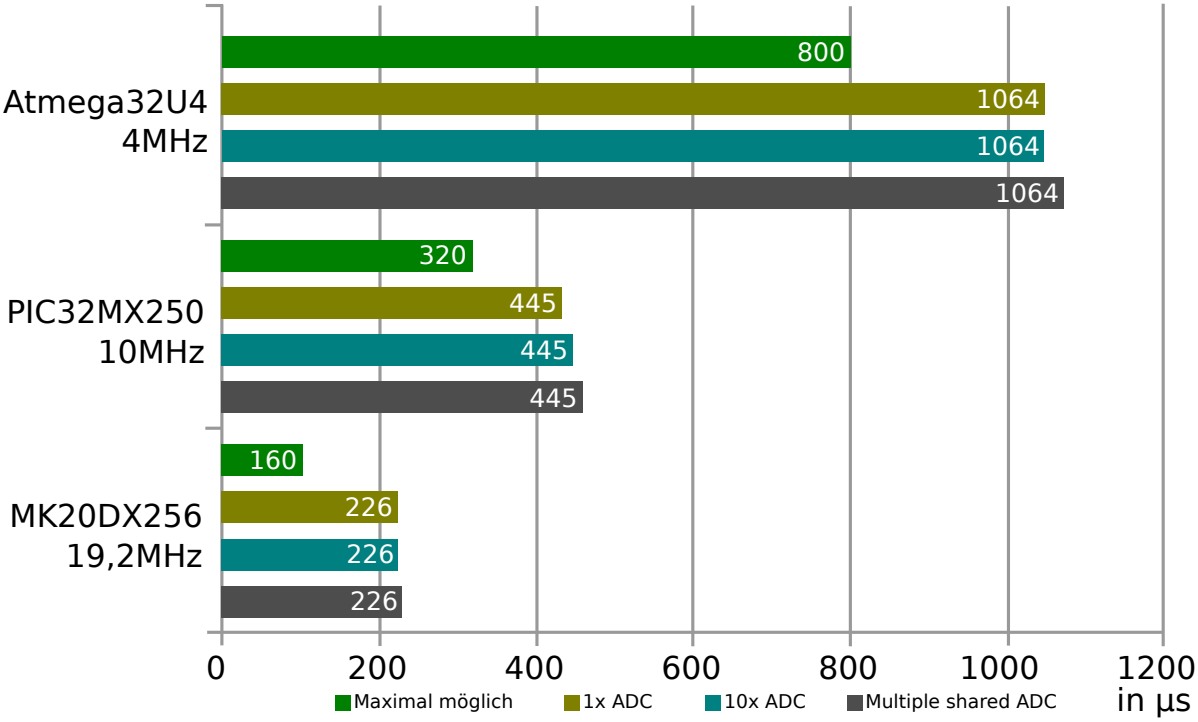


Abbildung 5.23: *SPI Benchmark der Erfassung aller 160 Taxel an den drei Mikrocontrollern, mithilfe der drei Erfassungsmodellen aus 5.2.*

Diese Ergebnisse geben dabei nur eine grobe Einschätzung, in wieweit sich die Modelle, bezogen auf die Geschwindigkeit, verhalten. Der entscheidendste Nutzen liegt dabei zum einen in der Unterscheidung der Hardware, bzw. der des elektronischen Aufbaus der Schaltung, und zum anderen kann sich eines der verschiedenen Modelle, je nach Software Implementierung, als am geeignetsten erweisen.

5.4 Datenübertragung

Bei der Übertragung handelt es sich um die Daten, die von der Erfassungseinheit gesammelt und nun an die nächst höhere Instanz übertragen werden sollen. Dazu bieten sich zu nächst eine Vielzahl möglicher Übertragungstechniken an. Die verbreitetsten sind in der Tabelle 5.4 dargestellt. Die darin aufgeführten, sind alle die zuvor in der Analyse näher untersucht wurden. Begutachtet man die Anforderungen und lässt die zuvor getätigte Analyse dabei mit einfließen, beschränkt sich die Auswahl auf einige wenige. So fallen die Schnittstellen *I²C*, *Microwire* und *RS 232* durch ihre ungenügende Geschwindigkeit weg. Hingegen liefern die beiden Schnittstellen *RS 485* und *Ethernet* zwar deutlich die Geschwindigkeit und die ausreichende Länge der Übertragsstrecke, dennoch kommen auch diese aufgrund ihrer komplexen Implementierung,

hardware- als auch softwaretechnisch, nicht für diesen Abschnitt der Übertragung in Frage. Des Weiteren wurde Aufgrund des konzeptionellen Ansatzes aus Kapitel 4 die Übertragung über kabellose Verfahren vernachlässigt. Diese bieten in einigen Fällen zwar die erforderlichen Eigenschaften, vgl. Kapitel 3, spielen aber, aufgrund ihrer nicht garantierten Zuverlässigkeit keine Rolle.

Schnittstelle	Übertragungsformat	Geräte	Länge	Geschwindigkeit
USB	asynchron, seriell, differentiell	127	5m	1,5M, 12M, 480M
SPI	synchron, seriell	>8	3m	>20M
RS 232	asynchron, seriell	2	15-30m	<500K
RS 485	asynchron, seriell, differentiell	>32	1200m	>10M
I²C	synchron, seriell	40	5m	<1M
Microwire	synchron, seriell	8	3m	<2M
Parallel Port	parallel	2	10m	8M
Ethernet	seriell, differentiell	1024	500m	10M, 100M, 1G

Tabelle 5.4: *Vergleich verbreiteter Mikrocontroller Schnittstellen.*

In dieser Arbeit kommen, unter den gesetzten Anforderungen und passend zu dem gewählten Konzept, die *USB* und *SPI* Schnittstelle für die Übertragung der Daten in Frage. Eine ausführliche Beschreibung und Detailbetrachtung der jeweiligen Übertragungstechniken, wird in den zwei nachfolgenden Sektionen durchgeführt.

5.4.1 USB Übertragung

USB ist ein weit verbreitetes Übertragungsverfahren, welches von vielen Mikrocontrollern bereit gestellt wird. Es bietet mit nur vier Leitungen sehr hohe Geschwindigkeiten von bis zu $480 MBit/s$ (Stand USB2.0) an. Damit stellt es beste Voraussetzungen, für die Übertragung der Daten der Taxel, bereit. Auch die Leitungslängen und die Anzahl der zulässigen Geräte entspricht, sowohl allen Vorgaben in den Anforderungen, als auch dem zuvor erarbeiteten Konzept.

Die USB Schnittstelle ist zudem ein sehr preisgünstiges verfahren, was besonders durch seine einfache Benutzung und Implementierung überzeugt. Des Weiteren besitzt heutzutage fast jedes moderne Geräte einen USB Anschluss, an den bis zu 127 Geräte angeschlossen werden können. Dabei muss es sich bei den Geräten nicht um den selben Typ von Gerät handeln. Ein weiterer Vorteil ist, dass die Schnittstelle eine $5V$ Spannung liefert, die es erlaubt das gesamte Gerät mit dieser Spannung zu versorgen. Das bietet sich besonders bei kleineren Geräten an und könnte auch in diesem System einen Einsatz finden.

Die erwähnte Geschwindigkeit von $480MBit/s$ bezieht sich auf *USB2.0* und das darin enthaltene *Hi-Speed* Protokoll. Dieses steht bei einfachen Mikrocontrollern mit geringeren System Taktraten nicht zur Verfügung. Weiter bei Mikrocontrollern verbreitet, ist das *USB 1.0* Protokoll. Hier steht mindestens das *USB Low-Speed*, mit Datenraten von bis zu $1,5MBit/s(187,5KByte/s)$ bereit. In vielen Fällen wird auch das *USB Full-Speed* Protokoll, mit Datenraten von bis zu $12MBit/s(1,5MByte/s)$, angeboten. Bei diesen Geschwindigkeiten handelt es sich jeweils um die maximalen Bruttowerte. Die Nettodatenrate, welche für die Übertragung der Daten zur Verfügung steht, ist von mehreren Faktoren und der jeweiligen Implementierung abhängig.

Eine zwingende Voraussetzung ist, dass beide Seiten der Kommunikation über das gleiche USB Protokoll verfügen. Ist dieses gegeben, nimmt der eingesetzte USB Transfertyp den nächsten großen Einfluss auf die Übertragungsrate. Die Tabelle 2.2 zeigt alle vier vorhandenen Typen. In diesem Fall würden sich aufgrund der Geschwindigkeit nur die *Bulk* und *Isochronous* Typen eignen, da nur mit diesen hohe Taktraten, über größere Datenpakete ($< 64Byte$), erzeugt werden können. Da es sich bei den Übertragungen um Daten handelt, die durch eine Interaktion in einem menschlichem Umfeld entstehen und auf diese auch in diesem Umfeld interagiert wird, handelt es sich um sicherheitskritische Daten, die zwingend fehlerfrei übertragen werden müssen. Da der Typ *Isochronous* keine Zusage über die Korrektheit der Übertragung tätigen kann, wird in diesem Ansatz der *Bulk*-Typ bevorzugt. Dieser bietet sich besonders bei der Übertragung großer Datenmengen an, wobei er aber keine Geschwindigkeit garantieren kann. Es steht immer genau so viel Bandbreite zur Verfügung, wie der Bus zur Zeit an Kapazität frei hat. Dadurch besteht keine garantierte Latenz und Bandbreite, dafür aber eine sichere Datenübertragung. Durch die kleinste Priorität des *Bulk*-Transfers, ist es in diesem System zwingend notwendig alle Teilnehmer auf dem Bus zu kennen und sich dabei über die möglichen Engpässe auf diesem bewusst zu sein. In der Testumgebung wird ein Bus verwendet, auf dem ausschließlich die Erfassungseinheit tätig ist, um das Maximum der Übertragungsrate zu erzielen.

Full-Speed-Geräte mit dem Bulk Transfer, können bis zu $64Byte$ große Pakete übertragen. Dabei ist es nicht notwendig das Paket mit $64Byte$ zu füllen, sondern auch eine kleinere Anzahl an Daten pro Paket, ist dabei möglich, allerdings wird der restliche Platz mit Null aufgefüllt. Der *Bulk*-Transfer verfügt, wie angesprochen, über Fehlererkennung und Handshake, somit ist eine fehlerfreie Übertragung gewährleistet und diese muss nicht gesondert durch z.B. Checksummen gesichert werden.

Des Weiteren hat USB den Vorteil, dass für viele Gerätetypen bereits Treiber auf den bekannten Betriebssystemen vorhanden sind, oder sich durch die Standard Deskriptoren einfach erstellen lassen. Durch diese Deskriptoren wird ein USB Gerät zusätzlich eindeutig identifizierbar und kann weitere Informationen, über das jeweilige Gerät, dem *Host* System mitteilen.

Nachfolgend werden nun mögliche USB Implementierungen beschrieben. Ausschlaggebend ist hierbei nicht nur die Implementierung der Software zur Übertragung der Daten, sondern auch der darunter liegende USB-Stack. Dazu werden verschiedene USB-Stacks miteinander

verglichen. Zuvor folgt eine kurze Beschreibung des Testaufbaus, in dem die verschiedenen Implementierungen auf ihre Verwendbarkeit getestet und evaluiert werden.

Evaluierungsverfahren

Bei diesem Testaufbau geht es nicht wie bei den zuvor getätigten Testsystemen, um den hardwaretechnischen Aufbau und die damit verbundenen Verdrahtung der einzelnen Bauteile. Denn durch die Verwendung eines fertigen Entwicklungsboards und die einfache Verwendung der USB Schnittstelle, ist es ausreichend in diesem Testabschnitt nur ein USB Kabel zwischen der Erfassungseinheit und dem Zielsystems zu verlegen. Der Rest des Testaufbaus bezieht sich ausschließlich auf die Software Komponenten in den beiden Kommunizierenden Geräten. Eines dieser Teilnehmer ist die Erfassungseinheit selbst. Die andere Seite ist ein vergleichbares Linux System, welches sich auch auf dem Roboter *Scitos G5* befindet.

Die Übertragung wird auf Seiten des Linux Systems mithilfe eines *Python* Scripts durchgeführt. Dazu wird aus dem Script, der Mikrocontroller die Erfassungseinheit dazu aufgefordert, die Daten zu senden. Anschließend wartet das Script auf diese Daten, um sie dann zu empfangen. Um die Übertragungsgeschwindigkeit anschließend in der Auswertung miteinander vergleichen zu können, wird in diesem Script, von Beginn der Anfrage bis hin zum letzten empfangenen Paket, die Zeit gemessen.

Bei der Übertragung spielt die Größe und Anzahl der Daten eine entscheidende Rolle. Deshalb werden zunächst große Datenpakete, über einen längeren Zeitraum, übertragen, um ein Gefühl dafür zu bekommen, was maximal möglich ist. Anschließend werden immer 160 bzw. 320 Byte Große Pakete versendet, was der Datengröße der Taxel eines Fingers entspricht. Dieses wird in einer Schleife mehrfach durchgeführt, um sowohl das Maximum und Minimum zu erhalten, als auch den Mittelwert für diese Übertragung.

Da alle zuvor vorgestellten Mikrocontroller über eine USB Schnittstelle verfügen, wird bei jeder das eben beschriebene Verfahren durchgeführt und anschließend deren Ergebnisse ausgewertet. Zuvor werden für allen Drei, jeweils die vorhanden *USB-Stacks* kurz beschrieben.

USB-Stacks der Mikrocontroller

Da das USB Protokoll sehr komplex und vielfältig ist, wäre eine Implementierung von Grund auf eine enorme Softwaretechnische Aufgabe. Aus diesem Grund stellen Hersteller, aber auch diverse freie Projekte, sogenannte *USB-Stacks* zur Verfügung. Diese beinhalten in den meisten Fällen bereits die ganzen *Low-Level* Implementierungen und viele USB Device Klassen. Bei diesen Stacks wird zwischen Hardware und Software Stacks unterschieden. Da letztere in ihren Fähigkeiten und Leistungen beschränkt sind, wird ausschließlich auf Hardware *USB Stacks* zurückgegriffen.

Atmel AVR (Atmega32U4): Für *Atmel AVR* Mikrocontroller stehen die meisten USB Stacks zur Verfügung. Zum einen gibt es vom Hersteller selbst den AVR USB Stack. Der wohl bekannteste freie und *open-source* USB Stack ist *LUFA*[19]. Das Projekt hat es sich zur Aufgabe gemacht die USB Schnittstelle, auch ohne große Kenntnisse der USB Technik, an einem AVR, zu nutzen. Damit lässt sich dieser nicht nur leicht handhaben, sondern in ihm enthalten sind auch sehr viele Beispiele, die den Einstieg erleichtern. Des Weiteren stehen alle bekannten USB Device Klassen zur Verfügung. Aus diesen Gründen wird dieser Stack für die nachfolgende Evaluierung eingesetzt.

Microchip (PIC32MX250): **Microchip** stellt in seinem *Harmony* Framework auch einen USB Stack bereit. Aber auch in diesem Fall wird auf den freien und *open-source* USB Stack M-Stack zurückgegriffen. Dieser ist mit folgenden *Microchip* Mikrocontrollers kompatibel: PIC 16F, 18F, 24F, und 32MX. Auch er verfügt über eine Handvoll Codebeispiele und beinhaltet bereits viele bekannte USB Device Klassen.

Freescale (MK20DX256): Auch *Freescale* stellt einen USB Stack bereit, welcher in seiner Dokumentation sehr beschränkt ist. Aufgrund dieser Tatsache, aber auch durch die guten Ergebnisse die mit dem *PJRC Teensy USB Stack* erzielt wurden (vgl. [57]), wird nicht der vom Herstelle selbst, sondern wieder der freie Stack verwendet. Auch dieser stellt viele Beispiele und bekannte USB Device Klassen zur Verfügung.

Diese drei vorgestellten *USB Stacks* werden nun in ihren jeweiligen Mikrocontroller implementiert und anschließend wie zuvor beschrieben in ihrer Geschwindigkeit evaluiert. Die Ergebnisse werden im nächsten Abschnitt ausgewertet.

Auswertung

Die Erfassung der Zeiten für USB Datenübertragung hat gezeigt, dass sich alle drei Mikrocontroller, trotz deutlich variierendem Systemtakt, nahezu gleich verhalten. Alle erreichen bei einer kontinuierlichen Übertragung von Paketen, sogar fast die maximal mögliche Übertragungsrate (in dargestellt), von $12 Mbit/s$. Das entspricht ca. 6800 Taxel Datensätzen und wäre damit deutlich in den Anforderungen. Allerdings handelt es sich hierbei, wie schon erwähnt, um eine dauerhafte Übertragung von Testdaten, was nicht dem Konzept entspricht.

Eine Übertragung die diesem entspricht, wäre wenn das Endsystem der Erfassungseinheit mitteilt, dass diese ihm den aktuellsten Datensatz schicken soll. Dieses ist in dargestellt und zeigt, dass die Zeiten sich drastisch verschlechtert haben. Das ist darauf zurückzuführen, dass die Zeit die der USB Stack für den Verbindungsaufbau benötigt einen sehr großen Teil einnimmt. Die reine Übertragung, der im Verhältnis geringen Datenmenge von $160 Byte$, geschieht auch in diesem Fall in ähnlichen Zeiten wie bei der dauerhaften Übertragung. Durch

[19] LUFA: **L**ightweight **U**sb **F**ramework for **A**vr

diese Umsetzung würden gerade einmal ca 165 Datensätze pro Sekunde übertragen werden, was sich damit deutlich außerhalb der Anforderungen befindet.

Da es sich bei den Daten um relativ kleine Datenmengen handelt, kann nur eine hohe Datenrate erzielt werden, wenn das USB Device, hier die Erfassungseinheit, seine Daten wie oben schon beschrieben, kontinuierlich schickt. Nur durch diesen konstanten Datenstrom kann die Zeit zum Aufbau der Verbindung vermieden werden. Damit der *USB Host* weiß, wann die Daten in dem kontinuierlichen Datenstrom beginnen, wird ein Start-Byte vor das Datenpaket gesetzt. So kann der *Host* auf dieses Start-Byte warten und anschließend die Daten empfangen. Die Ergebnisse dieser Methode sind in  dargestellt und zeigen im Vergleich zur Übertragung eines Paketes auf Anfrage, zwar eine deutliche Steigerung (ca. 1035 Pakete pro Sekunde), kommen aber an die Zeiten der kontinuierlichen Übertragung noch nicht ran. Damit läge es zwar im Bereich der Anforderungen, aber da es sich hier um Testdaten handelt, die durchgehend verschickt werden, muss man damit rechnen, dass sich die Zeiten im fertigen gesamten System noch etwas verlangsamen.

Bei dem Warten auf das Start Byte, wurde jeweils immer ein Byte gelesen und geprüft, ob es sich um das Start Byte handelt. War dieses der Fall, wurden anschließend alle Daten des Datenpakets empfangen. Dadurch dauerte das Warten auf das Start Byte, in den meisten Fällen, deutlich länger als die Übertragung der Daten selbst. Es wird zwar immer nur das eine Start-Byte gelesen, allerdings wird trotzdem der ganze Rahmen übertragen und der gesamte Stack durchlaufen. So dauert das Empfangen des einen Bytes fast genauso lange wie das eines ganzen Paketes. Aus diesem Grund wurden das Verfahren ein weiteres mal geändert. Um die kleinen Datenpaket zu versenden, werden zwei Datensätze, mit dem vorangestelltem Start-Byte, auf einmal gesendet. Der Empfänger ließt dann anstatt $161Byte$, $332Byte$, womit sichergestellt ist, dass selbst im *worst case* mindestens ein Datenpaket komplett enthalten ist. Damit wird zwar die doppelte Anzahl an Bytes übertragen und verringert dadurch die maximale Datenrate (), um ca. die Hälfte (). Dadurch können bis zu 3400 Datensätze pro Sekunde übertragen werden und ist damit das schnellste Verfahren, dass den Anforderungen entspricht.

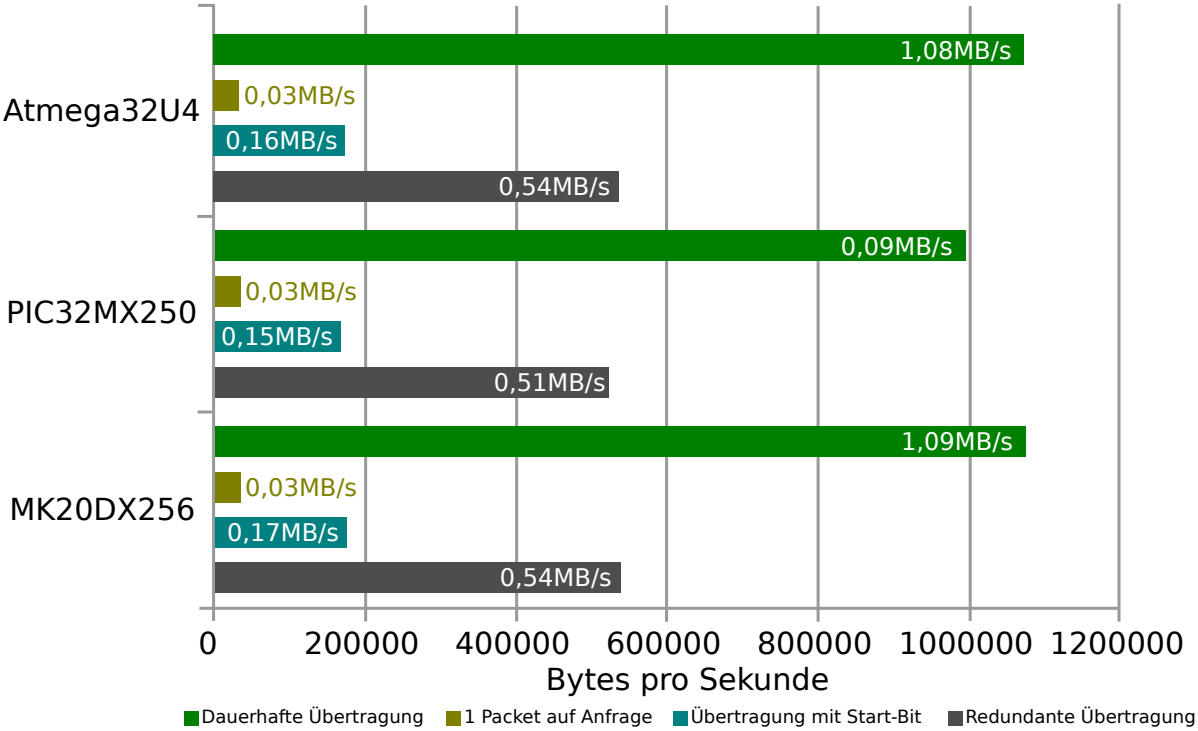


Abbildung 5.24: *USB Benchmark an verschiedenen Mikrocontrollern mit verschiedenen Übertragungsmethoden.*

Die Evaluierung der USB Übertragung hat gezeigt, dass sich alle drei Mikrocontroller, mit den jeweils vorgestellten USB-Stacks, dazu eignen, die Daten in angemessener Geschwindigkeit zu übertragen. Alle konnten eine Datenrate von mindestens $0,51MB/s$ erreichen, was damit den Anforderungen entspricht. Wichtig sei hier noch einmal zu erwähne, dass es sich um Testdaten handelte. In einem späteren Gesamtsystem, können sich die Datenraten dabei anders verhalten. Des Weiteren sei zu beachten, dass die Geschwindigkeit zwar deutlich ausreichend ist, es sich aber um kein Echtzeitfähiges USB Protokoll handelt und somit die Zeiten nie garantiert werden können.

5.4.2 SPI Übertragung

In diesem Abschnitt wird die mögliche Verwendung einer SPI Übertragung evaluiert. Die Schnittstelle wurde bereits ausführlich, sowohl in den Grundlagen 2.2.4, als auch bei der Evaluierung unter 5.2 und 5.3 beschrieben. Aus diesem Grund wird in diesem Teil auf die Funktionsweise und eine ausführliche Beschreibung der Vorteile, nicht groß eingegangen, sondern eben auf diese Abschnitte verwiesen.

Für die Übertragung eignet sich die SPI Schnittstelle besonders durch die hohe mögliche Geschwindigkeit. Außerdem ist eine Übertragung weiterhin in Echtzeit möglich, was bedeutet die Übertragungsgeschwindigkeit kann fest zugesagt werden, da sowohl Taktrate als auch das Protokoll bekannt sind. Zu beachten ist hier, dass die zu überbrückende Übertragungsstrecke für eine SPI Verbindung durchaus zu lang sein kann. Hinzu kommt, dass sich der *Master* und *Slave* der Übertragung, nicht auf derselben Platine befinden. Es muss also zwingend darauf geachtet werden, dass zum einen die Leitungen gegen Störungen genügend abgeschirmt werden und zum anderen eine Masseleitung zwischen den Geräten besteht, um auch über diese Strecke eine fehlerfreie Übertragung zu gewährleisten.

Jeder der drei evaluierten Mikrocontroller aus 5.3, besitzt mindestens eine SPI Schnittstelle. Jedoch wird bereits eine für die ADC Kommunikation verwendet. Da abgesehen von dem *PIC32MX250F128B*, welcher über drei SPI Schnittstellen verfügt, keine zusätzliche Schnittstelle zur Verfügung steht, bezieht sich diese Evaluierung der Übertragung im Detail ausschließlich auf diesen Mikrocontroller. Da es sich hier allerdings nur um eine Evaluierung zur Einschätzung für den Einsatz in ein Sensorsystem handelt, können die Ergebnisse der anderen Zwei, durch die Ergebnisse dieser Evaluierung und der bereits getätigte Evaluierung aus 5.3, theoretisch aufgestellt werden. So steht zwar kein direkt evaluierter Wert zur Verfügung, es kann aber dennoch ein Eindruck geschaffen werden, was möglich wäre. Denn es stehen durchaus auch Mikrocontroller der gleichen Produkt Familie, der anderen Zwei zur Verfügung, die eine weitere SPI Schnittstelle besitzen.

Evaluierungsverfahren

Bei dieser Evaluierung wird ausschließlich die Übertragungsgeschwindigkeit untersucht. Es wird nicht auf den hardwaretechnischen Aufbau und die damit verbundene Verdrahtung eingegangen. So besteht der Testaufbau im grundlegenden aus der Implementierung der Schnittstelle zur Übertragung der Daten von dem *PIC32MX250F128B* zum Zielsystem. Dieses System ist der Master dieser SPI Übertragung und wird wie schon unter 5.2, mithilfe des *FT232H* Chips umgesetzt. Dadurch steht genügend Leistung bereit, um die Evaluierung möglichst wenig durch den *Master* zu beeinflussen.

Durch die gewonnene Erfahrung aus der vorherigen SPI Evaluierung, wird auch in dieser Software Implementierung möglichst direkt an den jeweiligen Registern der SPI Schnittstelle programmiert. Für die Umsetzung stehen wieder mehrere Möglichkeiten zur Übertragung der Daten bereit: *Polling*, *Interrupt* und *DMA*.

Bei der klassischen Implementierung einer SPI-Übertragung (*Polling*), muss die CPU für jedes Byte das gesendet werden soll, warten bis der SPI-Controller bereit ist Daten entgegenzunehmen, um dann das Byte in den Sendepuffer zu schreiben. Dieses würde bedeuten, dass die CPU ständig damit beschäftigt wäre, zu Warten, dass sie die Daten senden kann. Da der

Mikrocontroller noch weitere Aufgaben, wie die Erfassung der Daten hat, eignet sich diese Implementierung nur sehr bedingt und wird in diesem Fall nicht berücksichtigt.

Durch die Verwendung von Interrupts, kann zwar diese Wartezeit genutzt werden, trotzdem muss aber jedes Byte einzeln behandelt werden. Bei hohen Übertragungsraten führt das dazu, dass der *Interrupt Handler* immer mehr Rechenzeit in Anspruch nimmt oder sogar überhaupt nicht mehr nachkommt den SPI-Controller schnell genug mit Daten zu beliefern und damit die Übertragung bremst. Dadurch wird auch das Interrupt Verfahren als nicht geeignet genug eingestuft und findet keine weitere Betrachtung.

Durch die hohe Anzahl an Daten bei maximaler Geschwindigkeit, bietet sich hier besonders der DMA Ansatz an, welcher in diesem Testaufbau zur Evaluierung eingesetzt wird. Dazu werden die Anzahl der zu übertragenden Daten und die Speicheradresse, an der der zu übertragende Datenblock steht, in ein Register geschrieben. Kommt nun die Anfrage des *Masters*, überträgt der DMA-Controller ohne CPU Einfluss das erste Datenwort und zählt die Adresse um einen Schritt nach oben, die Anzahl nach unten und wartet, bis das SPI-Interface wieder bereit ist. Das wiederholt sich so lange, bis das Register bei 0 angekommen ist. Das ganze läuft somit parallel im Hintergrund ab, die CPU kann also weiterhin für anderen Dinge verwendet werden. Sind alle Daten Übertragen worden, kann dieses durch einen Interrupt oder einem *Flag* im Register, der CPU mitgeteilt werden.

Um diese Implementierung der SPI Schnittstelle zu evaluieren, werden 160 Byte große Testpakete, mithilfe eines *Python*-Scripts empfangen und zeitlich ausgewertet. Erwartet werden hier sehr ähnliche Ergebnisse wie bei der Evaluierung unter 5.3.

Auswertung

Die Auswertung der Übertragung 160 Byte großer Pakete von Taxeldaten, hat wie erwartet, gleiche Ergebnisses wie bereits unter 5.3, ergeben. So dauert eine Übertragung $445\mu s$, was einer Geschwindigkeit von $\frac{160Byte}{445\mu s} \approx 0.36MB/s$ entspricht. Da sich diese Zeit nicht von der aus der vorherigen Evaluierung unterscheidet, kann auch angenommen werden, dass auch die anderen zwei Mikrocontroller ähnliche Ergebnisse liefern würden. Damit kann die Geschwindigkeit bei dem *Atmega32U4* von $\frac{160Byte}{1064\mu s} \approx 0.15MB/s$ und bei dem *MK20DX256* von $\frac{160Byte}{226\mu s} \approx 0.71MB/s$, wenn auch nur in der Theorie, angenommen werden. Die Ergebnisse sind in der Abbildung 5.25 dargestellt.

5.4.3 Auswertung Datenübertragung

Die hier evaluierte Datenübertragung bezog sich auf die Übertragung der gesammelten Daten der Erfassungseinheit. Dazu wurden zwei Schnittstellen für diesen Einsatz als geeignet bewertet und anschließend in ihrer Geschwindigkeit und Aufwand ausführlich begutachtet.

Diese Evaluierung hat gezeigt, dass es sowohl mit einer *USB Full-Speed*, als auch mit einer SPI Übertragung möglich ist, die Daten in den geforderten Anforderungen an das System, zu übertragen. Im Fall von USB konnten dabei alle drei Mikrocontroller, mit den jeweils vorgestellten *USB Stacks*, in die Evaluierung mit einbezogen werden. Anders bei der SPI Übertragung, da hier nur der *PIC32MX250F128B* über eine weitere SPI Schnittstelle verfügt. Aus diesem Grund wurde die Analyse für die anderen nicht durchgeführt, aber anhand der Ergebnisse des *PIC32MX250F128B* in diesem Teil und den Ergebnissen aus der Sektion 5.3, für einen möglichen Einsatz, theoretisch angenommen.

Wichtig sei hier noch einmal zu erwähnen, dass es sich hier um Testdaten handelt. In einem späteren Gesamtsystem können sich die Datenraten anders verhalten. Des Weiteren ist bei der USB Verbindung zu beachten, dass die Geschwindigkeit zwar deutlich ausreichend ist, es sich aber um kein echtzeitfähiges Protokoll handelt und somit die Zeiten nie garantiert werden können.

Abschließend stellt die Abbildung 5.25 eine kurze Zusammenfassung der schnellsten Übertragungsraten für beide Schnittstellen, der jeweiligen Mikrocontroller, dar.

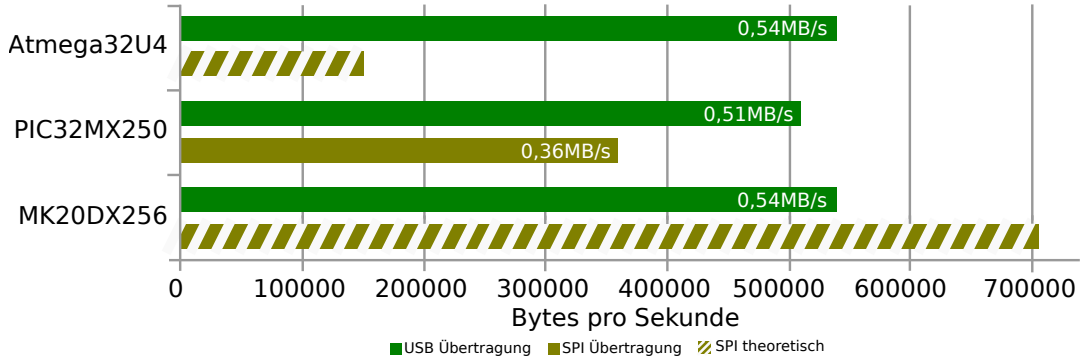


Abbildung 5.25: *Die jeweils schnellsten Übertragungsraten der USB und SPI Schnittstelle der drei Mikrocontroller.*

5.5 Zusammenfassung

Die Erfassungseinheit aus dem Konzept des Gesamtsystems aus Abbildung 4.1 wurde hier ausführlich erarbeitet und evaluiert. Dazu wurde die Einheit wiederum in mehrere Einheiten unterteilt, um so nicht nur den Überblick zu behalten, sondern auch jedes Teilsystem für sich unabhängig zu untersuchen. Dabei konnte gezeigt werden, dass selbst mit einfachsten Mitteln eine schnelle Erfassung von vielen Taxeln möglich ist. So konnten in einigen Fällen bis zu 160 Taxel in nur $0,55ms$ erfasst und übertragen werden. Diese Zahlen liegen deutlich in den Anforderungen und werden im späteren noch dazu verwendet, das komplette System in Kapitel 7 zu evaluieren.

6 Evaluierung der Verarbeitungseinheit

Die Verarbeitungseinheit dient im allgemeinen dazu, Daten zu sammeln und weiter zu verteilen. Damit stellt sie den Datenknoten in dem System dar. Dadurch entstehen zwei entschiedene Vorteile: Das Endsystem muss sich nur um die Kommunikation mit dieser Einheit befassen und zugleich ist es auch nur mit dieser einen Einheit verbunden. Damit ergeben sich eine Vielzahl an qualitativer und quantitativer Möglichkeiten zur Übertragung und erlaubt zusätzlich eine bessere Modularität. Dazu trägt auch bei, dass sich die Einheit nah an den einzelnen Erfassungsmodulen befindet.

Nachfolgend wird nun das Konzept dieser Einheit, zur Datenverarbeitung und Weiterleitung evaluiert.

Abbildung 6.1: *Konzept der Verarbeitungseinheit.*

6.1 Datenverarbeitung und Weiterleitung

Die Abbildung 6.2 zeigt die Verarbeitungseinheit (eine detailliertere Darstellung von Bauteilen, die zur Inbetriebnahme benötigt werden, sind hier nicht berücksichtigt) in einem vereinfachten Systemaufbau. Sie besteht hauptsächlich aus einem Mikrocontroller, welcher sich um die Aufnahme der Daten über den Datenbus, deren Verarbeitung und Weiterleitung kümmert.

Für den dargestellten Datenbus zur Datenaufnahme, bieten sich besonders durch die räumliche Nähe ($< 30cm$) zwischen Sensormodul und der Verarbeitungseinheit, einige einfache und

wenig komplexe Übertragungsverfahren, vergleiche Tabelle 5.4, an. Da es sich hier um die Kommunikation mit der Erfassungseinheit handelt, werden in diesem Teil aber ausschließlich die Schnittstellen in Betracht gezogen, die nach der ausführlichen Evaluierung aus Kapitel 5.4, noch infrage kommen. Dieses entspricht dem gewählten *bottom-up* Ansatz und spart eine weitere ausführliche Evaluierung. So stehen für die Übertragung, von der Erfassungseinheit zur Verarbeitungseinheit, die beiden Schnittstellen SPI und USB zur Verfügung.

Nun gilt es auch für diese Einheit einen Mikrocontroller zu wählen, der die Anforderungen an die Arbeit und das erstellte Konzept gerecht wird. Hier könnte wie auch bei der Erfassungseinheit verwendet, auf einfache Mikrocontroller zurückgegriffen werden. Allerdings würde das zu starken zeitlichen Verzögerungen führen, da die Zeit der gesamten Übertragung dadurch, im Bereich von $200 - 300\mu s$ (vgl. 5.4), verlängert werden würde. So sollte an dieser Stelle ein leistungsstärkerer Controller eingesetzt werden, der u. a. durch seine hohe Taktrate die erwähnte Verzögerung dezimiert.

Da sich die Verarbeitungseinheit nicht mehr direkt an den Fingern des Greifers befindet, sondern nur in deren Nähe, steht im allgemeinen mehr Platz für diese Einheit zur Verfügung. Dennoch sollte sie den Roboter in seiner Bewegung nicht behindern oder einschränken. Dieses würde den Einsatz eines größeren und leistungsstärkeren Controller entgegenkommen. So wären auch Verfahren wie Ethernet oder komplexere und leistungsstarke Bussysteme (vgl. 3.6), für die Weiterleitung möglich. Dieses steht damit aber teilweise im Konflikt zu den Anforderungen an das System. Denn es soll möglichst mit einfachen und wenig komplexen Bauteilen und Verfahren auskommen. Aus diesen Gründen werden diese Ansätze erst einmal vernachlässigt und es wird versucht ausschließlich über eine USB Verbindung die Anforderungen zu erfüllen.

Somit hat die Verarbeitungseinheit die Aufgabe, die Daten entweder über eine USB oder SPI Verbindung zu Erfassen und diese wiederum über eine USB Verbindung weiterzuleiten. Diese beiden möglichen Verfahren werden in den beiden nachfolgenden Abschnitten beschrieben und evaluiert.

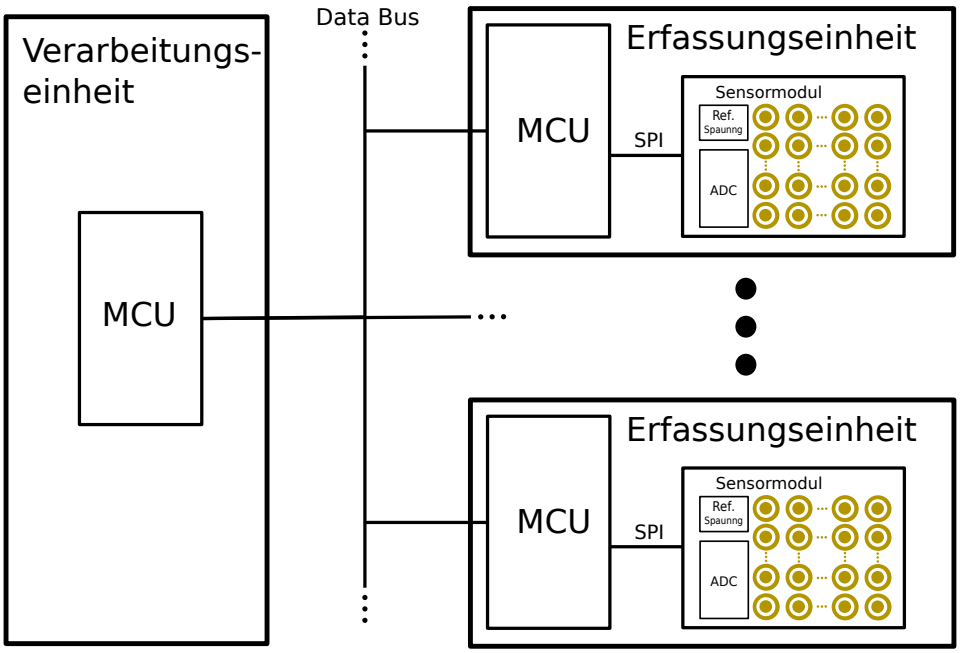


Abbildung 6.2: *Vereinfachter Systemaufbau der Verarbeitungseinheit*

6.1.1 USB Erfassung und USB Weiterleitung

Die Erfassung sowie Weiterleitung der Daten wird über die USB Schnittstelle durchgeführt. Dazu werden nachfolgend zwei Ansätze beschrieben und evaluiert.

Mikrocontroller Ansatz

Bei einer USB zu USB Übertragung, wäre die Verarbeitungseinheit zum einen ein *USB Host*, um mit den einzelnen Erfassungseinheiten (*USB Device*) zu kommunizieren. Zusätzlich wäre sie für das Endsystem aber auch ein *USB Device*. Da es pro USB Controller immer nur einen *USB Host* geben darf, würden in diesem Fall zwei benötigt. Mikrocontroller die zwei USB Controller besitzen, sind allerdings die Ausnahme und von einigen Herstellen gar nicht vorhanden.

Möglich wäre allerdings ein Wechsel zwischen *USB Host* und *USB Device*. Damit würde wieder nur ein Port und somit nur auch nur ein *USB Controller* benötigt werden. Dieser Wechsel würde aber immens viel Zeit in Anspruch nehmen, da bei jedem Wechsel der zwei Modi, die Enumeration der Devices neu durchlaufen werden muss und stellt somit keinen möglichen Einsatz für dieses System dar.

Eine Alternative wäre der Einsatz von zusätzlichen externen Komponenten. So stellt z.B. *FTDI* Chips bereit, die eine Mikrocontroller-USB-Kommunikation ermöglichen ohne das dieser einen

weiteren *USB Controller* benötigt. Diese Chips verwenden dabei allerdings wiederum, die bekannten Schnittstellen, wie z.B. SPI oder I²C, zur Kommunikation zum Mikrocontroller. Damit handelt es sich bei der Übertragung auf Seiten des Endsystems zwar um USB Verbindung, auf der Seite der Verarbeitungseinheit hingehen um eine SPI Schnittstelle. Dadurch würden auch alle Vorteile der USB Übertragung verloren gehen. Dieses entspricht nicht dem Konzept und findet an dieser Stelle auch keine weitere Beachtung.

Diese kurze Betrachtung hat ergeben, dass es nicht ohne großen Aufwand möglich ist, unter Berücksichtigung der Anforderungen, das Konzept mithilfe eines Mikrocontrollers zu erfüllen.

USB-Hub Ansatz

Diesen Ansatz der USB zu USB Übertragung, ist im allgemeinen nur ein einfaches Durchreichen bzw. Weiterleiten der USB Daten. Dazu kommt anstatt eines Mikrocontrollers, ein USB-Hub zum Einsatz. Damit fällt zwar die Möglichkeit weg, die Daten auf der Einheit zu verwalten, dafür singt der Implementierungsaufwand, Software- sowie Hardwareseitig.

Jedes USB Device, welches sich an dem Hub befindet, wird am Endsystem als eigenständiges Device angezeigt und bleibt damit eindeutig identifizierbar. Auch eine eindeutige ID für den Hub ist durch den USB-Hub-Deskriptor möglich. Denn wie jedes andere USB Gerät, ist auch der Hub ein *USB Device* und besitzt u. a. auch eine *Vendor ID*, *Product ID*, und *Device ID*, welche sich in den meisten Fällen editieren lassen. So ist dem Endsystem auch bekannt, welche Erfassungseinheiten hinter der Verarbeitungseinheit hängen und das Konzept bleibt damit weiter, wie definiert erhalten.

USB-Hubs stehen auch hier wieder von vielen Herstellern zu Verfügung. Bei den Eigenschaften wird im allgemeinen zwischen der Anzahl der USB Ports und der Geschwindigkeit unterschieden. Gängige Werte sind hier vier Ports und *USB Hi-Speed* $(480 Mbit/s)$. Dieses erfüllt auch der *USB2504-JT* USB-HUB von *Microchip*. Durch seine *TQFP-64* Bauform lässt sich dieser wieder mit einfachen Mitteln verarbeiten und auch der elektronische Aufwand hält sich in einem angemessenen Rahmen (vgl. 6.3).

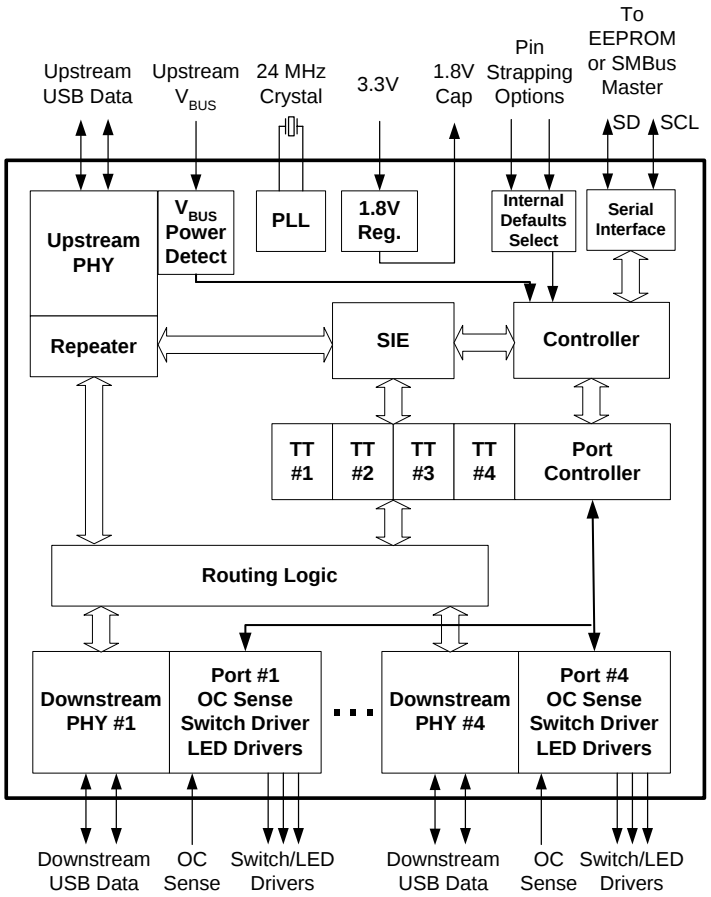


Abbildung 6.3: *4-Port Block Diagramm des USB-Hubs USB2504.*

Dieser Hub besitzt bereits einen $24MHz$ *Crystal*, mit dem die hohe Taktrate für *USB-2.0* generiert wird. Zusätzlich lässt er sich z.B. über die I²C Schnittstelle konfigurieren, um u. a. die USB Deskriptoren anzupassen. Um die Konfiguration dauerhaft zu halten, kann diese auch in den EEPROM geschrieben werden. Das würde bedeuten, dass jedem Hub im Sensorsystem nur einmal Initial seine ID mitgeteilt werden muss.

Wie beschrieben, handelt es sich hierbei um eine 4-Port USB-Hub, was die maximale Anzahl an Geräten von 127 stark mindern würde, allerdings können die Hubs beliebig oft verschachtelt werden. Wichtig ist die Berücksichtigung, dass auch jeder Hub ein Device ist und sich somit die zahl der maximal möglichen Erfassungseinheiten verringert.

Eine der entscheiden Faktoren dieser Evaluierung, hat bisher noch keine Betrachtung gefunden. Durch den Einsatz eines *USB2.0* Hubs sind Datenraten von bis zu $480Mbit/s$ möglich. Da die Erfassungseinheiten allerdings nur mit *Full Speed* arbeiten, kann Protokoll bedingt auch nur mit $12Mbit/s$ übertragen werden. Es ist aber möglich, mehrere Full-Speed-Geräte parallel zu betreiben. Denn ein USB-2.0-Endsystem und ein USB-2.0-Hub kommunizieren immer mit

Hi-Speed, selbst wenn an dem Hub Full-Speed-Geräte angeschlossen sind. Der Hub hat dabei die Aufgabe, die Daten dieser Geräte in das Hi-Speed-Protokoll zu verpacken. Erst wenn die Anzahl der Geräte überschritten wird, bricht die Datenrate aller an diesen Host angeschlossenen Full-Speed-Geräte auf Geschwindigkeiten deutlich unter denen eine USB-1.1-Verbindung ein; der Durchsatz von Hi-Speed-Geräten am selben Hub bleibt jedoch unbeeinflusst.

Denkbar wäre hier natürlich auch der Einsatz fertiger USB-Hubs, die es in jedem Elektrofachmarkt zu kaufen gibt. Hier wäre allerdings keine eindeutige Identifizierung der Verarbeitungseinheit möglich.

Da es sich hierbei um das reine weiterleiten der Taxeldaten handelt, wird keine weitere Bestimmung der maximalen Datenraten durchgeführt. Es wird dabei auf die Ergebnis aus Kapitel 5.4.1 verwiesen und angenommen, das diese weiterhin bestand haben.

Auswertung

Die Evaluierung hat ergeben, dass eine Verarbeitungseinheit, die dem Konzept entspricht, nicht die beste Lösung ist und nur mit deutlichem Mehraufwand erreicht werden kann. So wurde von einem Einsatz eines Mikrocontrollers abgesehen und stattdessen ein USB-Hub verwendet. Dieser liefert alle nötigen Anforderungen und ist in Sachen, besonders bezogen auf den Implementationsaufwand der Software, deutlich einfacher.

In Sachen Geschwindigkeit treten keine Verringerungen im Vergleich zu den Testergebnissen aus 5.4.1 auf. Es wird sogar durch den Einsatz eines USB-2.0-Hubs die Möglichkeit geschaffen, mehrere Taxeldaten der Erfassungseinheiten gleichzeitig zu übertragen.

Etwas mehr Aufwand wird durch den Einsatzes des USB-Hubs, auf der der Seite des Endsystems geschaffen. So muss es sich nicht, wie im Konzept geplant, nur um die Verwaltung der Verarbeitungseinheiten kümmern, sonder auch alle Erfassungseinheiten müssen von ihm verwaltete werden, da der Hub diesbezüglich keine Logik besitzt.

6.1.2 SPI Erfassung und USB Weiterleitung

Die Erfassung sowie Weiterleitung der Daten wird über die SPI Schnittstelle durchgeführt. Dazu werden nachfolgend auch hier zwei Ansätze beschrieben und evaluiert.

Mikrocontroller Ansatz

Bei der USB Erfassung und USB Weiterleitung, wird ein Mikrocontroller eingesetzt, der die Daten über die SPI Schnittstelle erfasst und über die USB Schnittstelle weiterleitet. Für die Zeit der SPI Übertragung der Erfassten Taxeldaten, von dem Sensormodul bis zur Verarbeitungs-

einheit, werden bereits mindestens fast $500\mu s$ benötigt, somit bleiben für den Weitertransport an das Endsystem nur noch ca $600\mu s$, um den Anforderungen gerecht zu werden. Diese Zeitspanne ist in der Theorie, bei einer USB-Full-Speed Verbindung, ausreichend. Da der Aufbau einer USB Übertragung aber viel Zeit in Anspruch nimmt (vgl. 5.4.1), wird das vorhandene Zeitfenster sehr eng. Um dieses zeitlich zu bewältigen, wird ein Mikrocontroller mit einer High-Speed-USB Schnittstelle benötigt. Dabei dauert der Aufbau zwar ähnlich lange, aber durch die deutlich schnellere Übertragungsrate von bis zu $480MBitz/s$, kann die Zeit an dieser Stelle evtl. wieder gut gemacht werden. Zwar stehen Controller mit dieser Schnittstelle zur Verfügung, welche sich auch im angemessen Rahmen in dem System integrieren lassen, dennoch wird an dieser Stelle auf diesen Einsatz verzichtet. So wird auch wie schon unter 6.1.1, der Ansatz über eine direkte Weiterleitung vorgezogen. Dieser ist sowohl bezogen auf den Implementierungsaufwand als auch auf elektronischer Ebene, deutlich effizienter, schneller und einfacher und erfüllt damit deutlich mehr den Anforderungen an das gesamte System.

SPI zu USB Konverter Ansatz

Bei diesem Ansatz der SPI Erfassung und USB Weiterleitung, wird statt eines Mikrocontroller ein *Single Channel Hi-Speed USB to Multipurpose IC* eingesetzt. Dieses konvertiert u. a. die SPI Übertragung in eine Hi-Speed-USB Verbindung. Auf Seiten der Erfassungseinheit handelt es sich dabei um ein reines SPI Protokoll. Auf dem Zielsystem hingehen steht ein USB Device zur Verfügung. Dieses Bauteil kümmert sich somit um die gesamte Umwandlung des Protokolls, bzw. um die Kapselung des SPI Protokolls in das USB Protokoll.

Ein möglicher Chip ist der *FT232H* der Firma *FTDI*, der bereits in 5.2 und 5.4.2 zum Einsatz kam. Mit diesem lassen sich alle SPI Transferraten der Erfassungseinheit übertragen, ohne das zusätzlich Zeit beansprucht wird. Durch diverse I/O-Pins lassen sich auch eine Vielzahl an SPI Geräte ansprechen, welche durch den Einsatz von *Demultiplexern* noch weiter erhöht werden könnte. Um die Identifizierung der Verarbeitungseinheit zu gewährleisten, lassen sich auch hier die USB Deskriptoren, persistent in einem EEPROM verändern.

Mit dem Entwicklungsboard *UM232H* existiert bereits ein sehr kompaktes System, welches sowohl in einem Prototypen, als auch in einem fertigen Sensorsystem, integriert werden könnte. Aber auch eine Entwicklung eines eigenen Boards einem *FT232H*, stellt durch die einfache Handhabung keinen großen Aufwand dar.

Da es sich hierbei um das reine weiterleiten der Taxeldaten handelt, die bereits unter 5.4.2 evaluiert wurde, wird hier auf diese Ergebnisse verwiesen.

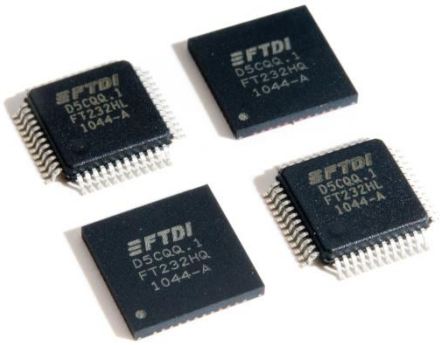


Abbildung 6.4: *FT232H SINGLE CHANNEL HI-SPEED USB TO MULTIPURPOSE UART/FIFO IC der Firma FTDI.*

6.1.3 Auswertung

Die Evaluierung hat ergeben, dass eine Verarbeitungseinheit, die dem Konzept entspricht, nicht die beste Lösung ist. So wurde von einem Einsatz eines Mikrocontrollers abgesehen und stattdessen ein SPI-USB-Konverter verwendet. Dieser entspricht allen Anforderungen und ist besonders bezogen auf den Implementationsaufwand der Software, deutlich einfacher umzusetzen.

In Sachen Geschwindigkeit treten keine Verringerungen, im Vergleich zu den Testergebnissen aus 5.4.2, auf und es kommt zu keiner Einschränkungen, z.B. bezogen auf die Anzahl der maximalen Erfassungseinheiten im System.

Etwas mehr Aufwand wird, durch den Einsatzes dieses Konverters, auf der Seite des Endsystems geschaffen. Die Software muss sich nicht, wie im Konzept geplant nur um die Verwaltung der Verarbeitungseinheiten kümmern, sonder auch alle Erfassungseinheiten müssen von ihm verwaltete werden.

6.2 Zusammenfassung

Die Verarbeitungseinheit aus dem Konzept des Gesamtsystems, abgebildet in 4.1(b), wurde hier ausführlich erarbeitet und evaluiert. Dabei ist das Konzept aus Gründen des Implementierungsaufwands und Vorteilen bei der Übertragungsgeschwindigkeit, verändert worden. So wurde auf den Einsatz eines Mikrocontrollers in beiden Varianten der Übertragung (SPI und USB), verzichtet. Dabei wurden aber alle Anforderungen das gesamte Sensorsystem eingehalten.

Es konnte mit einfachen Mitteln und geringen Aufwand eine Erfassung und Weiterleitung, der Daten der Erfassungseinheiten, erreicht werden. Diese Daten können dabei mit bis zu $0,71 MByte/s$ an das Endsystem übertragen werden.

7 Evaluierung des Sensorsystem

Abschließend der Evaluierung, wird in diesem Kapitel das entwickelte Konzept als ganzes betrachtet. Dazu wird das in Abbildung 4.1(b) dargestellte Konzept, mithilfe der zuvor getätigten Evaluierung der einzelnen Teilkomponenten, detailliert beschrieben und anschließend evaluiert. Mit diesem detaillierteren Konzept das gesamten Systems, ist es dann möglich zu prüfen, ob es alle gesetzten Anforderungen gerecht wird. Als Kernpunkt seht hier das Zusammenspiel der einzelnen Komponenten, bezogen auf die Geschwindigkeit im Fokus. Dazu wird in dem Abschnitt 7.2 eine ausführliche Validierung durchgeführt. Abschließend werden alle Ergebnisse Zusammengefasst.

7.1 Detailliertes Konzept des Sensorsystems

Das Grundkonzept aus der Abbildung 4.1(b) stellt eine sehr abstrakte Form des Konzeptes dar. Diese wird nun, mithilfe der zuvor getätigten Evaluierung der Erfassungs- und Verarbeitungseinheit, detaillierter beschrieben und stellt ein konkretes Konzept auf.

In Abbildung 7.1 ist das komplette zu evaluierende System abgebildet, welches in mehreren, teilweise sogar unabhängigen, Teilkomponenten aufgebaut ist, die sich zugleich auf verschiedenen Ebenen einer hierarchischen Struktur befinden. Im Vergleich zu den vorherigen Abbildungen der einzelnen Komponenten, sind in diesem bereits konkrete Entscheidungen getroffen worden. So sind z.B. die Übertragungsverfahren bereits fest definiert. Diese Entscheidungen in der Konzeption, basieren auf den Ergebnissen der vorherigen zwei Kapitel. Dabei wurde eine mögliche Umsetzung gewählt, die eine hohe Erwartung hat, die geforderten Anfordern zu erreichen.

Auf der obersten Ebene befindet sich das Endsystem, an denen sich eine oder mehrere Sensoreinheiten befinden können. Diese Sensoreinheiten liefern über eine USB Schnittstelle die Sensordaten an das Endsystem. Durch die sieben Bit interne Adressierung der angeschlossenen Geräte im USB-Controller, würde sich die Möglichkeit ergeben, 127 Geräte pro USB-Controller, anzuschließen. Durch die Verwendung von *USB 2.0*, sind dabei Datenraten von bis zu $480 Mbit/s$ möglich, welche damit eine ausreichende Geschwindigkeit, für die Übertragung der Daten, zur Verfügung stellt.

Die in der Sensoreinheit enthaltene Verarbeitungseinheit, besteht in diesem Fall nur aus einem USB 2.0 Hub. Dieser leitet die Daten der einzelnen Erfassungseinheiten an den zuvor erwähnten USB Controller des Endsystems weiter. Da es sich hier um einen USB 2.0 Hub handelt, könnte dieser auch ohne weiteres die Daten mehreren Erfassungseinheiten parallel übertragen, wodurch sich die eben erwähnte hohe Datenrate von bis zu $480 Mbit/s$ ergibt. Allerdings wird das in der nachfolgenden Validierung nicht mit berücksichtigt, zeigt aber dass in Sachen Geschwindigkeit noch deutlich Platz für Verbesserungen ist. Jede Einheit ist dabei eindeutig durch eine ID zu identifizieren und zusätzlich ist dem Endsystem auch bekannt welche Erfassungseinheiten sich an diesem Hub befinden.

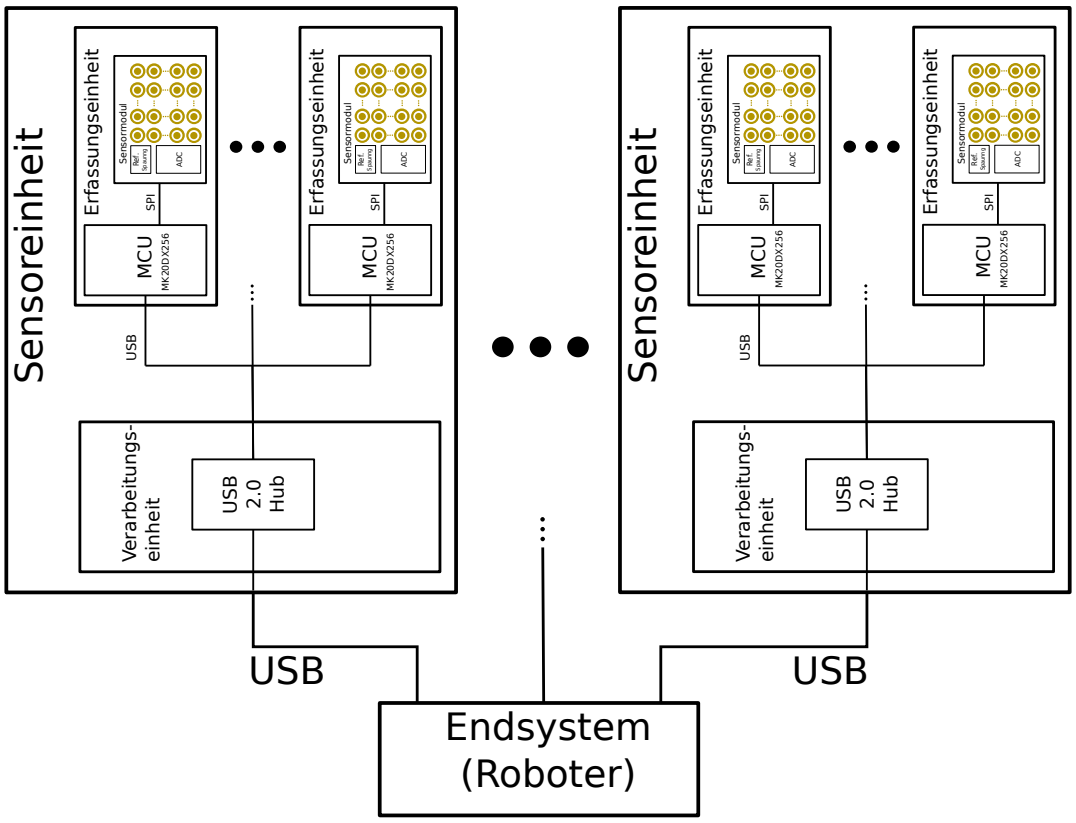


Abbildung 7.1: *Detailliertes Konzept des gesamten taktilen Sensorsystems.*

In dem System der Erfassungseinheit befindet sich als Prozessoreinheit der Mikrocontroller *MK20DX256*. Dieser ist ein USB Bulk Device mit einer eindeutigen identifizierbaren ID. Über die SPI Schnittstelle werden die 160 Taxel, die sich in einer Matrix-Schaltung befinden, wie unter 5.2.2 beschrieben erfasst. Dazu kommt neben dem externen 16 Port ADCs (*AD7490*) noch eine Referenz Spannung (*AD780*) zum Einsatz, die von dem ADC, zur Umwandlung der analogen Werte, benötigt wird.

7.2 Validierung der Geschwindigkeit

In diesem Abschnitt wird das zuvor erstellte gesamte Konzept, zur Erfassung und schnellen Übertragung der Drucksensordaten an ein Zielsystem, validiert. Dazu wird geprüft, ob das Konzept den Anforderungen entspricht und das System mit diesem umsetzbar ist.

Eines der Hauptkriterien dieser Arbeit ist es, eine schnelle Abtastrate der Sensoren zu ermöglichen, wobei besonders zwei Faktoren eine große Rollen spielten: Die benötigte Zeit zur **Datenerfassung** und **Datenübertragung**. Dazu wurde in den beiden vorherigen Kapiteln, die einzelnen Teilkomponenten, auch bezogen auf die Geschwindigkeit, ausführlich evaluiert. Die Ergebnisse werden nun in einen Zusammenhang gestellt und sind in Abbildung 7.3 in einem Zeitdiagramm dargestellt. Dabei wurden allerdings nicht alle möglichen Kombinationen berücksichtigt, die durch die verschiedensten Evakuierungen, aus den beiden vorherigen Kapiteln, möglich gewesen wären. Es gilt hierbei um die Validierung des gesamten Konzepts aus 7.1. Dabei handelt es sich bei der Validierung der Zeit, wie in den Anforderungen erwartet, um zwei Module bzw Finger. Dieses Konzept wird nachfolgend mithilfe des Diagramms 7.2 ausführlich, bezogen auf die Geschwindigkeit, beschrieben und validiert.

Beginnend bei der Datenerfassung spielt der A/D-Wandler die ausschlaggebendste Rolle. Ist dieser nicht schnell genug, werden alle weiteren Schritte verlangsamt. Bei der Verwendung eines A/D-Wandler mit $1MSPS$, benötigt eine Digitalisierung damit $1ms$. Dabei handelt es sich allerdings um den maximalen Wert, der nur unter besten Bedingungen erreicht wird. In diese, Fall würde die Erfassung aller 16 Ports eines ADCs $16 * 1\mu s = 16\mu s$ dauern. Das hier zu validierende Sensormodul, muss diese Wandlung allerdings 10 mal durchführen, damit alle 160 Taxel erfasst werden. Daraus ergibt sich die Zeit von $16\mu s * 10 = 160\mu s$ für die gesamte Erfassung aller Taxel eines Moduls. Dieser theoretische Wert wurde durch die Evaluierung der Erfassungseinheit unter 5.3.5 zwar nicht erreicht, aber es konnte mit $226\mu s$, für die gesamte Erfassung der 160 Taxel, ein sehr guter Wert erreicht werden. Um ein wenig Toleranz bei der Berechnung zu erhalten, wird von einer Zeit von $250\mu s$ für die Datenerfassung der 160 Taxel ausgegangen ( ).

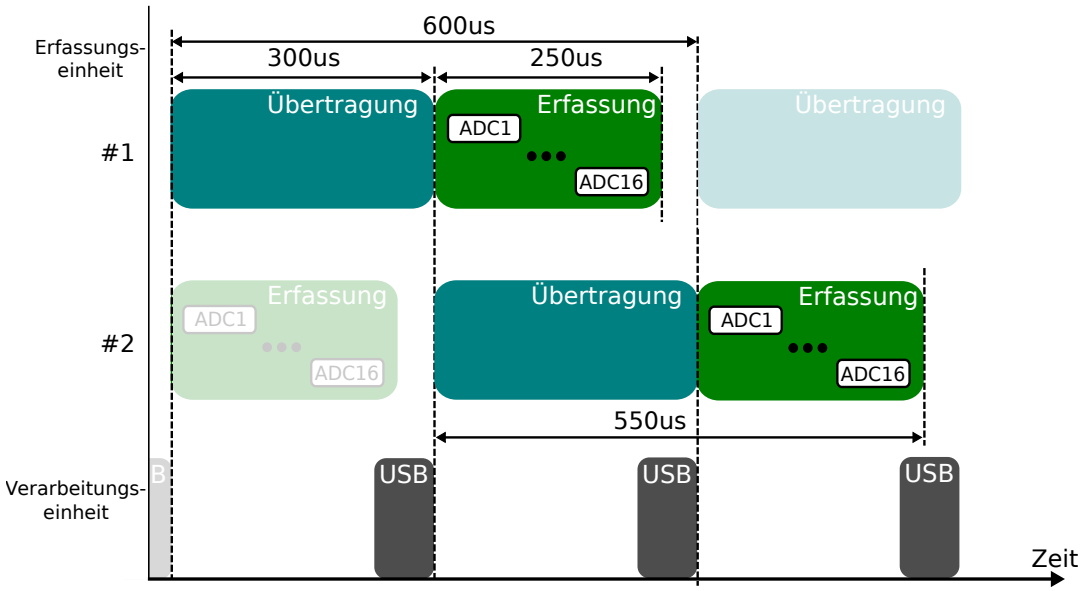


Abbildung 7.2: *Zeitdiagramm zur Veranschaulichung der parallelen Datenerfassung und Datenübertragung zweier Sensor Module.*

Die anschließende Übertragung der eben erfassten Daten an das Zielsystem (Roboter), besteht konzeptionell aus zwei Teilen. Zuerst die Übertragung an die Verarbeitungseinheit und darauf folgend an das Zielsystem. Dieses Konzept wurde allerdings durch die Verwendung der USB Schnittstelle, außer Kraft gesetzt. So gehen die Daten zwar weiterhin über die Verarbeitungseinheit und werden dort gebündelt, aber ohne das sie von einem Mikrocontroller erfasst werden. So wird für die Berechnung der gesamten Zeit, die Zeit für die Übertragung von der Erfassungseinheit bis hin zum Zielsystem verwendet, welche in ■ dargestellt ist. Die in ■ abgebildete Zeitspanne, soll in der Darstellung zeigen, das die USB Übertragung der Verarbeitungseinheit in der Zeit der gesamten Übertragung enthalten ist.

Um die Übertragung mit möglichst wenig Daten zu belasten, werden die $16bit$ Datenwerte verkleinert. Zum einen wird die darin enthaltene $4bit$ Adresse des Ports weggelassen, da diese Identifizierung auch durch das Protokoll, bzw. im Treiber erledigt werden kann und somit einige Datenbits bei der Übertragung einspart. Zum andern wird der $12bit$ Datenwert auf $8bit$ reduziert. Da diese Größe, anhand der Evaluierung unter 5.1, als ausreichend anzusehen ist. Damit fallen für die digitalisierten Taxel $160Taxel * 8bit = 1280bit$ bzw. $160Byte$ große Datenmengen an. Diese müssen nun über die $12Mbit$ schnelle USB Verbindung verschickt werden und benötigen in der Theorie die folgende Zeit:

$$Zeit_{USB} = \frac{Daten}{Geschwindigkeit_{USB}} = \frac{1280bit}{12Mbit/s} = 0.0001067s = 106,7\mu s \qquad (7.1)$$

Auch dieser Wert konnte natürlich nicht ganz erreicht werden. Durch die Evaluierung in 5.4.1 konnte eine Zeit für die gesamte Übertragung der $160Byte$ großen Sensordaten von $294\mu s$ erreicht werden. Auch hier wird der Wert wieder aufgerundet und wird nachfolgend mit $300\mu s$ bestimmt. Daraus folgt die Dauer der Übertragung eines kompletten Satz an Daten, pro Modul in $550\mu s$.

Da die Zeit der Übertragung länger dauert als die Erfassung der Daten, kann ein Modul während es seine Daten überträgt, das Andere die Daten schon erfassen und somit sofort verschicken, wenn es den Zeitslot für die Übertragung erhält. Dadurch spielt für die endgültige Erfassungsrate des gesamten System, ab dem Einsatz von zwei Erfassungseinheiten, nur die Übertragung der Daten eine entscheidende Rolle. Bei einem Modul dauert eine Übertragung $550\mu s$, was einer Taktrate von über $1800Hz$ entspricht. Bei zwei Modulen liegt die Zeit bei $600\mu s$ und einer Erfassungsrate von noch immer über $1600Hz$. Jedes weitere Modul würde die Dauer der Übertragung um $300\mu s$ Verlängern: $N * (300\mu s)$.

Der Verlauf der Erfassungsrate bei unterschiedlicher Anzahl an Modulen, mit jeweils 160 Taxel, ist in Abbildung 7.3 dargestellt. Dieser ist deutlich zu entnehmen, dass bei dem Einsatz von zwei Modulen, die geforderten $800Hz$ deutlich erreicht werden. Selbst bei über zehn Modulen ist diese noch über $200Hz$, was auch im Vergleich zu vielen anderen Arbeiten und Systemen als gutes Ergebnis zu behandeln ist.

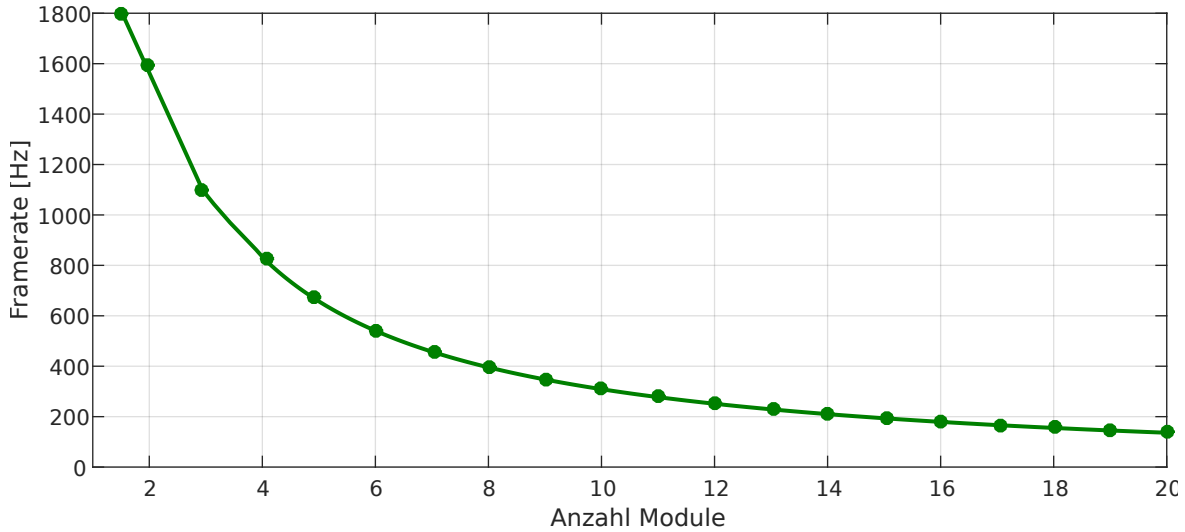


Abbildung 7.3: *Verlauf der Erfassungsrate bei Erhöhung der Anzahl der Module (160 Taxel pro Modul).*

7.3 Auswertung

Das entwickelte Konzept des Sensorsystems hat in 7.2 gezeigt, dass es in Sachen Geschwindigkeit deutlich in den Anforderungen liegt. Des Weiteren lässt es sich mit einfachen Mitteln herstellen und erfüllt somit auch diese Anforderung. Zwar ist ein maschinell erstelltes PCB notwendig, nicht nur um die aufgedruckten Elektroden zu verwirklichen, sondern auch um die Größe in einem angemessenen Rahmen zu halten, dieses lässt sich aber aufgrund der nur zwei benötigten *Layer* leicht erstellen. Des Weiteren lassen sich alle benötigten Bauteile, die dieses System benötigt, per Hand integrieren und löten. Auch das Aufteilung in mehrere Module, macht die Herstellung deutlich weniger komplex und aufwendig.

Außerdem wurden ausschließlich Komponenten benutzt die zum Standard gehören und einfach zu beschaffen sind. Zusätzlich ist die Anzahl der Bauteile sehr gering gehalten. So wird sicher gestellt, dass auch bei einem späteren Defekt oder Nachbau von Modulen, die Bauteile weiterhin beschaffen werden können. Durch die ausschließliche Verwendung von Standard Interfaces, wird dieses verstärkt. Zusätzlich stehen eine Vielzahl an kostengünstigen Alternativen zur Verfügung. Somit ist eine weitere Anforderung erfüllt.

Das ganze System ist so aufgebaut, dass es Problemlos skaliert werden kann. Eine Verarbeitungseinheit kann nahezu um beliebig viele Sensor Module erweitert werden. Hier ist allerdings zu beachten, dass bei 160 Taxel Modulen, die Erfassung bei jedem zusätzlichem Modul um $(300\mu s)$ steigt. Es besteht auch die Möglichkeit, weitere Gesamtsysteme per USB anzuschließen. Auch die Anzahl der Taxel eines Moduls kann, ohne großen Einfluss auf das gesamte System, verändert werden. Durch den Modularen Aufbau fällt bei einer Konfigurationsänderungen keine Änderungen bei den Sensoren/Aktoren an, da jedes Modul für sich selbst arbeitet und zuständig ist. Somit entsteht ein sehr flexibles System. Des Weiteren ist die Intelligenz bei jedem Teil des Systems gering, da dieses auf mehrere Ebenen verteilt ist.

7.4 Zusammenfassung

In diesem und den beiden vorherigen Kapiteln wurde das Konzept aus Abbildung 4.1(b) ausführlich beschrieben und evaluiert. Dazu wurden sowohl die einzelnen Teilkomponenten, als auch das Konzept als ganzes, betrachtet. Die Abschließende Validierung und Auswertung hat gezeigt, dass das taktile Sensorsystem alle definierten Anforderungen erfüllt und in einigen diese sogar deutlich übersteigt. In dem nachfolgendem Kapitel wird das System des Sensors zum rutschsicheren Greifen, nach diesem Konzept, umgesetzt.

8 Das taktile Sensorsystem

In diesem Kapitel wird das zuvor erstellte Konzept, für den Assistenzroboter *SCITOS G5* und seinen Zwei-Finger-Greifer, umgesetzt. Dabei wird sich auf die Ergebnisse der ausführlichen Evaluierung der einzelnen Komponenten gestützt, welche großen Einfluss auf das hier umgesetzte Sensorsystem, haben. Nachdem die einzelnen Komponenten des Systems entwickelt wurden, wird eine Software zur Validierung auf dem Zielsystem erstellt. Mit dieser können die Taxeldaten dann erfasst und ausgewertet werden.

Das Sensorsystem teilt sich, wie im Konzept (siehe Abbildung 7.1) dargestellt, in drei wesentliche Teile. Dabei steckt, bezogen auf den Aufwand der Entwicklung, der größere Teil in der Erfassungseinheit. Die Verarbeitungseinheit dient in diesem Ansatz eher als eine lokale Verteilereinheit, die die Daten der einzelnen Erfassungseinheiten an die entfernte Zentrale weiterleitet. Auf den Roboter selbst wird, bis auf die Erstellung einer Software zum testen des entwickelten Systems, nicht weiter eingegangen. Nachfolgend werden diese drei erwähnten Komponenten detaillierter beschrieben

8.1 Erfassungseinheit

Die Erfassungseinheit besteht aus mehreren einzelnen Teilkomponenten, welche von unten beginnend kurz beschrieben werden:

Auf unterster Ebene befindet sich der einzelne Taxel, über den, mithilfe einer Spannungsteiler-Schaltung, die Spannung gemessen wird, die sich durch das Ausüben von Druckkraft ändert (vgl. 5.1.1). Diese Spannungsänderung kommt durch die Elektroden des Taxels und dem darauf liegenden, leitfähigen Polymer zustande (vgl. 5.1.1). Von diesen Taxeln befinden sich 160 Stück in einer $10 * 16$ Matrix-Anordnung (siehe Abbildung 5.8) in der Erfassungseinheit. Diese Sensoren werden über einen externen ADC, von einem Mikrocontroller erfasst, der die Daten wiederum direkt weiter an die Verarbeitungseinheit schickt.

Nachfolgend wird die Umsetzung, zuerst auf elektronischer Ebene und anschließend die Software zur Steuerung der gesamten Erfassungseinheit, ausführlich beschrieben. Durch die getrennte Betrachtung, können einige Designentscheidungen der Hardware erst im Teil der Software nachvollziehbar erscheinen.

8.1.1 Aufbau der Schaltung

Die Abbildung 8.1 zeigt einen vereinfachten Aufbau der Schaltung der gesamten Erfassungseinheit. Hier wurden, zur besseren Veranschaulichung viele elektronische Bauteile weggelassen und einige teilweise abstrakter und vereinfacht dargestellt. Dieser Aufbau wird nun nachfolgend, in seinen Teilen, beschrieben.

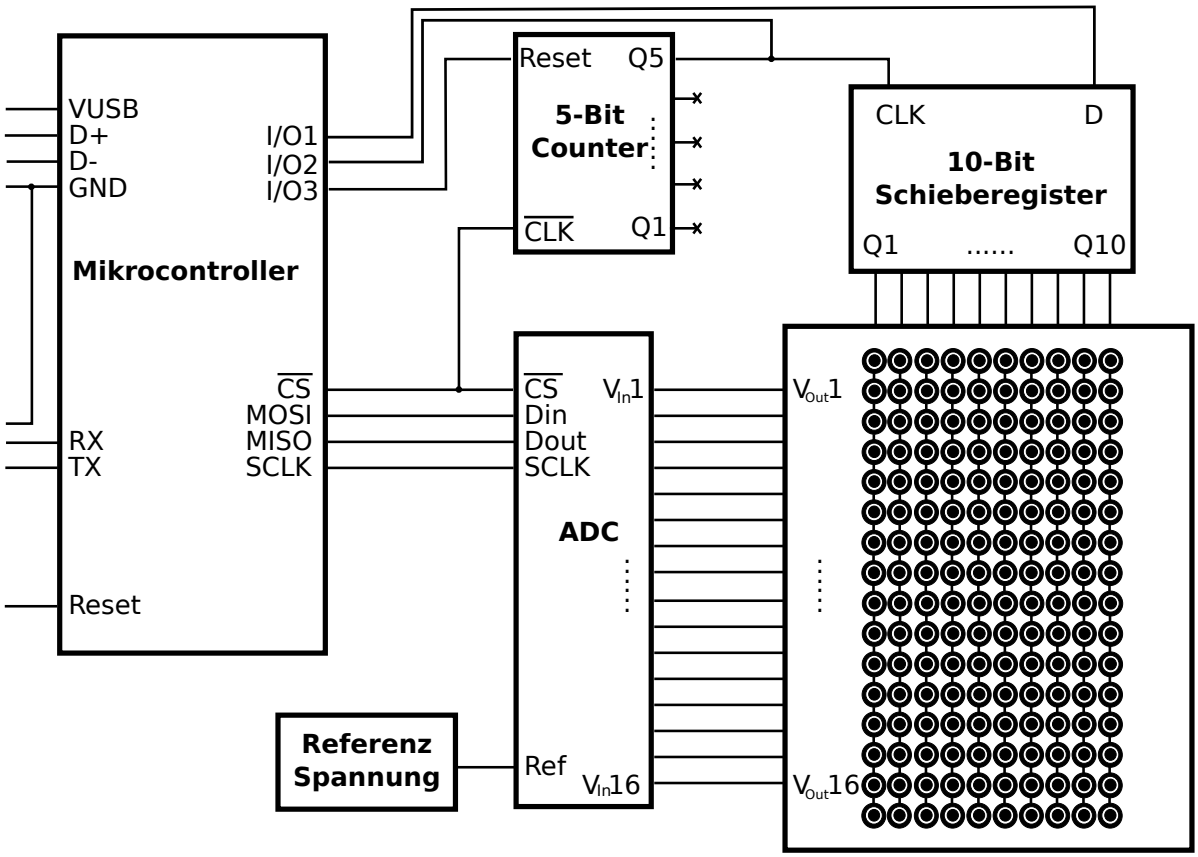


Abbildung 8.1: *Abstrakter Aufbau der Schaltung der Erfassungseinheit.*

Sensormodul

Das Sensormodul besteht aus 160 Taxeln die in einer $10 * 16$ Matrix-Anordnung aufgebaut sind. Als Elektrode wird die Kreisform aus 5.3(c) gewählt. Diese hat einen Durchmesser von $3,5mm$ und einen Abstand zum benachbarten Taxel von $0,5mm$. Damit hat das Sensormodul die Maße $64mm * 35mm$ und liegt unter der maximalen Fingerlänge des Greifers (vgl. 2.4.1). Als leitfähiges Material wird das unter 5.1.1 evaluierte, piezoelektrische Polymer verwendet, welches im Zusammenspiel mit den Taxeln, nicht nur in Sachen Präzision, sondern auch in der Unterscheidung mehrere Kräfte, sehr gute Ergebnisse erzielen konnte.

ADC

Zur Erfassung der Taxel, wird der externe 16-Port ADC *AD7490* (Abb. 5.9(a)) verwendet und mit diesem die Daten, wie in 5.2.2 beschrieben, erfasst. An jedem Port des ADCs ist eine der 16 Reihen der Sensor-Matrix angeschlossen. Über die 10 Spalten werden die einzelnen Taxel nacheinander mit $5V$ Spannung versorgt, so können alle 160 Taxel der Matrix mit diesem einen ADC Baustein erfasst werden. Für die Umwandlung der analogen Spannungswerte, benötigt der Wandler eine $2,5V$ Referenzspannung (*AD780*), die an den dafür vorgesehenen Port des ADCs angeschlossen wird. Der gewandelte Wert wird über die SPI Schnittstelle an den Mikrocontroller übertragen. Dafür werden die vier Leitungen, *SCKL*, $\overline{\text{CS}}$, *MISO* und *MOSI*, zwischen den beiden Komponenten benötigt. Ein detaillierter Ausschnitt des ADCs, aus dem Schaltplan der Erfassungseinheit, ist in Abbildung 8.2 dargestellt.

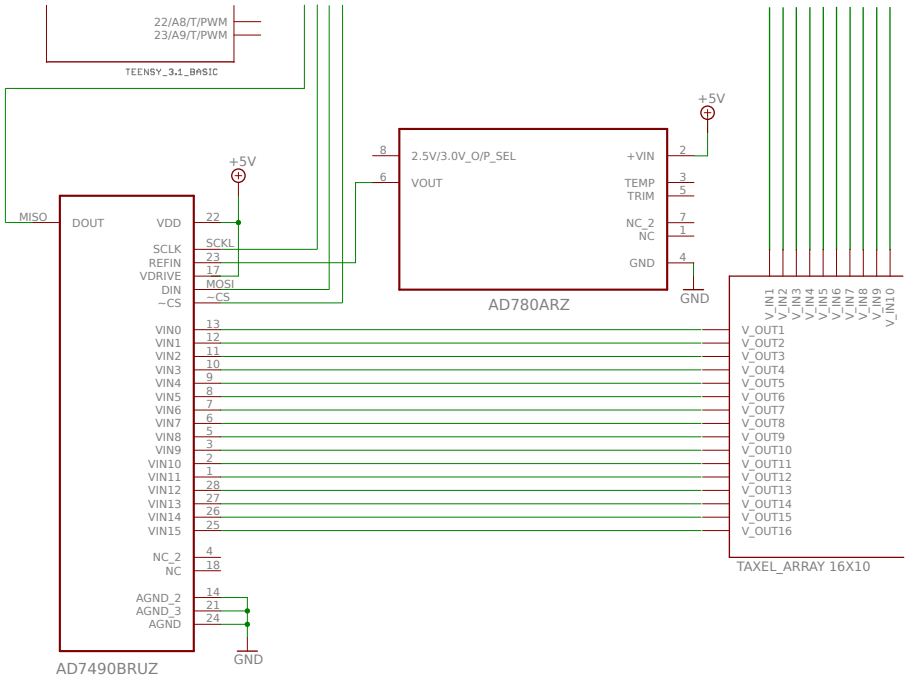
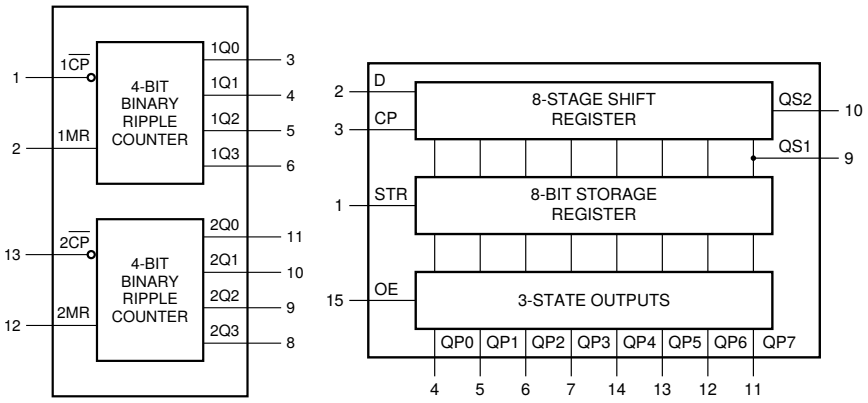


Abbildung 8.2: *Ausschnitt des ADCs aus dem Schaltplan der Erfassungseinheit.*

Schieberegister

Für die erwähnte Iteration der Spalten, zur Ansteuerung der Taxel, wurde in 5.2.2 diese über I/O-Ports des Mikrocontrollers geschaltet. Um die Schaltung allerdings unabhängiger zu gestalten, wird ein 5-Bit Zähler und ein ein 10-Bit Schieberegister hinzugefügt. Damit ist nicht nur die Erfassung von 16 Taxeln, sondern aller 160 in einem Stück möglich, ohne dass der Mikrocontroller eingreifen muss. Zur Steuerung der Spalten, wird dazu indirekt das $\overline{\text{CS}}$ -Signal der SPI

Übertragung genutzt. Bei jedem $\overline{\mathrm{CS}}$ -Takt, wird eine Konvertierung durchgeführt, was bedeutet, dass nach 16 Takten alle 16 Taxeldaten der aktiven Spalte ausgelesen sind; somit muss alle 16 Takte die nächste Spalte mit Spannung versorgt werden. Dazu wird mithilfe des 5-Bit Zählers ein neues Signal erzeugt, dass alle 16 Takte, wiederum einen Takt erzeugt. Aufgrund der Verfügbarkeit, wird der *74HC393* ein Dual *4-bit binary ripple counter* verwendet. Dieser Baustein besteht intern aus zwei 4-Bit Zählern und entspricht allen Anforderungen. Dieser ist detailliert in Abbildung 8.3(a) dargestellt. Neben der Verbindung der $5V$ Spannung und Masse, wird die $\overline{\mathrm{CS}}$ -Leitung der SPI Schnittstelle an den Eingang des ersten Zählers (*1~CP*) angeschlossen. Bei dem Eingang handelt es sich auch, wie bei dem $\overline{\mathrm{CS}}$ -Signal, um einen invertierten Takt, somit ist kein Inverter nötig. Da es sich intern um zwei 4-Bit-Zähler handelt, aber 5-Bit benötigt werden, wird der Ausgang *1Q3* an den Eingang *2~CP* geschaltet. Damit steht am Port *2Q0* ein Sechzehntel der Taktrate des $\overline{\mathrm{CS}}$ -Signals zur Verfügung. Um die Zähler seitens des Mikrocontrollers noch steuerbar zu halten, werden beide Reset Ports (*1MR* und *2MR*) an einen gemeinsamen I/O-Port angeschlossen.



(a) *74HC393* (b) *74HC4094*

Abbildung 8.3: *Funktionsdiagramm des Dual 4-bit binary ripple counter 74HC393 und Logikdiagramm des 8-stage shift-and-store bus register 74HC4094*

Das neu erzeugte Signal *Q5* wird nun dazu benutzt, die Spalten der Sensormatrix zu schalten. Wie schon bei der Beschreibung des Verfahrens unter 5.2, wird das Selektieren durch das Schieben einer logischen „1“ durch die Spalten umgesetzt. Aus diesem Grund bietet sich hier ein 10-Bit Schieberegister an. In der Initialisierungsphase schiebt der Mikrocontroller eine „1“ in das Register, damit ist die erste Spalte mit Spannung versorgt. Anschließend wird diese „1“, durch logische Nullen, getriggert von dem Signal *Q5*, das alle 16 $\overline{\mathrm{CS}}$ -Takte erzeugt wird, durch das Register geschoben und selektiert somit immer die jeweilige Spalte.

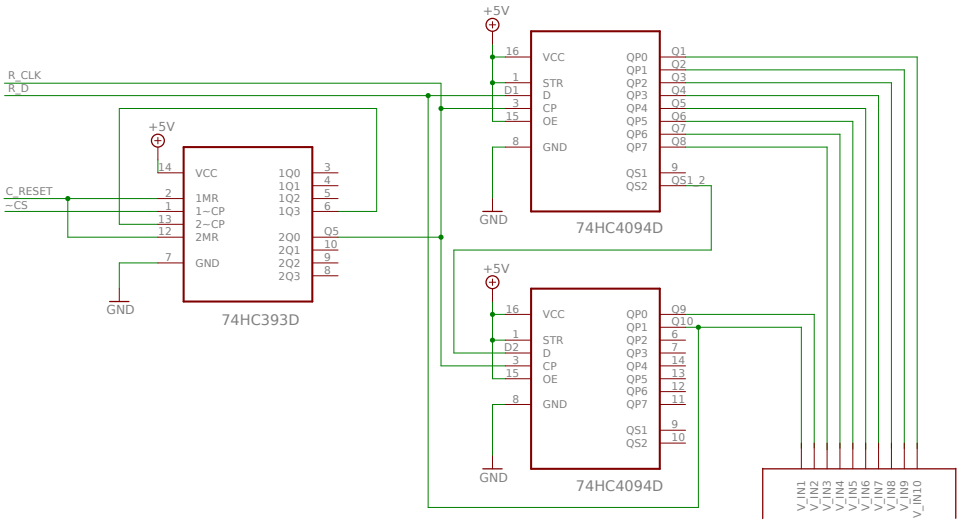


Abbildung 8.4: *Ausschnitt des Schieberegisters aus dem Schaltplan der Erfassungseinheit.*

Wieder aus Gründen der Verfügbarkeit, wird kein 10-Bit Schieberegister, sondern es werden zwei *8-stage shift-and-store bus register* verwendet. Dabei handelt es sich um das 8-Bit Schieberegister *74HC4094* der Firma *NXP*. Diese sind wie in Abbildung 8.4 miteinander verknüpft, um das 10-Bit Schieberegister umzusetzen. Dazu werden beide über den selben Takt (*Q5*) getriggert. Der Dateneingang *D1*, wird zum einem an den Mikrocontroller angeschlossen, um die erste „1“ in das Regster zu schieben, zum anderen an den Ausgang *Q10*. Dadurch wird die „1“ nach dem zehnten Takt wieder an die erste Stelle des Registers geschrieben. Wichtig ist, dass die *D1* Leitung zum Mikrocontroller, nach dem einmaligen setzten, auf *Low* gezogen wird. Für den Übertrag in das zweiten Register, wird an dessen Eingang *QS_2* angeschlossen. Dadurch wandert die „1“ nach dem achten Takt in sein Register. Damit die Daten beim auftreten des jeweiligen Takts direkt zu den Ausgängen *Q1-Q10* durchgereicht werden, müssen die Eingänge *STR* und *OE* an $5V$ Spannung liegen. Dieses ermöglicht die automatische Iteration der Taxel-Spalten.

Mikrocontroller

Für die Initialisierung der Taxel-Erfassung, ist der Mikrocontroller zuständig. Sowohl Zähler, Schieberegister als auch der ADC, müssen vor Beginn der Datenerfassung konfiguriert werden. Da sich die einzelnen Komponenten gegenseitig beeinflussen, ist die Reihenfolge der Konfiguration von entscheidender Bedeutung, welches unter 8.1.2 ausführlich beschrieben wird. Für die Initialisierung rund um das Schieberegister werden drei Leitungen benötigt, die jeweils an den I/O-Ports des Mikrocontrollers angeschlossen sind. Dabei handelt es sich zum einen um eine Leitung zum zurücksetzten des Zählers (*C_RESET*). Zum anderen um die zwei nötigen Leitungen (*R_D* und *R_CLK*), um Initial die „1“ in das Register zu schieben. Die Konfi-

guration des ADCs wird über die SPI Schnittstelle vollzogen, über die der Mikrocontroller auch die Taxeldaten erfasst.

Zum Einsatz kommt hier der Mikrocontroller *MK20DX256* bzw. das komplette Board *Teensy 3.1*. Dieses hat in der Evaluierung unter 5.3 ergeben, dass es alle Anforderungen erfüllt und im Vergleich zu den anderen evaluierten Mikrocontrollern, besonders bezogen auf die Geschwindigkeit, die besten Ergebnisse liefert. Da das *Teensy 3.1* Board in seinen Maßen sehr klein ist (vgl. 5.3.4), wird an dieser Stelle darauf verzichtet die Schaltung rund um den *MK20DX256* selbst zu entwerfen. So wird das komplette Board in die Erfassungseinheit integriert. Neben den bereits erwähnten Leitungen der SPI Schnittstelle und den drei zur Initialisierung der Schieberegister-Schaltung, werden noch die Pins für die USB Schnittstelle, welche zugleich auch als Versorgungsspannung verwendet werden, benötigt. Zusätzlich wird noch der *Reset* Pin herausgeführt, um die Möglichkeit zu besitzen, den Controller von extern neu zu starten. Die Schaltung rund um das Mikrocontroller-Board, zeigt ein Ausschnitt aus dem Schaltplan in Abbildung 8.5.

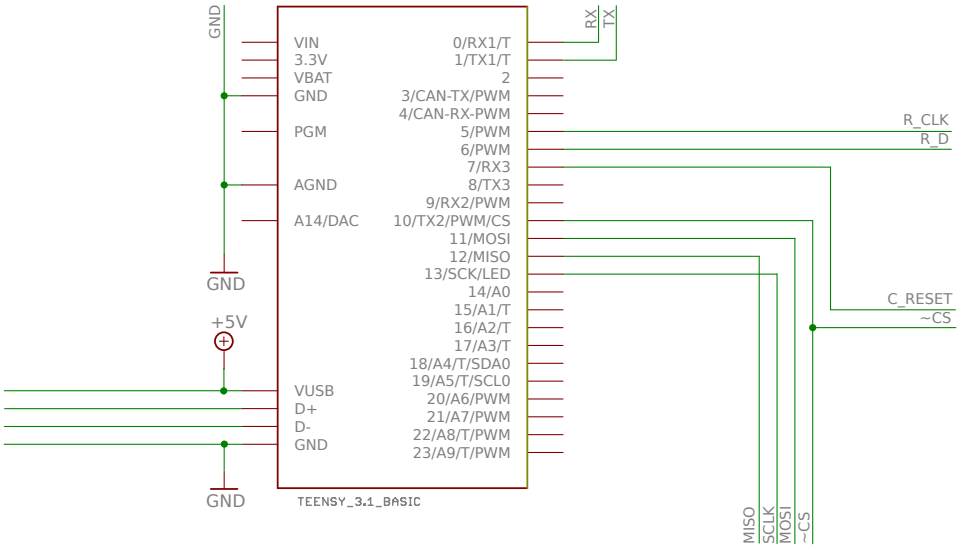


Abbildung 8.5: *Ausschnitt des Mikrocontrollers aus dem Schaltplan der Erfassungseinheit.*

8.1.2 Software auf dem Mikrocontroller

In diesem Abschnitt wird die entwickelte Software auf dem Mikrocontroller beschrieben. Diese hat zum einen die Aufgabe, die einzelnen Hardware-Komponenten, die im vorherigen Teil beschrieben wurden, zu initialisieren. Zum anderen ist sie dafür zuständig, die Erfassung der Taxeldaten einzuleiten, zu verwalten und die Daten weiterzuleiten. Mithilfe von abstrakten Flussdiagrammen, wird diese nachfolgend beschrieben.

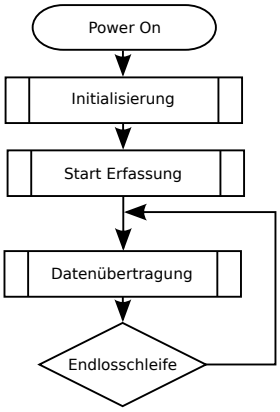


Abbildung 8.6: *Programmablauf auf der obersten Ebene*

In Abbildung 8.6 ist der Ablauf des Hauptprogramms dargestellt. Es besteht aus der Initialisierung der Hard- und Software, dem Starten der Erfassung und anschließend aus der Main-Schleife, die nur durch einen Hardware-Reset verlassen werden kann.

Initialisierung von Hard- und Software

Die Initialisierung der einzelnen externen Komponenten, aber auch die des Mikrocontrollers selbst, sind die entscheidendsten Voraussetzungen, damit die Erfassung der Daten reibungslos funktioniert. Alle Schritte sind in abstrakter Form in dem Flussdiagramm aus Abbildung 8.7 dargestellt.

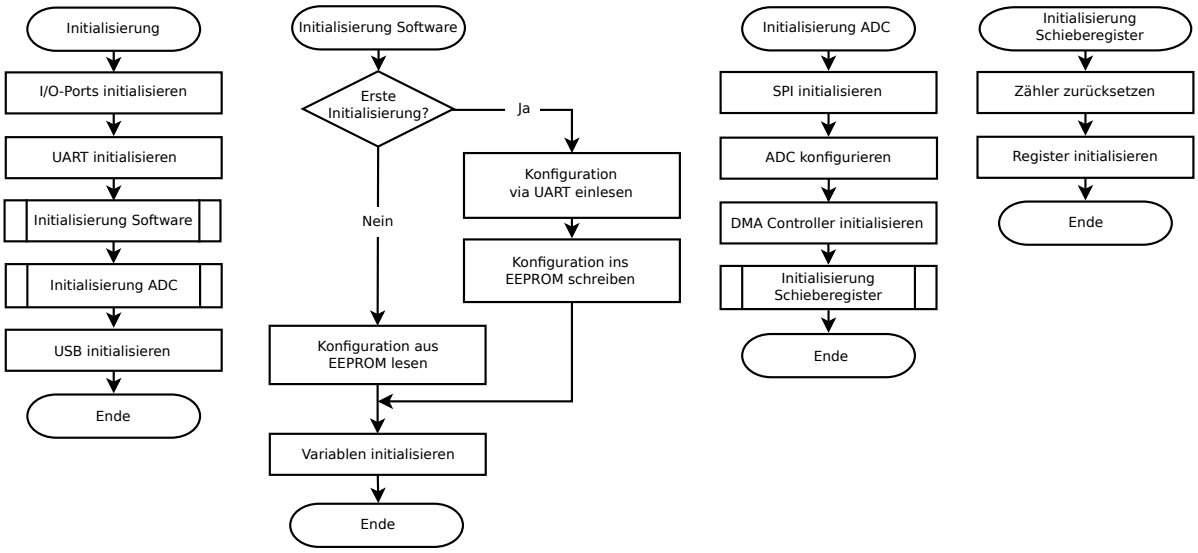


Abbildung 8.7: *Abstraktes Flussdiagramm der gesamten Initialisierung.*

Beginnend werden alle benötigten I/O-Ports in ihrer Richtung und Pegel gesetzt, da dieses sonst schnell zu fehlerhaften Verhalten, bis hin zu Defekten an den externen Komponenten, führen kann.

Als zweites wird die UART Schnittstelle konfiguriert, da über diese u. a. Debug-Nachrichten versendet werden, die auch schon bei der weiteren Initialisierung helfen können. Des Weiteren wird sie zur Konfiguration von Parametern bereits im nächsten Schritt eingesetzt.

Bei der Initialisierung der Software, wird zuerst überprüft, ob sich bereits eine Konfiguration der Erfassungseinheit im EEPROM befindet, bzw. ob es sich bei der Initialisierung um die erste Ausführung handelt. Steht noch keine Konfiguration zur Verfügung, wird diese über die UART Schnittstelle eingelesen. Dabei handelt es sich in dem Fall, um die eindeutige ID der Erfassungseinheit, die nachfolgend in den EEPROM geschrieben und beim der nächsten Initialisierung automatisch aus dem EEPROM gelesen wird. Weitere Konfigurationsparameter, die allerdings nur optional sind, wären u. a. die Datengröße der Taxel und die SPI Übertragungsgeschwindigkeit. Anschließend werden allen Variablen ihre entsprechenden benötigten Werte zugewiesen oder der benötigte Speicher allokiert. Dabei handelt es sich z.B. um die 160 Taxeldaten, die Abhängig der Konfiguration, 4-12Bit groß sein können. Als *Default*-Wert ist 8Bit konfiguriert, mit dem auch in diesem System gearbeitet wird.

Durch die Initialisierung der Software stehen jetzt, sowohl der benötigte Speicher für die Taxeldaten, sowie die Konfigurationsparameter, die für die Initialisierung des ADCs benötigt werden, bereit. Da der ADC über die SPI Schnittstelle angesprochen wird, ist die Konfiguration dieser die grundlegendste Vorbedingung. Neben der Generierung des Taktes *SCKL* ($19,2MHz$) (vgl. 5.3.4), wird die Bit-Reihenfolge (*MSB*), der SPI Mode (*Mode 2*), der Chip-Select Pin (*Output*, *High*) eingestellt und die Schnittstelle in den Master-Mode versetzt. Dabei handelt es sich um Einstellungen, die von dem ADC Baustein *AD7490* vorausgesetzt werden, um mit ihm zu Kommunizieren. So kann dieser anschließend, wie bereits in 5.2.1 beschrieben, konfiguriert werden. Ab diesem Zeitpunkt ist die Erfassung der Taxel über den ADC bereits möglich, dennoch ist die Initialisierung noch nicht abgeschlossen. Denn die komplette Datenerfassung der Taxel soll über den DMA Controller laufen, um so nicht nur an Geschwindigkeit zu gewinnen (vgl. 5.3.5), sondern auch die CPU möglichst wenig zu Belasten und deren Zeit in Anspruch zu nehmen (vgl. 5.3.1). Dazu wird in dem SPI Register, der DMA-Mode und in dem DMA Modul die *SCKL* aktiviert. Des Weiteren ist es nötig der SPI-Schnittstelle mitzuteilen, dass sie die $\overline{\mathrm{CS}}$ -Leitung automatisch bedient. Zusätzlich werden zwei DMA Kanäle, jeweils einer für SPI TX und SPI RX, benötigt. Das ermöglicht es dem DMA Controller mit der SPI Schnittstelle und dem dahinter liegenden ADC zu kommunizieren. Des Weiteren wird ein Interrupt konfiguriert, der einen IRQ auslöst, sobald alle gewünschten Daten gesendet bzw. empfangen wurden.

Der DMA Controller ermöglicht es nun die 16 Daten eines ADCs, ohne dabei viel CPU Zeit zu benutzen, zu empfangen. Um allerdings alle 160 Taxeldaten zu erfassen, müssen die jeweiligen Spalten der Matrix iteriert werden. Allerdings ist dieser nicht in der Lage, diese Spalten der Matrix eigenständig zu schalten. Deshalb kommt hier das Schieberegister zum Einsatz.

Dieses wird durch die von der Hardware getriggerte $\overline{\mathrm{CS}}$ -Leitung, automatisch durchiteriert (vgl. 8.1.1). Da durch die vorherige Konfiguration des ADCs, das $\overline{\mathrm{CS}}$ -Signal bereits den Zähler inkrementiert hat, wird dieser über die *C_RESET*-Leitung zurückgesetzt. Zusätzlich benötigt das Schieberegister Initial eine „1“ in ihrem ersten Register, welches mithilfe der beiden Signale *R_CLK* und *R_D* umgesetzt wird.

Im letzten Punkt der Initialisierung, wird die USB Schnittstelle, zur Datenübertragung an das Zielsystem, über die Verarbeitungseinheit, konfiguriert. Da durch den verwendeten USB-Stack bereits eine Vielzahl an fertigen USB Geräten zur Verfügungen stehen (vgl. 5.3.4), wird in diesem Fall auch auf eben eines dieser zurückgegriffen. Die Taxeldaten sollen wie in 5.3 beschrieben, versendet werden. Dabei handelt es sich um eine einfache USB-Bulk-Übertragung, wodurch sich die Verwendung der USB *Communication Device Class* (CDC) besonders anbietet, da dieses genau die einfache Bulk-Übertragung durchführt. Damit wird nicht nur die Arbeit auf Seiten des Mikrocontrollers verringert, sondern zugleich auch auf dem Zielsystem, denn für diese USB Geräte, stehen auf allen bekannten Betriebssystemen, bereits Treiber bereit. Da im Rahmen dieser Arbeit keine eigene VendorID zur Verfügung steht, wurde eine frei definiert. Um einen kleinen Bezug auf den Herstellen zu geben, in diesem Fall die Hochschule für Angewandte Wissenschaften, wurde das Kürzel HAW, anhand ihrer Stellen im Alphabet (H=8, A=1, W=24), gewählt. Dabei wurde darauf geachtet, dass die ID 0x8118 bisher noch nicht vergeben wurde, um Konflikte mit anderen Geräten zu vermeiden. Zusätzlich wird im Deskriptor des USB-Devices noch die ProductID definiert, um dem Endsystem mitzuteilen, um welches Gerät es sich genau handelt. In diesem Fall wird für den *16x10 tactile sensor* die ID 0x0010 gewählt. Neben den IDs im Deskriptor werden noch einige String-Deskriptoren definiert, um das Gerät zusätzlich, auch in Form von Text, zu beschreiben.

Start Erfassung

Nach der Initialisierung, wird die Datenerfassung, wie in dem Ablaufdiagramm aus Abbildung 8.8 abstrakt dargestellt, gestartet.

Für diese Erfassung stehen drei Datenpuffer zur Verfügung, dessen Anwendung und Zweck in den nachfolgenden Teilen erläutert wird. Um die Daten über den DMA Controller per SPI Schnittstelle zu erhalten, ist es nach der vorherigen Initialisierung nur noch nötig, dem DMA Controller die Adresse des Datenpuffers und die Anzahl der zu übertragenen Daten mitzuteilen. Dabei ist es wichtig, dass der übergebene Datenpuffer nicht zeitgleich über die USB Schnittstelle übertragen wird. Dieses wird anhand eines *Pointers* (*p_current_buffer*) umgesetzt, der immer die Adresse des Datenpuffers beinhaltet, der gerade versendet wird. Damit ist sichergestellt, dass die Daten beim Übertragen nicht durch den DMA Controller manipuliert werden. Anschließend kann die Datenerfassung gestartet werden.

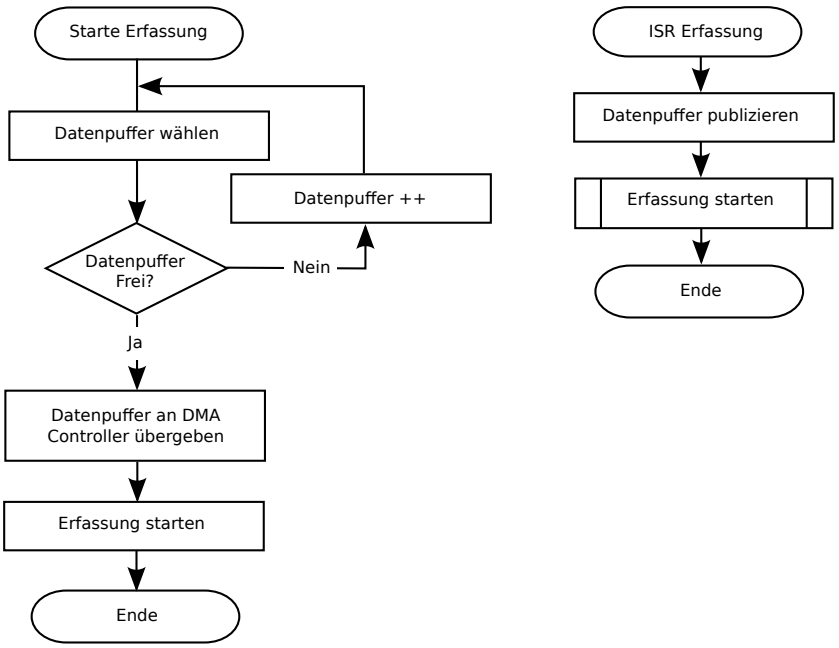


Abbildung 8.8: *Abstraktes Flussdiagramm der Erfassung.*

Wurden alle 160 Daten, je $16bit$, empfangen, schmeißt der DMA Controller einen Interrupt, der von der ISR des Mikrocontrollers aufgefangen und bearbeitet wird. Diese ISR ist in Abbildung 8.8 in einem Flussdiagramm dargestellt. Um dem Prozess der Übertragung mitzuteilen, in welchem Puffer sich die aktuellen Daten befinden, wird die Adresse dieses Puffers in die Variable *p_next_buffer* geschrieben. Anschließend wird die Übertragung erneut gestartet. Da in diesem Fall schon zwei Puffer belegt sein könnten, einer der zur Zeit übertragen wird und der zweite mit den zuvor aktuellen, erfassten Daten, wird ein dritter Puffer benötigt. Sollte der DMA Controller die zweite Erfassung abschließen, bevor die Übertragung der vorherigen Daten begonnen hat, wird der alte Puffer durch den aktuellen ersetzt. Somit stehen für die nächste USB Übertragung, immer die aktuellsten Daten zur Verfügung.

Übertragung in der Endlosschleife

Die Übertragung der Daten läuft in einer Endlosschleife, die nur durch einen Hardware-Reset verlassen werden kann (siehe Abbildung 8.6). Denn die einzige Aufgabe, die diese Erfassungseinheit hat, ist es Daten zu erfassen und diese zu übertragen. So versucht der Mikrocontroller durchgehend diese erfassten Daten zu verschicken. Dazu prüft er, ob bereits ein Datenpuffer mit erfassten Daten vorhanden ist. Ist dieses nicht der Fall, springt er aus der Datenübertragung. Da jedoch keine weitere Aktion in der Hauptschleife vorhanden ist, springt er direkt wieder in die Übertragung und prüft, ob Daten vorhanden sind. Hat der DMA Controller über die Variable *p_next_buffer* mitgeteilt, dass aktuelle Daten vorhanden sind, wird dieser Puf-

fer für den DMA Controller gesperrt (vgl. 8.1.2). Anschließend werden die 16-Bit-Datenwerte je nach Konfigurierung in die entsprechenden Datenwerte umgewandelt, um die Datenübertragung dadurch zu beschleunigen. In diesem Fall werden die Taxelwerte von $16bi$ auf $8bit$ reduziert. Dazu wird die 4-Bit-Portzuweisung (vgl. 5.2.1) und zusätzlich die 4 niederwertigsten Bits der Daten entfernt. Diese Entfernung der 4 Bits hat keine Auswirkung auf die Qualität der Daten, da der Sensor erst bei Werten größer als 20 anspringt (vgl. 5.1). Da durch die Wegnahme der untersten 4 Bit maximal ein Dezimalwert von 15 erzeugt werden kann, entsteht kein Verlust bei der Qualität, jedoch ein Gewinn bei der Übertragungsgeschwindigkeit.

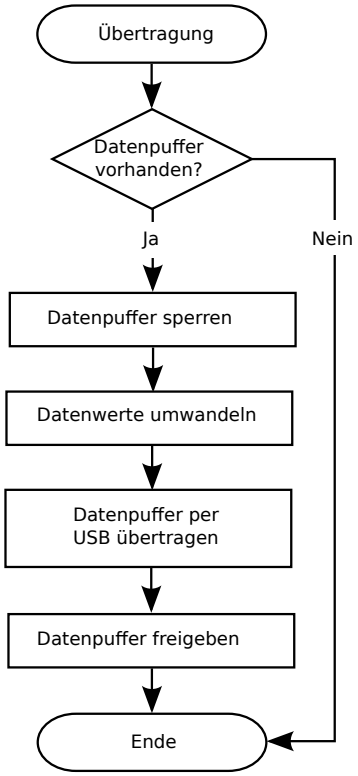


Abbildung 8.9: *Abstraktes Flussdiagramm der Datenübertragung.*

Anschließend werden die umgewandelten Daten über das USB CDC Device übertragen. Um den Puffer anschließend wieder für den DMA Controller frei zugeben, wird die Adresse des Puffer aus der Variable *p_current_buffer* entfernt. Der gesamte Ablauf ist in der Abbildung 8.9 dargestellt.

8.2 Verarbeitungseinheit

Die Verarbeitungseinheit hat die Aufgabe die Daten der Erfassungseinheiten an das Zielsystem, den Roboter *SCITOS G5*, weiterzuleiten. Dazu wurden in Kapitel 6 mehrere Ansätze evaluiert, mit dem Resultat, dass die besten Ergebnisse mit einer einfachen Weiterleitung der Daten erzielt werden kann. Aus diesem Grund wird an dieser Stelle genau dieser Ansatz gewählt.

Verwendet wird auch hier der USB-Hub *USB2504* der Firma *Microchip*. Er unterstützt die volle USB 2.0 Spezifikation und benötigt keine Firmware Entwicklung. Wie schon unter 6.1.1 erwähnt, können diverse USB-Full-Speed-Geräte (12Mb/s), parallel mit ihrer vollen Geschwindigkeit übertragen. So ist es in diesem Setup möglich, dass beide Erfassungseinheiten ihre Daten durchgehen, ohne Einschränkungen auf dem Weg der Übertragung, an das Zielsystem schicken können.

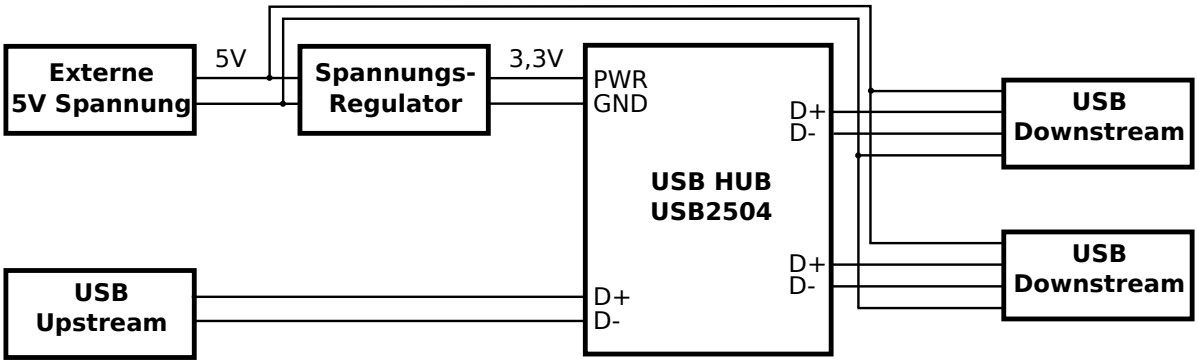


Abbildung 8.10: *Einfacher Aufbau der Schaltung der Verarbeitungseinheit.*

Ein vereinfachter Aufbau der Schaltung ist in Abbildung 8.10 dargestellt. Die Inbetriebnahme des Hubs ist mit sehr wenig Aufwand verbunden, u. a. auch weil bereits alle benötigten Widerstände für die USB Ports in dem Baustein enthalten sind. Aus diesem Grund wird hier auf eine detaillierte Beschreibung verzichtet und auf die Schaltung des offizielle Evaluierungsboard verwiesen [**?**].

Der Hub liest beim Start seine Konfiguration aus dem EEPROM, die es ermöglicht den Hub auch ohne weitere Modifikationen oder Anpassungen im vollen Umfang zu verwenden. In diesem Fall wird aber der Deskriptor angepasst, um den Anforderung gerecht zu werden. So wird über die *I²C* Schnittstelle, der Deskriptor, wie unter 8.1.2, modifiziert. Als ProductID erhält der Hub in diesem Fall die 0x0001.

8.3 Software auf dem Zielsystem

Die Entwicklung einer Software auf dem Zielsystem soll nicht Teil dieser Arbeit sein. Durch das verwendete ROS auf dem Computersystem des *SCITOS G5*, stehen bereits diverse Softwarelösungen zur Verfügung, die auch eine Vielzahl von Greifoperationen beinhalten [vgl. 64]. Eine Arbeit die sich durch die Inspiration des menschliche Greifverhalten, ausführlich mit diesem Thema beschäftigt hat, ist in [**?**] beschrieben.

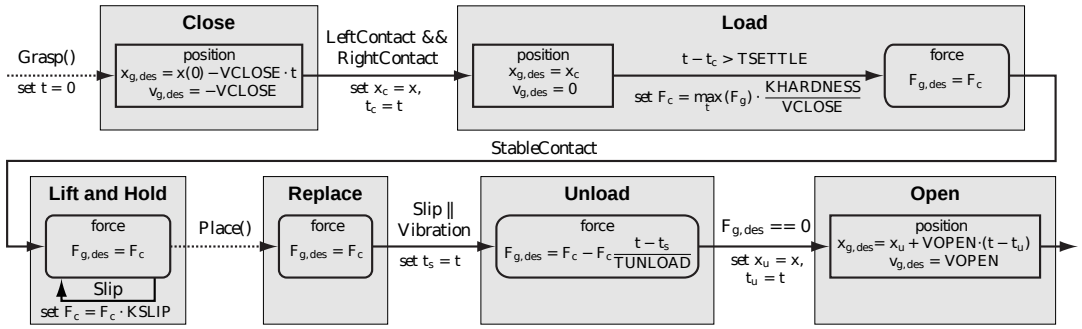


Abbildung 8.11: *Ablaufdiagramm einer nach dem Menschen inspirierter Greifkontrolle für einen Roboter aus [**?**].*

Um die bisher erreichten Ergebnisse aber auch auf dem System, für das der entwickelte Sensor gedacht ist, zu bestätigen, wird eine kleine Anwendung auf diesem entwickelt. Dieses soll die Daten beider Erfassungseinheiten über die Verarbeitungseinheit empfangen, grafisch darstellen und Kenndaten bereitstellen.

Die Anwendung wird in *Python* entwickelt, was eine schnelle und einfache Entwicklung ermöglicht. Des Weiteren stellt ROS eine *Python* API zur Verfügung.

Die Applikation besteht aus zwei Teilen, zum einen aus der GUI, die für die grafische Darstellung der Werte zuständig ist und zum anderen aus zwei Threads, die über die USB Schnittstelle die Taxeldaten erfassen. Die GUI zeigt jeweils alle 160 Taxel Daten, in der exakten Nachbildung des Sensormoduls an. Jeder Taxel ist je nach Wert, farbig dargestellt und beinhaltet den jeweiligen Wert. Zusätzlich wird auch der aktuelle Datendurchsatz angezeigt, der sich jeweils auf die Erfassungseinheit bezieht. Für die Datenerfassung werden die Daten durchgehend von den Threads erfasst und über eine *Queue* der GUI übergeben, welche die Daten dann alle $10ms$ in der Grafik aktualisiert.

Bei den Daten handelt es sich um 162 Byte, die in einem Byte-Stream übertragen werden. Es stehen, außer dem Start-Byte (0xFF) und Stop-Byte (0xFE), keine weiteren Informationen in den übertragenen Paketen bereit. Um die reinen Datenpakete mit den Taxeln in der richtigen Reihenfolge zu erhalten, wird zuerst der Stream synchronisiert. In diesem Fall, wird Byte für Byte überprüft, ob es sich dabei um das Start-Byte handelt. Trifft diese Bedingung zu, wer-

den ab diesem Zeitpunkt immer genau 162 Byte empfangen. Da es nun möglich ist, dass das System bei hoher Last Pakete nicht rechtzeitig empfangen kann und diese dadurch verworfen werden, wird durch eine anschließende Prüfungen diesem entgegengewirkt. So werden die Daten mithilfe einer *regular expression* zwischen dem Start- und Stop-Byte extrahiert, was zugleich sicherstellt, dass es sich um den erwarteten Paketrahmen gehandelt haben muss. Dieses Verfahren eignet sich allerdings nur bei einer dauerhaften bzw. längeren Datenübertragung. Durch das anfängliche Warten auf das Start-Byte kann sehr viel Zeit vergehen und die Übertragung, bei nur einem Datensatz, enorm verringern (vgl. 5.3). Da es sich hierbei aber um eine dauerhafte Erfassung handelt, spielt die anfängliche Synchronisation keine zeitliche Rolle.

Durch dies Testanwendung, konnten die Taxeldaten der beiden Erfassungseinheiten mit einem Datendurchsatz von über 13000 Paketen pro Sekunde, mit jeweils 160 Taxeldaten erfasst werden. Das entspricht einem Gesamtdurchsatz von über 6000 und damit einer Abtastrate von mehr als $6kHz$.

Abbildung 8.12: *Testanwendung auf dem Zielsystem*

8.4 Auswertung

Durch die vorangegangene Evaluierung, konnte in wenigen Schritte eine Umsetzung erstellt werden.

Das entwickelte Sensorsystem besteht aus zwei Erfassungseinheiten, die jeweils aus einem Array von 10x16 Taxeln bestehen. Dabei haben die Taxel eine hohe Dichte der Auflösung von unter $5mm$. Auch in Sachen Geschwindigkeit, liegt dieses System deutlich in den Anforderungen. So schafft es einen Durchsatz, der Taxeldaten beider Erfassungseinheiten zusammen, von über 6000 Paketen pro Sekunde, was eine Abtastrate von über $6kHz$ entspricht. Diese Ergebnisse konnte bisher kein vergleichbares System liefern. Des Weiteren lässt es sich mit einfachen Mitteln herstellen und erfüllt somit auch diese Anforderung. Zwar ist ein maschinell erstelltes PCB notwendig, um die aufgedruckten Elektroden zu verwirklichen, dieses lässt sich aber aufgrund der nur zwei benötigten *Layer* leicht erstellen. Des Weiteren lassen sich alle benötigten Bauteile, die dieses System benötigt, per Hand integrieren. Auch das Aufteilung in mehrere Module, macht die Herstellung deutlich weniger komplex und aufwendig.

Außerdem wurden ausschließlich Komponenten benutzt die zum Standard gehören und einfach zu beschaffen sind. Durch die ausschließliche Verwendung von Standard Interfaces, wird dieses verstärkt. Zusätzlich stehen eine Vielzahl an kostengünstigen Alternativen zur Verfügung. Somit ist jeder in der Lage sich dieses oder ein vergleichbares System, ohne großen Aufwand zu erstellen.

9 Zusammenfassung und Fazit

Mit diesem Kapitel schließt die Entwicklung des taktilen Sensorsystems zur Schlupferkennung ab. Dazu wird nachfolgend die Arbeit zusammengefasst und ein Fazit aus den Erkenntnissen gezogen.

9.1 Zusammenfassung der Arbeit

Die Entwicklung von Assistenzrobotern steigt stetig voran und zeigt immer mehr seine Wichtigkeit. Besonders großes Potential liegt dabei auf Roboter, im Gesundheitsmarkt und in der Altenpflege, aber auch in Privathaushalten steigt das Interesse. In Zukunftsszenarien ist ein Leben ohne Roboter wohl kaum noch wegzudenken, besonders wenn man sich die fortschreitende Automatisierung und Digitalisierung vor Augen führt.

Eine der wichtigsten Fähigkeiten, ohne die auch der Alltag eines Menschen sehr schwierig wäre, ist die Manipulation in seiner Umgebung. Deshalb stellt dieses eine der wichtigsten Voraussetzungen dar, um sich das Wort Assistenz in Assistenzroboter zu verdienen und den Sprung in die menschliche Umgebung zu schaffen.

Für den Großteil der Manipulation wird, wie beim Menschen die Hand, auch beim Roboter ein Tast- und Greifwerkzeug benötigt. Dieses muss mit allen Herausforderungen, die in einem menschlichen Umfeld entstehen, zurecht kommen. Das bei Menschen die Wahrnehmung an den Händen besonders intensiv und ausgeprägt ist, zeigt die Wichtigkeit einer vergleichbaren taktilen Einheit am Roboter.

Da sich das menschliche Umfeld ständig im Wandel befindet, wird der Roboter immer wieder auf neue und damit ihm unbekannte Objekte treffen. So ist eine der größten Herausforderung, das Greifen und somit das Manipulieren, von unbekannten Objekten. Genau dieses Problem wird in dieser Arbeit ausführlich analysiert und zur Lösung ein taktiles Sensorsystem entwickelt, das die Möglichkeit schaffen soll, genau diese unbekannten Objekte sicher zu greifen.

Im Abschnitt der Grundlagen (Kapitel 2) wird Hintergrundwissen für die Entwicklung des taktilen Sensorsystems erläutert. Dazu wird im ersten Schritt der Begriff taktile Sensorik beschrieben und ein Einblick in mögliche Technologien gegeben, die sich für ein solches System eignen könnten. Da es sich bei diesem System der taktilen Wahrnehmung nicht nur alleine um die Sensorik dreht, werden anschließend auch noch Themen wie Datenübertragung, Prozessor

Einheiten und letztendlich auch der Roboter selbst, behandelt. Durch diese Einführung in die grundlegende Thematik, wird das Verständnis für den Rest der Arbeit gestärkt.

In Kapitel 3 werden die Anforderungen an das zu entwickelnde System erarbeitet und aufgestellt. Dazu werden sich beginnend, anhand von verwandten und vergleichbaren Arbeiten, erste Eindrücke in die Problematiken und deren Problemlösungen geholt. Um dem Faktor des menschlichen Umfeld gerecht zu werden, folgt anschließend eine Betrachtung des menschlichen Tastsinns. Unter anderem dadurch kann gezeigt werden, welche Probleme es für ein solches System zu lösen gilt. Diese Problemlösungen werden nachfolgend aufgestellt und versucht diese, durch eine ausführliche Analyse in diversen Bereichen der Thematik, in Anforderungen an das taktile Sensorsystem umzuwandeln. So stehen am Ende dieses Abschnitts Anforderung, wie eine Taktrate eine von $800Hz$ bei der Erfassung der taktilen Daten, definiert. Weitere wichtige Anforderungen an dieses System sind geringe Kosten und ein ebenfalls geringer Aufwand, denn nur so kann ein System massentauglich werden und den Weg in alle Haushalte finden. Des Weiteren fördert es die Entwicklung, da jeder in der Lage ist, sich dieses System für seine Zwecke der Forschung zu erstellen und damit die Entwicklung weiter vorantreiben kann.

Mit dem Bezug auf das vorherige Kapitel wird nun das Konzept (Kapitel 4) erstellt. Dieses ist dabei in drei Komponente, auf verschiedenen hierarchischen Ebenen, unterteilt. Auf der ersten Ebene stehen die Erfassungseinheiten, welche für die Erfassung der einzelnen Taxel und das anschließende weiterreichen dieser Daten an die nächste Ebene zuständig sind. An dieser Stelle befindet sich dann die zweite Komponente, die Verarbeitungseinheit, die sich in der nächst höheren Ebene, der hierarchischen Übertragungstruktur, befindet. Sie hat zum einen die Aufgaben, die Erfassungsmodule zu steuern und die Daten der Module wiederum an das Endsystem weiterzureichen. Als letzte Instanz im System steht dann die dritte Einheit, das Endsystem, welches die Daten der vorherigen Module aufnimmt und dadurch dem Roboter wertvolle Informationen liefern kann.

Da es bei einem System wie diesem, um ein riesiges Spektrum an möglichen Problemlösungen handelt, wird vor der Umsetzung dieses Konzepts eine ausführliche Evaluierung (Kapitel 5, 6 und 7) des Systems und deren einzelnen Teilkomponenten durchgeführt. Dieses schafft nicht nur einen besseren Überblick, sondern ermöglicht es auch schon früh Fehler zu erfassen und mehrere Lösungen mit einzubeziehen. Zusätzlich ist der Erfolg für die Umsetzung, unter Einhaltung der Anforderungen, schon vorher bekannt. So haben die Ergebnisse der Evaluierung ergeben, dass durch dieses Konzept alle Anforderungen erfüllt werden konnten und in manchen Fällen sogar noch übertroffen wurden. Auch im Vergleich zu vergleichbaren Arbeiten gehört dieses Konzept, bezogen auf die Geschwindigkeit und die Kosten für Material und Aufwand, zu den führenden auf diesem Gebiet.

Der weitere Fokus der Arbeit liegt in der Umsetzung des taktilen Sensorsystems (Kapitel 8). Dabei wurde sich in nahezu allen Teilen an das zuvor evaluierte Konzept gehalten und konnte allen Anforderungen gerecht werden. Es besteht aus zwei Erfassungseinheiten, die jeweils aus

einem Array von 10x16 Taxeln bestehen. Die Taxel bestehen aus zwei aufgedruckten Elektroden, die mit einem leitfähigen Polymer, die Druckkraft in elektrische Spannung umwandeln. Diese Spannung wird von einem externen ADC über eine Spannungsteilerschaltung aufgenommen. Damit nicht jeder der 160 Taxel einen ADC benötigt, wird durch geschicktes Schalten der einzelnen Taxel, nur ein ADC mit 16 Eingänge benötigt. Die durch den ADC erfassten Daten werden von einem Mikrocontroller, über die SPI Schnittstelle erfasst und über eine USB-Full-Speed ($12Mbit/S$) Übertragung weiter gereicht. Diese Daten werden nicht wie im Konzept vorgesehen, von der Verarbeitungseinheit erfasst und an das Zielsystem geschickt, sondern direkt weitergereicht. Dazu besteht sie ausschließlich aus einem USB 2.0 Hub, der die Datenströme der Erfassungseinheiten bündelt und über eine deutlich schnellere USB-High-Speed Verbindung weiterleitet. Da dieses erlaubt eine Vielzahl an Daten der verschiedenen Erfassungseinheiten parallel zu übertragen, hat es einen großen Geschwindigkeitsvorteil. Zusätzlich ist es, bezogen auf Material und Aufwand, eine besonders kostengünstige Möglichkeit.

Am Ende steht ein taktiles System zur Verfügung, dass zum einen alle Anforderungen erfüllt und besonders im Verhältnis von Kosten, Aufwand und Geschwindigkeit, hervorragende Ergebnisse erzielt. So schafft dieses System einen Durchsatz von Taxeldaten beider Erfassungseinheiten, von über 13000 pro Sekunde, was einer gesamten Abtastrate von mehr als $6kHz$ entspricht. Diese Ergebnisse konnte bisher kein vergleichbares System liefern.

9.2 Fazit zum taktilen Sensorsystem

Die stetig wachsende Nachfrage elektronischer Helfer im menschlichen Umfeld, zeigt die Wichtigkeit der Assistenzroboter in naher Zukunft. Diese Arbeit konnte durch die Konzeption und Umsetzung eines taktilen Sensorsystems dazu beitragen, die Forschung und Entwicklung in diesem Gebiet weiter voranzutreiben und damit die Mensch-Roboter-Interaktion zu fördern.

Besonders durch den kostengünstigen Aufbau und die geringe Komplexität des Systems, lässt sich dieses einfach für diverse Anwendungszwecke einsetzen und nachbauen. Darüber hinaus, ist das Sensorsystem einfach, in nahezu alle möglichen Systeme Integrierbar. Diese ist besonders auf die USB Schnittstelle und den einfach Byte-Stream zurückzuführen. Neben den bereits erwähnten Eigenschaften, sticht es zusätzlich besonders durch die hohe Geschwindigkeit ($6kHz$) bei der Datenerfassung hinaus und bietet dabei trotzdem eine gute Auflösung der Sensorfläche (10x16 Taxel bei einer Dichte von unter $5mm$).

Damit legt dieses Arbeit eine gute Basis für diverse nachfolgende und weiterführende Arbeiten auf dem Gebiet des autonomen Greifens, Erkundung, Interaktion und für vielen weitere, auch über diesen Bereich hinaus.

Literaturverzeichnis

[1] ABDALLAH, M.E.H. ; LINN, D.M.W.L. ; PLATT, R.H. ; REILAND, M.J.O. ; SANDERS, A.M.H.: *Interaktives Robotersteuerungssystem und Verwendungsverfahren*. 2011. – URL http://www.google.com/patents/DE102010045529A1?cl=de. – DE Patent App. DE201,010,045,529

[2] APPLE, Inc.: *Online Verfügbar*. 2009. – URL http://www.apple.com/ipodtouch/features/technology.html. – Zugriffsdatum: 2010-09-30

[3] AXELSON, Jan: *Parallel Port Complete: Programming, Interfacing, Using the PCs Parallel Printer Port*. New edition. Lakeview Research, 2 1997. – ISBN 9780965081917

[4] AXELSON, Jan: *Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems (Complete Guides series)*. 2nd. Lakeview Research, 12 2007. – ISBN 9781931448062

[5] AXELSON, Jan: *USB 2.0: Handbuch für Entwickler;[Datenübertragung und Transfertypen, USB On-The-Go; Entwicklung, Design und Programmierung von USB-Geräten; Visual-Basic-und Visual-C++-Anwendungen]*. mitp Verlags GmbH & Co. KG, 2007

[6] BANKS, Jessica L.: *Design and Control of an Anthropomorphic Robotic Finger with Multi-point Tactile Sensation*, University of Michigan, Master Thesis, May 2001

[7] BEGEJ, S.: Planar and finger-shaped optical tactile sensors for robotic applications. In: *Robotics and Automation, IEEE Journal of* 4 (1988), oct, Nr. 5, S. 472 –484. – ISSN 0882-4967

[8] BROOKS, Douglas: *Differential Signals*. – URL http://www.ieee.li/pdf/essay/differential_signals.pdf. – zuletzt abgerufen am 2014.11.18

[9] BRÜHN, A. ; BRÜHN, X.: *Verfahren zur Erfassung dreidimensionaler Oberflächen*. Juli 2011. – URL https://www.google.com/patents/DE102009054295A1?cl=de. – DE Patent App. DE200,910,054,295

[10] BUDWILL, Dr. D.: *Statistische Maßzahlen*. – URL http://www.chemgapedia.de/vsengine/vlu/vsc/de/ch/16/bbz/bbz_statwahr.vlu/Page/vsc/de/ch/16/bbz/bbz_statwahr_statmass.vscml.html

[11] BUNDES GESUNDHEITSBERICHTERSTATTUNG DES des. – URL http://www.gbe-bund.de/oowa921-install/servlet/oowa/aw92/dboowasys921.xwdevkit/xwd_init?gbe.isgbetol/xs_start_neu/&p_aid=i&p_aid=62187230&nummer=322&p_sprache=D&p_indsp=-&p_aid=1104856. – Zugriffsdatum: 2014.07.31. – zuletzt abgerufen am 2014.07.31

[12] BÄCHLIN, Marc ; MILIGHETTI, Giulio ; BEYERER, Jürgen ; KUNTZE, Helge-Björn: *Vorrichtung zum schlupf-überwachten, kraftschlüssigen Ergreifen, Halten und Manipulieren eines Objektes mittels einer Greiferanordnung*. 10 2006. – URL http://www.patent-de.com/20061026/DE102005032502B3.html

[13] BÄNISCH, Nicolas: *Druckmessung für den Zweifingergreifer eines Assistenzroboters*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, February 2012

[14] BÄNISCH, Nicolas: *Modellierung eines Antiblockiersystems zur Bremsweganalyse*, Hochschule für Angewandte Wissenschaften Hamburg, Ausarbeitung, February 2012

[15] BÄNISCH, Nicolas: *Tactile Sensing Arrays for Assistant Robots*, Hochschule für Angewandte Wissenschaften Hamburg, Projektarbeit, October 2014

[16] BÄNISCH, Nicolas: *Tactile Sensing Arrays for Assistant Robots*, Hochschule für Angewandte Wissenschaften Hamburg, Projektarbeit, Februar 2015

[17] BÜCH, Christian: *SPI - Serial Peripheral Interface*. june 2006. – URL http://www.uni-koblenz.de/~physik/informatik/MCU/SPI.pdf. – zuletzt abgerufen am 2015.08.30

[18] CANNATA, Giorgio ; MAGGIALI, M. ; METTA, G. ; SANDINI, G.: An embedded artificial skin for humanoid robots. In: *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, Aug 2008, S. 434–438

[19] CASTELLANOS-RAMOS, Julián ; NAVAS-GONZÁLEZ, Rafael ; MACICIOR, Haritz ; SIKORA, Tomasz ; OCHOTECO, Estíbalitz ; VIDAL-VERDÚ, Fernando: Tactile sensors based on conductive polymers. In: *Microsystem Technologies* 16 (2010), S. 765–776. – URL http://dx.doi.org/10.1007/s00542-009-0958-3. – ISSN 0946-7076

[20] CHOI, Byungjune ; CHOI, Hyouk R. ; KANG, Sungchul: Development of tactile sensor for detecting contact force and slip. In: *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, Aug 2005, S. 2638–2643

[21] CHUANG, Cheng-Hsin ; LIOU, Yi-Rong ; CHEN, Chih-Wei: Detection system of incident slippage and friction coefficient based on a flexible tactile sensor with structural electrodes. In: *Sensors and Actuators A: Physical* 188 (2012), Nr. 0, S. 48 – 55. – URL http://www.sciencedirect.com/science/article/pii/

S0924424712000969. – Selected papers from The 16th International Conference on Solid-State Sensors, Actuators and Microsystems. – ISSN 0924-4247

[22] Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, and Philips (Veranst.): *Universal Serial Bus Revision 2.0 specification*. 2.0. April 2000. – URL http://www.usb.org/developers/docs/

[23] COTTON, Darryl P. ; CHAPPELL, P.H. ; CRANNY, A. ; WHITE, N.M. ; BEEBY, S.P.: A Novel Thick-Film Piezoelectric Slip Sensor for a Prosthetic Hand. In: *Sensors Journal, IEEE* 7 (2007), May, Nr. 5, S. 752–761. – ISSN 1530-437X

[24] CUTKOSKY, Mark R. ; HOWE, Robert D. ; PROVANCHER, William R.: Force and Tactile Sensors. In: *Handbook of Robotics*. Springer, 2008, S. 455–476

[25] DAHIYA, RavinderS. ; VALLE, Maurizio: Tactile Sensing Technologies. In: *Robotic Tactile Sensing*. Springer Netherlands, 2013, S. 79–136. – URL http://dx.doi.org/10.1007/978-94-007-0579-1_5. – ISBN 978-94-007-0578-4

[26] DAHIYA, R.S. ; METTA, G. ; VALLE, M. ; SANDINI, G.: Tactile Sensing: From Humans to Humanoids. In: *Robotics, IEEE Transactions on* 26 (2010), Feb, Nr. 1, S. 1–20. – ISSN 1552-3098

[27] DAMIAN, D.D. ; MARTINEZ, H. ; DERMITZAKIS, K. ; HERNANDEZ-ARIETA, A. ; PFEIFER, R.: Artificial ridged skin for slippage speed detection in prosthetic hand applications. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Oct 2010, S. 904–909. – ISSN 2153-0858

[28] DEVICES, Analog: *16-Channel, 1 MSPS, 12-Bit ADC with Sequencer in 28-Lead TSSOP*. – URL http://www.analog.com/media/en/technical-documentation/data-sheets/AD7490.pdf

[29] DRIMUS, Alin ; KOOTSTRA, Gert ; BILBERG, Arne ; KRAGIC, Danica: Design of a flexible tactile sensor for classification of rigid and deformable objects. In: *Robotics and Autonomous Systems* 62 (2014), Nr. 1, S. 3 – 15. – URL http://www.sciencedirect.com/science/article/pii/S092188901200125X. – New Boundaries of Robotics. – ISSN 0921-8890

[30] FREESCALE: *Freescale*. 6, 11 2011. – URL http://cache.freescale.com/files/32bit/doc/ref_manual/K20P121M100SF2RM.pdf

[31] FUTAI, N. ; MATSUMOTO, K. ; SHIMOYAMA, I.: Simulation, fabrication and evaluation of microinductor-based artificial tactile mechanoreceptor embedded in PDMS. In: *Micro Electro Mechanical Systems, 2003. MEMS-03 Kyoto. IEEE The Sixteenth Annual International Conference on*, jan. 2003, S. 206 – 209. – ISSN 1084-6999

[32] FÄASSLER, Matthias: Force Sensing Technologies: Studies on Mechatronic. In: *SMAC - Stockholm Music Acoustics Conference*, 2010

[33] GASSMANN, Eugen ; GRIES, Anna: *Elektronische Druckmesstechnik: Grundlagen, Anwendungen und Geräteauswahl*. 1. Süddeutscher Verlag onpact, 12 2009. – ISBN 9783937889955

[34] HÄGELE, Martin ; BLÜMLEIN, Nikolaus ; KLEINE, Oliver: Wirtschaftlichkeitsanalysen neuartiger Servicerobotik-Anwendungen und ihre Bedeutung für die Robotik-Entwicklung. In: *Eine Analyse der Fraunhofer Institute IPA und ISI im Auftrag des BMBF, Fraunhofer Gesellschaft* (2011)

[35] HARMON, Leon D.: Automated Tactile Sensing. In: *The International Journal of Robotics Research* 1 (1982), Nr. 2, S. 3–32. – URL http://ijr.sagepub.com/content/1/2/3.short

[36] HORN, Gert van der ; HUIJSING, Johan: *Integrated Smart Sensors: Design and Calibration (The Springer International Series in Engineering and Computer Science)*. 1998. Springer, 12 1997. – ISBN 9780792380047

[37] HOSHI, T. ; SHINODA, H.: A Sensitive Skin Based on Touch-Area-Evaluating Tactile Elements. In: *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2006 14th Symposium on*, march 2006, S. 89 – 94

[38] INSTRUMENTS, Texas: *12/10/8-Bit, 1 MSPS, 16/12/8/4-Channel, Single-Ended, Micro-Power, Serial Interface ADCs*. – URL http://www.ti.com/lit/ds/slas605a/slas605a.pdf

[39] JOHANSSON, R. ; FLANAGAN, J.: *The Senses: A Comprehensive Reference*. Bd. 6: *Tactile Sensory Control of Object Manipulation in Humans*. S. 67–86. In: *Volume 6: Somatosensation* Bd. 6, Elsevier, 2008. – URL http://dx.doi.org/10.1016/b978-012370880-9.00346-7

[40] JOHANSSON, R.S. ; WESTLING, G.: Roles of glabrous skin receptors and sensorimotor memory in automatic control of precision grip when lifting rougher or more slippery objects. In: *Experimental Brain Research* 56 (1984), Nr. 3, S. 550–564. – URL http://dx.doi.org/10.1007/BF00237997. – ISSN 0014-4819

[41] KAMPMANN, Peter ; KIRCHNER, Frank: A Tactile Sensing System for Underwater Manipulation. In: *Proceedings of the workshop on: Advances in Tactile Sensing and Touch based Human-Robot Interaction to be held in conjunction with the 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI 2012)*, o.A., 3 2012

[42] KELM, Hans-Joachim: *USB 2.0: Studienausgabe*. 4. Franzis, 2006. – ISBN 9783772372902

[43] KERPA, O. ; WEISS, K. ; WORN, H.: Development of a flexible tactile sensor system for a humanoid robot. In: *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on* Bd. 1, Oct 2003, S. 1–6 vol.1

[44] KOLBE, Felix: *Einsatz des ROS-Frameworks zur Umgebungskartierung und Navigation des SCITOS G5-Service-Roboters*. 2011

[45] KOLBE, Felix: *Goal Oriented Task Planning for Autonomous Service Robots*. Berliner Tor 5, 20099 Hamburg, HAW Hamburg, Diplomarbeit, 2013

[46] KRISHNA, G.M. ; RAJANNA, K.: Tactile sensor based on piezoelectric resonance. In: *Sensors, 2002. Proceedings of IEEE* Bd. 2, 2002, S. 1643 – 1647 vol.2

[47] KYBERD, P J. ; CHAPPELL, P H.: Characterization of an optical and acoustic touch and slip sensor for autonomous manipulation. In: *Measurement Science and Technology* 3 (1992), Nr. 10, S. 969. – URL http://stacks.iop.org/0957-0233/3/i=10/a=005

[48] MAHESHWARI, Vivek ; SARAF, Ravi: Tactile Devices To Sense Touch on a Par with a Human Finger. In: *Angewandte Chemie International Edition* 47 (2008), Nr. 41, S. 7808–7826. – URL http://dx.doi.org/10.1002/anie.200703693. – ISSN 1521-3773

[49] MEISEL, Andreas: http://www.informatik.haw-hamburg.de/meisel.html. – zuletzt abgerufen am 2015.01.29

[50] MICROCHIP: *PIC32MX1XX/2XX: Section 6. Oscillators*. – URL http://ww1.microchip.com/downloads/en/DeviceDoc/61112H.pdf

[51] MICROCHIP: *Release Notes for the MPLAB® XC32 C/C++ Compiler for PIC32 MCUs*. 6 2015. – URL http://ww1.microchip.com/downloads/en/DeviceDoc/xc32-v1.40-release-notes.html

[52] NICHOLLS, Howard R.: *Advanced Tactile Sensing for Robotics*. River Edge, NJ, USA : World Scientific Publishing Co., Inc., 1992. – ISBN 9810208707

[53] NILSSON, M.: Tactile sensing with minimal wiring complexity. In: *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on* Bd. 1, 1999, S. 293–298 vol.1. – ISSN 1050-4729

[54] NXP: *The I2C-bus specification*, January 2014. – URL http://www.nxp.com/documents/user_manual/UM10204.pdf

[55] OTT, H. W.: Partitioning and Layout of a Mixed Signal PCB. In: *Printed Circuit Design* (2001), Jun. – URL http://www.hottconsultants.com/pdf_files/june2001pcd_mixedsignal.pdf

[56] P. J. STOFFREGEN, R. C.: https://www.pjrc.com/teensy/. – zuletzt abgerufen am 2015.05.23

[57] P. J. STOFFREGEN, R. C.. – URL https://www.pjrc.com/teensy/benchmark_usb_serial_receive.html. – zuletzt abgerufen am 2015.09.01

[58] PAPAKOSTAS, T.V. ; LIMA, J. ; LOWE, M.: A large area force sensor for smart skin applications. In: *Sensors, 2002. Proceedings of IEEE* Bd. 2, 2002, S. 1620 – 1624 vol.2

[59] PRESSURE PROFILE SYSTEMS, Inc.: *Online Verfügbar*. 2007. – URL http://www.pressureprofile.com/products-robotouch. – Zugriffsdatum: 2015-09-30

[60] QUIGLEY, Morgan ; CONLEY, Ken ; GERKEY, Brian P. ; FAUST, Josh ; FOOTE, Tully ; LEIBS, Jeremy ; WHEELER, Rob ; NG, Andrew Y.: ROS: an open-source Robot Operating System. In: *ICRA Workshop on Open Source Software*, 2009

[61] R. WICHERT, T. Norgall M. B.: *Individuelle Gestaltung und Anpassung bestehender Wohnkonzepte*. january 2009. – URL http://www.aal.fraunhofer.de/publications/AAL_Individuelle-Gestaltung_Wohnkonzepte_final.pdf

[62] ROBERTS, L. ; SINGHAL, Girish ; KALIKI, R.: Slip detection and grip adjustment using optical tracking in prosthetic hands. In: *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, Aug 2011, S. 2929–2932. – ISSN 1557-170X

[63] ROLAND, Michael: *USB Storage Device Class - Überblick und Einbindung in AVR AT90USB1287*, Fachhochschule Oberösterreich, Bachelor Thesis, March 2007

[64] ROMANO, Joe: *ROS WIKI: pr2 gripper sensor action*. 2014. – URL http://wiki.ros.org/pr2_gripper_sensor_action. – Zugriffsdatum: 2015-09-01

[65] ROS WIKI 2013b: *ROS WIKI: ROS/Concept*. 2013. – URL http://wiki.ros.org/ROS/Concepts. – Zugriffsdatum: 2015-01-20

[66] ROS WIKI 2013c: *ROS WIKI: ROS/Introduction*. 2013. – URL http://wiki.ros.org/ROS/Introduction. – Zugriffsdatum: 2015-01-20

[67] RUSSELL, R.A.: *Robot tactile sensing*. Prentice Hall, 1990. – ISBN 9780137815920

[68] SALATHÉ, Jan J.: *Dynamische KonVguration mit AVB im TTEthernet*. 2014. – URL http://core.informatik.haw-hamburg.de/images/reports/sal-dkavbtte-13a.pdf

[69] SCHIESSLE, Edmund: *Sensortechnik und Meßwertaufnahme*. Vogel Business Media, 1992. – ISBN 9783802304705

[70] SCHLEICHER, Manfred: *Digital Interfaces and Bus Systems: Fundamentals and practical advice for the connection of field devices*. JUMO, 6 2005. – ISBN 9783935742030

[71] SCHMIDT, Peer A. ; MAËL, Eric ; WÜRTZ, Rolf P.: A sensor for dynamic tactile information with applications in human-robot interaction and object exploration. In: *Robot. Auton. Syst.* 54 (2006), Dezember, Nr. 12, S. 1005–1014. – URL http://dx.doi.org/10.1016/j.robot.2006.05.013. – ISSN 0921-8890

[72] SCHMIDT, Peer A. ; MAËL, Eric ; WÜRTZ, Rolf P.: A sensor for dynamic tactile information with applications in human–robot interaction and object exploration. In: *Robotics and Autonomous Systems* 54 (2006), Nr. 12, S. 1005 – 1014. – URL http://www.sciencedirect.com/science/article/pii/S0921889006001138. – ISSN 0921-8890

[73] SCHMIDT, Robert F. (Hrsg.) ; SCHAIBLE, Hans-Georg (Hrsg.) ; BIRBAUMER, Niels (Hrsg.) ; BRAITENBERG, V. (Hrsg.) ; BRINKMEIER, H. (Hrsg.) ; DUDEL, J. (Hrsg.) ; EYSEL, U. (Hrsg.) ; HANDWERKER, H.O. (Hrsg.) ; HATT, H. (Hrsg.) ; ILLERT, M. (Hrsg.) ; JÄNIG, W. (Hrsg.) ; KUHTZ-BUSCHBECK, J.P. (Hrsg.) ; RÜDEL, R. (Hrsg.) ; SCHAIBLE, H.-G. (Hrsg.) ; SCHMIDT, R.F. (Hrsg.) ; SCHÜTZ, A. (Hrsg.) ; ZENNER, H.P. (Hrsg.): *Neuro- und Sinnesphysiologie (Springer-Lehrbuch)*. 5., neu bearb. Aufl. Springer, 9 2005. – URL http://link.springer.com/book/10.1007%2F3-540-29491-0. – ISBN 9783540257004

[74] SCHMIEDECKE, Christoph: *Aktor-Sensor-Simulation für Assistenzroboter*. Berliner Tor 5, 20099 Hamburg, HAW Hamburg, Diplomarbeit, 2013

[75] SCHOEPFER, Matthias ; SCHUERMANN, Carsten ; PARDOWITZ, Michael ; RITTER, Helge: Using a Piezo-Resistive Tactile Sensor for Detection of Incipient Slippage. In: *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, June 2010, S. 1–7

[76] SCHOPFER, M. ; RITTER, H. ; HEIDEMANN, G.: Acquisition and Application of a Tactile Database. In: *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, S. 1517–1522. – ISSN 1050-4729

[77] SCHUNK GmbH (Veranst.): *PG 70 Manual*. 2010

[78] SCHUNK GmbH (Veranst.): *LEICHTBAUMODUL PRL*. 2011. – URL http://www.schunk.com/schunk_files/attachments/OM_AU_PRL__DE.pdf. – zuletzt abgerufen am 2015.04.23

[79] SCHURMANN, C. ; KOIVA, R. ; HASCHKE, R. ; RITTER, H.: A modular high-speed tactile sensor for human manipulation research. In: *World Haptics Conference (WHC), 2011 IEEE*, June 2011, S. 339–344

[80] SCHWEGLER, Johann S. ; LUCIUS, Runhild: *Der Mensch - Anatomie und Physiologie: Schritt für Schritt Zusammenhänge verstehen*. 5., überarbeitete Auflage. Thieme, Stuttgart, 7 2011. – ISBN 9783131001559

[81] SCHÜRMANN, Carsten ; HASCHKE, Robert ; RITTER, Helge: Modular high speed tactile sensor system with video interface. In: *Tactile sensing in Humanoids—Tactile Sensors and beyond IEEE-RAS Conference on Humanoid Robots*, 2009

[82] SCHÜRMANN, Carsten ; SCHÖPFER, Matthias ; HASCHKE, Robert ; RITTER, Helge: A High-Speed Tactile Sensor for Slip Detection. In: PRASSLER, Erwin (Hrsg.) ; ZÖLLNER, Marius (Hrsg.) ; BISCHOFF, Rainer (Hrsg.) ; BURGARD, Wolfram (Hrsg.) ; HASCHKE, Robert (Hrsg.) ; HÄGELE, Martin (Hrsg.) ; LAWITZKY, Gisbert (Hrsg.) ; NEBEL, Bernhard (Hrsg.) ; PLÖGER, Paul (Hrsg.) ; REISER, Ulrich (Hrsg.): *Towards Service Robots for Everyday Environments* Bd. 76. Springer Berlin Heidelberg, 2012, S. 403–415. – URL http://dx.doi.org/10.1007/978-3-642-25116-0_27. – ISBN 978-3-642-25115-3

[83] SHIMOJO, M. ; NAMIKI, A. ; ISHIKAWA, M. ; MAKINO, R. ; MABUCHI, K.: A tactile sensor sheet using pressure conductive rubber with electrical-wires stitched method. In: *Sensors Journal, IEEE* 4 (2004), oct., Nr. 5, S. 589 – 596. – ISSN 1530-437X

[84] SHINODA, H. ; ASAMURA, N. ; HAKOZAKI, M. ; WANG, Xinyu: Two-dimensional signal transmission technology for robotics. In: *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on* Bd. 3, Sept 2003, S. 3207–3212 vol.3. – ISSN 1050-4729

[85] SHINODA, H. ; OASA, H.: Wireless tactile sensing element using stress-sensitive resonator. In: *Mechatronics, IEEE/ASME Transactions on* 5 (2000), Sep, Nr. 3, S. 258–265. – ISSN 1083-4435

[86] SLADE, Mel ; JONES, Mark H. ; SCOTT, Jonathan B.: Choosing the right microcontroller: A comparison of 8-bit Atmel, Microchip and Freescale MCUs / Faculty of Engineering, The University of Waikato. 2011. – Forschungsbericht

[87] SOHGAWA, Masayuki ; HIRASHIMA, Daiki ; MORIGUCHI, Yusuke ; UEMATSU, Tatsuya ; MITO, Wataru ; KANASHIMA, Takeshi ; OKUYAMA, Masanori ; NOMA, Haruo: Tactile sensor

array using microcantilever with nickel–chromium alloy thin film of low temperature coefficient of resistance and its application to slippage detection. In: *Sensors and Actuators A: Physical* 186 (2012), Nr. 0, S. 32 – 37. – URL http://www.sciencedirect.com/science/article/pii/S0924424712000076. – Selected Papers presented at Eurosensors {XXV} Athens, Greece, 4-7 September 2011. – ISSN 0924-4247

[88] SOMEYA, Takao ; SEKITANI, Tsuyoshi ; IBA, Shingo ; KATO, Yusaku ; KAWAGUCHI, Hiroshi ; SAKURAI, Takayasu: A large-area, flexible pressure sensor matrix with organic field-effect transistors for artificial skin applications. In: *Proceedings of the National Academy of Sciences of the United States of America* 101 (2004), Nr. 27, S. 9966–9970. – URL http://www.pnas.org/content/101/27/9966.abstract

[89] Statistisches Bundesamt (Veranst.): *statistisches Jahrbuch 2007, Bevölkerung und Erwerbstätigkeit, Entwicklung der Privathaushalte bis 2025*. Wiesbaden, Germany, 2008

[90] STRUSS, Ben: *3D-Kartierung von halbstatischen Indoor Umgebungen*, Hochschule für Angewandte Wissenschaften, Ausarbeitung, Mai 2013

[91] TREMBLAY, M.R. ; CUTKOSKY, M.R.: Estimating friction using incipient slip sensing during a manipulation task. In: *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, May 1993, S. 429–434 vol.1

[92] VATANI, Morteza ; ENGEBERG, Erik D. ; CHOI, Jae-Won: Force and slip detection with direct-write compliant tactile sensors using multi-walled carbon nanotube/polymer composites. In: *Sensors and Actuators A: Physical* 195 (2013), Nr. 0, S. 90 – 97. – URL http://www.sciencedirect.com/science/article/pii/S0924424713001362. – ISSN 0924-4247

[93] VON HAGEN, William: *The definitive guide to GCC*. Apress, 2006. – URL https://sensperiodit.files.wordpress.com/2011/04/hagen-the-definitive-guide-to-gcc-2e-apress-2006.pdf

[94] WALZ, Jörg-Dieter: *Fraunhofer-Institut für Produktionstechnik und Automatisierung. Jahresbericht 2012*, Mai 2013. – URL http://publica.fraunhofer.de/documents/N-238784.html

[95] WITTGRUBER, Friedrich: *Digitale Schnittstellen und Bussysteme: Einführung für das technische Studium (Studium Technik) (German Edition)*. 2., überarb. u. erw. Aufl. 2002. Vieweg+Teubner Verlag, 10 2002. – ISBN 9783528174361

[96] ZHANG, Ting ; LIU, Hong ; JIANG, Li ; FAN, Shaowei ; YANG, Jing: Development of a Flexible 3-D Tactile Sensor System for Anthropomorphic Artificial Hand. In: *Sensors Journal, IEEE* 13 (2013), Feb, Nr. 2, S. 510–518. – ISSN 1530-437X

[97] ZHANG, Xiaoyou ; LIU, Rongqiang: Slip detection by array-type pressure sensor for a grasp task. In: *Mechatronics and Automation (ICMA), 2012 International Conference on*, Aug 2012, S. 2198–2202

[98] ZHOU, Junwei ; MASON, Andrew: Communication Buses and Protocols for Sensor Networks. In: *Sensors* 2 (2002), Nr. 7, S. 244–257. – URL `http://www.mdpi.com/1424-8220/2/7/244`. – ISSN 1424-8220

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) bzw.§24(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 14. September 2015 Nicolas Bänisch