TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

Diplomarbeit

Schwenksteuerung eines FTIR-Mikroskop-Filters zur Orientierungsgradbestimmung von PP-Proben während kontinuierlicher Zugversuche

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs, eingereicht an der TU Wien, Fakultät für Maschinenwesen und Betriebswissenschaften, von

Hans WEISS

Mat. Nr. 0726202

unter der Leitung von

Senior Scientist Dipl.-Ing. Dr.techn. Thomas Koch

Institut für Werkstoffwissenschaft und Werkstofftechnologie, E308

Wien, Oktober 2021

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass die vorliegende Arbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen von mir selbstständig erstellt wurde. Alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, sind in dieser Arbeit genannt und aufgelistet. Die aus den Quellen wörtlich entnommenen Stellen, sind als solche kenntlich gemacht.

Das Thema dieser Arbeit wurde von mir bisher weder im In- noch Ausland einer Beurteilerin/einem Beurteiler zur Begutachtung in irgendeiner Form als Prüfungsarbeit vorgelegt. Diese Arbeit stimmt mit der von den Begutachterinnen/Begutachtern beurteilten Arbeit überein.

Ich nehme zur Kenntnis, dass die vorgelegte Arbeit mit geeigneten und dem derzeitigen Stand der Technik entsprechenden Mitteln (Plagiat-Erkennungssoftware) elektronisch-technisch überprüft wird. Dies stellt einerseits sicher, dass bei der Erstellung der vorgelegten Arbeit die

hohen Qualitätsvorgaben im Rahmen der geltenden Regeln zur Sicherung guter wissenschaftlicher Praxis „Code of Conduct“ an der TU Wien eingehalten wurden. Zum anderen werden durch einen Abgleich mit anderen studentischen Abschlussarbeiten Verletzungen meines persönlichen Urheberrechts vermieden.

Wittau, 4. Oktober 2021

Stadt und Datum

Unterschrift

Vorwort

Aktuelle Entwicklungen der Kunststofftechnik zeigen einen wachsenden Forschungsdrang innerhalb dieses umfangreichen Gebiets. Zukunftsträchtige Themen wie 3D-Druck und faserverstärkte Kunststoffe führten mich nach meiner Bachelorarbeit, welche von der Herstellung von Faserverbunden handelte, dazu, mich innerhalb meiner Diplomarbeit erneut mit Kunststoffen zu beschäftigen. Persönlich ist es für mich besonders interessant, welche neuen Möglichkeiten die Forschung in Zukunft herbeiführen wird. Mit meiner Arbeit möchte ich hierzu einen Teil beitragen.

In diesem Zusammenhang ist es mir besonders wichtig meinen Dank auszudrücken.
Ein besonderer Dank geht an meinen Betreuer Herrn Dr. Koch, der mir bei Fragen und Problemen immer zur Seite stand. Als besonders hilfreich dabei empfand ich, dass Lösungsvorschläge für immer wieder neu auftretende Probleme schon während des Gesprächs zielführend und präzise dargestellt wurden. So konnten jegliche Missverständnisse erfolgreich umgangen werden.
Herr Koch, ich danke Ihnen, dass Sie mir diese Arbeit ermöglicht und Sie sich immer ausreichend Zeit für mich genommen haben. Besonders werden mir auch die interessanten Gespräche nach den Besprechungen zur Diplomarbeit in Erinnerung bleiben.
Weiters danke ich meinem guten Freund Bernhard Steiner, MSc., der mich bei Fragen zu Programmierung unterstützte und welcher mich bei der Erstellung des DPT-Auswertungstools tatkräftig unterstützte.
Außerdem möchte ich mich bei meinen Eltern und meinen beiden Großmüttern für deren finanzielle Unterstützung, deren Geduld und deren Motivationsleistung während meines gesamten Studiums bedanken.
Abschließend möchte ich mich noch bei meiner Partnerin Sabrina Willert, MA für Ihre Unterstützung als Motivatorin und Lektorin bedanken.

Inhaltsverzeichnis

Abstract

A method for optimizing the characteristics of plastics is stretching. In this thesis, an existing method for determining the degree of orientation of polymers taken from a previous diploma thesis was revised as well as widely automatized. Within this preceding thesis, a tensile testing machine for the stretching of polymer films under an infrared spectroscope was developed. To determine the degree of orientation, it is necessary to perform two measurements of the test specimen oriented perpendicularly to each other. These are undertaken with FTIR spectroscopy using a polarization filter. Previously, these trials were conducted in steps. Thus, the tensile test had to be interrupted for each measurement. To avoid the process of creep with the stepped version of the test execution, measurements were taken during the tensile test and considered as promising. These measurements are not only cumbersome because of the numerous manual adjustments of the polarization filter, but also prone to failure due to possible human error. Thus, a solution for these issues was intended to be developed in this thesis.

For the automatization of the swivel, the development of a drive together with control electronics and software became necessary. It was quintessential to take the cramped space conditions into consideration and to find a way of making the drive detachable so that surrounding devices would not have to be altered for a permanent attachment. The system is built autarkically; an additional computer or rather additional software and a display area are thus bypassed. The construction of the complete swivel mechanism was done interactively, as problems mainly first occurred during the development process. The whole approach is fully documented in this thesis. The basic idea was to realize the swivel control utilizing a simple time switch. As this partially led to problems with synchronicity between the spectroscopy measurements and the realized swiveling procedure, a possibility of communication between the two components has successfully been determined and implemented. The picture taking which was necessary for the evaluation of the stretch was also automatized later in the process and triggered by the control electronics.

Throughout the entire development process, regular trials using the current prototype were undertaken. The evaluation of this data as well as the advancement of the evaluation method are part of this thesis. For instance, an already existing tool for

determining amplitudes, which is necessary for calculating the degree of orientation, has been reprogrammed and enhanced.

Kurzfassung

Eine Methode zur Eigenschaftsoptimierung von Kunststoffen ist die Verstreckung. Innerhalb dieser Arbeit wurde eine bestehende Methode aus einer vorhergehenden Diplomarbeit zur Bestimmung des Orientierungsgrades von Polymeren revidiert und weitgehend automatisiert. Innerhalb der vorangehenden Arbeit wurde eine Zugprüfmaschine für die Verstreckung von Polymerfolien unter dem Infrarotspektroskop entwickelt. Zur Bestimmung des Orientierungsgrades ist es notwendig, zwei mit dem Polarisationsfilter senkrecht aufeinander stehende Messungen des Prüfkörpers mit dem FTIR-Spektroskop aufzunehmen. Bisher wurden diese Versuche gestuft durchgeführt. Das bedeutet, dass der Zugversuch für jede Messung unterbrochen wurde. Um Kriechvorgänge bei der gestuften Versuchsdurchführung zu umgehen, wurden Messungen während des laufenden Zugversuchs erprobt und als vielversprechend befunden. Da diese Messungen aufgrund zahlreicher manuell durchzuführender Einstellvorgänge des Polarisationsfilters sehr aufwändig und, durch menschliches Versagen, fehleranfällig sind, sollte nun in dieser Arbeit eine Lösung für dieses Problem erarbeitet werden.

Für die Automatisierung der Schwenkvorgänge, wurde die Entwicklung eines Antriebes samt Steuerelektronik und Software notwendig. Besonders wichtig dabei war, die beengten Platzverhältnisse zu berücksichtigen und einen Weg zu finden, den Antrieb abnehmbar zu gestalten, sodass umliegende Geräte nicht für die dauerhafte Befestigung verändert werden müssen. Das System ist autark aufgebaut, ein zusätzlicher Computer bzw. zusätzliche Software und Anzeigefläche am Bildschirm wird somit umgangen. Der Bau des gesamten Schwenkmechanismus erfolgte iterativ, da Probleme meist erst im Entwicklungsverlauf auftraten. Die gesamte Herangehensweise ist lückenlos in dieser Arbeit dokumentiert. Die Grundidee sah vor, die Schwenksteuerung mittels einer einfachen Zeitschaltung zu realisieren. Da dies zu Problemen bei der Synchronität zwischen Spektroskopie-Messung und dem durchgeführten Schwenkvorgang führte, wurde erfolgreich eine Möglichkeit zur Kommunikation eruiert und implementiert. Die für die Auswertung der Dehnung nötigen Bilder werden im späteren Verlauf der Entwicklung ebenfalls automatisiert durch die Steuerelektronik ausgelöst.

Während des gesamten Entwicklungsprozesses wurden regelmäßig Versuche mit den aktuellen Prototypen der Schwenksteuerung durchgeführt. Die Auswertung dieser Daten sowie die Weiterentwicklung der Auswertemethode sind ebenfalls Teil dieser Arbeit. So wurde beispielsweise ein bereits vorhandenes Tool zur Bestimmung der Amplituden, welche für die Errechnung des Orientierungsgrades notwendig sind, mit Python neu programmiert und verbessert.

Abbildungsverzeichnis

Formelverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ABS	...	Acrylnitril-Butadien-Styrol
BBP	...	Benzylbutylphthalat, Weichmacher
CFK	...	Carbonfaserverstärkter Kunststoff
CNC	…	Computerized Numerical Control
DBP	...	Dibutylphthalat, Weichmacher
DC	…	Direct Current, Gleichstrom
DEHP	...	Diethylhexylphthalat, Weichmacher
DPT	...	Datenpunkttabelle
EU	...	Europäische Union
fam	…	Orientierungsgrad in der amorphen Phase
fav	…	Orientierungsgrad in der Mischphase
fc	…	Orientierungsgrad in der kristallinen Phase
FE-Analyse	...	Finite-Elemente-Analyse
FFF	…	Fused Filament Fabrication
FTIR	...	Fourier-Transformations-Infrarotspektrometer
GND	…	Ground, Masse
I/O	...	Input/Output
I2C	…	Inter-Integrated Circuit
IDE	…	Integrated Development Environment
NAS	…	Network Attached Storage, Netzwerkfestplatte
PC	...	Personal Computer
PET	...	Polyethylenterephthalat
Poti	…	Potentiometer, verstellbarer Widerstand
PP	...	Polypropylen
PS	...	Polystyrol
PVC	...	Polyvinylchlorid
PWM	…	Pulsweitenmodulation
RST	…	Reset
SCL	…	Serial Clock, Taktleitung
SDA	...	Serial Data, Datenleitung

1. Einleitung - Ziel der Arbeit

Kunststoffe finden sich schon seit langem überall in unserem Alltag wieder. Ihre Allgegenwärtigkeit zeigt wie wichtig dieser Werkstoff in den letzten Jahrzehnten für den Menschen wurde. Obwohl die meisten Kunststoffe hauptsächlich aus nur zwei chemischen Elementen, dem Kohlen- und Wasserstoff bestehen, sind diese vielseitig wie kaum ein anderer Werkstoff. Der Grund für diese Vielfalt liegt in der unterschiedlichen Anordnung der Polymerketten, welche zu völlig unterschiedlichen Eigenschaften führen kann. Das Spektrum reicht von dehnbaren Elastomeren, über harte und spröde Duroplaste, bis hin zu den am häufigsten verwendeten Thermoplasten. Typische Vertreter aus dem Alltag für Elastomere sind beispielsweise Autoreifen und Gummibänder. Werden hitzebeständige Kunststoffe benötigt so bedient man sich meist an Duroplasten. So finden sich Duromere häufig als Gehäuse für elektronische Komponenten wieder. Eines der bekanntesten Duromere ist Epoxidharz. Zusammen mit Kohlenstofffasern und dem nötigen technischen Know-How können daraus kohlenfaserverstärkte Verbundwerkstoffe hergestellt werden. Diese hochpreisigen High-Tech-Werkstoffe finden sich heute überall dort wieder, wo Leichtbau und besonders hohe Steifigkeit priorisiert werden (1). Die Anisotropie der hergestellten Bauteile ist hier ausschlaggebend. Die Möglichkeit, die Eigenschaften der Bauteile richtungsabhängig durch gezieltes Ausrichten der Fasern zu steuern ist für Ingenieure besonders interessant. Ein Rohr, das beispielswiese hauptsächlich auf Zug belastet wird, kann somit durch die Fasern genau auf diese Belastung optimiert werden. In diesem Fall werden die Fasern also hauptsächlich in Längsrichtung zeigen. Genau diese Optimierung macht Verbundwerkstoffe oft zu Leichtbaumaterialen. Stahlbeton ist ein typisches Beispiel dafür, dass Verbundwerkstoffe nicht automatisch mit Leichtbau assoziiert werden dürfen. Aktuell sind CFK-Bauteile meist noch zu teuer für die breite Masse, sodass man diese meist nur von Motorrennsport, Tragflächenbau und teuren Luxusfahrzeugen kennt (2). Der bereits seit Jahren erhältliche BMW i3 kann dabei als Pionier aufgezählt werden. Mit ihm wurde schon früh versucht, mit Leichtbau und reinem Elektroantrieb bei der Öffentlichkeit zu punkten. Bis zu diesem Zeitpunkt wurden CFK-Bauteile hauptsächlich händisch gefertigt. Bei der Herstellung des i3 wurde jedoch auf maschinelle Fertigung der High-Tech-Bauteile gesetzt. Neueste Bestrebungen der Politik lassen erahnen, dass die Zukunft der Automobiltechnik in der

verbrennungskraftmaschinenfreien Technik liegen wird, da ansonsten immer strenger werdende Abgasnormen vermutlich nicht eingehalten werden können. Um Reichweiten elektrisch angetriebener Fahrzeuge zu erhöhen wird es unumgänglich sein, dessen Gewicht möglichst weitgehend zu reduzieren. Dies lässt vermuten, dass Kunststoffe auch in Zukunft eine immer wichtigere Rolle spielen werden. Auch Hobbyisten haben den High-Tech-Werkstoff bereits für sich entdeckt. Sei es im Angelsport, Modellbau oder im Radsport. Besonders mit Hilfe eines Fahrradrahmens aus CFK kann man sich den Vorteil der anisotropen Bauweise gut vorstellen. Mittels FE-Analyse können die am stärksten beanspruchten Bereiche berechnet werden. Diese Bereiche können durch intelligente Ausrichtung der Fasern verstärkt werden. Um dieselben Eigenschaften mit Hilfe eines isotropen Werkstoffs, wie z.B. Aluminium, zu erreichen, müsste die Wandstärke des Rahmens vergrößert werden. Damit einhergehend nimmt auch die Masse des Rahmens zu. Abschließend hierzu muss allerdings erwähnt werden, dass CFK-Bauteile bei stoßartiger Belastung bei vorheriger Beschädigung des Verbunds zu Sprödbrüchen neigen. Diese können plötzlich auftreten und zu schweren Stürzen führen. Deshalb ist es sinnvoll, darauf zu achten, dass Kohlenstofffaserverbunde nicht durch äußere Krafteinwirkung beschädigt werden. Am Rande soll noch erwähnt sein, dass es sich bei Faserverbundwerkstoffen nicht zwangsweise um von Menschen entwickelte Materialien handelt. So gilt die Natur als Urheber dieser Werkstoffklasse. Holz, einer der ältesten Baustoffe der Menschheit zählt ebenfalls zu den Faserverbunden. Der Faserbestandteil Cellulose und die umliegende Matrix Lignin, welche man sich als Kleber der Fasern vorstellen kann, machen Holz bis heute zu einem nützlichen Roh- und Werkstoff in vielen Disziplinen (3). Neueste Entwicklungen machen Holz sogar zum Hightech-Material. So ist es Forschern gelungen durch ein zweistufiges, simples Verdichtungsverfahren, die Eigenschaften so zu verändern, dass das Ergebnis die Festigkeit vieler Metalle übersteigt. Dabei ist der Hightech-Werkstoff bedeutend leichter als seine Gegenspieler (4). Der Einsatz von Holz im Gebäudebau ist aus brandschutztechnischer Sicht mittlerweile problemlos mittels Schutzanstrichen möglich. Beispielsweise befindet sich am Flughafen Wien-Schwechat ein Hangar dessen Grundgerüst aus Holz besteht. Durch eine spezielle Kombination aus Holz und Brandschutzanstrich hält diese Halle einen Vollbrand bis zum Kollaps etwa viermal länger durch als vergleichbare Aufbauten aus Stahl.

Wie bereits erwähnt sind Thermoplaste die am häufigsten verwendeten Kunststoffe. Besonders charakteristisch ist die Eigenschaft der Verformbarkeit unter Temperatureinfluss. Thermoplaste finden sich überall in unserem Alltag wieder. So sind die meisten Kunststoffgehäuse aus ABS hergestellt, ebenfalls ein Thermoplast. Beispiele hierfür sind Gehäuse von Fernsehern, Laptops, Fitnessgeräten, Fensterrahmen und viele mehr. Viele kennen den Versuch aus ihrer Kindheit, bei dem man leere Joghurtbecher, die meist aus Polystyrol hergestellt werden, im Backofen erwärmt. Dabei wird der Becher zu einer planen kreisrunden Platte. Man spricht hierbei vom Memory-Effekt. Die bei der Herstellung des Bechers, durch schnelles Abkühlen, eingefrorenen und gedehnten Verschlaufungen der Makromoleküle gehen durch die Erwärmung zurück in ihre Ausganglage. Der Antrieb hierfür ist die Tendenz zum geringsten Widerstand.

Auch 3D-Drucke im FFF Verfahren werden erst durch Thermoplaste möglich, wobei erwähnt sei, dass der Ursprung des kommerziellen 3D-Drucks in der Stereolithografie liegt. Der 3D-Druck, der mittlerweile auch im Privatbereich leistbar wurde, wird neue, bis jetzt unbekannte Möglichkeiten schaffen. Bereits heute hilft der 3D-Druck Chirurgen dabei Operationen durch Modelle besser vorzubereiten. So werden speziell angefertigte Bohrschablonen verwendet. Auch eigens angefertigte Hüftgelenke werden bereits für den Patienten gedruckt (5). Gipsträger könnten in Zukunft, durch 3D Drucke einen erheblichen Komfortzuwachs genießen. Abbildung 1 zeigt ein Prototyp, der nicht juckt, nicht riecht, wasserfest und nach abgeschlossener Heilung recycelbar ist.

Dass der 3D Druck in Zukunft nicht nur innerhalb der Kunststofftechnik als ernstzunehmende Herstellungsmethode fußfassen könnte, zeigt eine weitere interessante Entwicklung in der Bauindustrie. So könnten in Zukunft Grundmauern mittels eines überdimensional großen Extruders binnen kürzester Zeit aus Beton gedruckt werden. Diese Drucker sollen vor allem in Katastrophengebieten schnell für Obdach sorgen. Abbildung 2 zeigt ein Konzept dieser Maschine.

Abbildung 1: Exoskelett aus dem 3D Drucker (6)

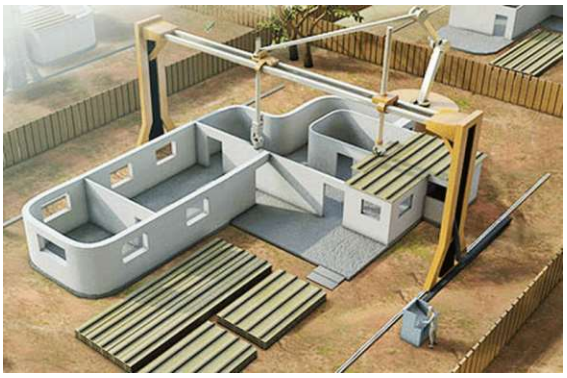


Abbildung 2: Grundmauern aus dem 3D Drucker (7)

Die vielfältige Einsetzbarkeit der Kunststoffe und deren unterschiedlichste Eigenschaften, erwecken den Eindruck uneingeschränkter Möglichkeiten. Die gleichzeitig oft kostengünstige Massenproduktion macht dieses Material zu einem wirtschaftlich höchst interessanten Gut. Eine sich daraus ergebende belastende Folge für die Umwelt ist naheliegend. In jüngsten Jahren werden Kunststoffe in der Öffentlichkeit häufig negativ in Zusammenhang gebracht. Durch verschmutze Strände, „Mikroplastik“ in den Meeren und in Kunststoffresten verfangene Tiere wird durch die Medien völlig zu Recht auf ein Fehlverhalten hingewiesen. Bis vor kurzem war es üblich Kunststoffbeutel für jeden Einkauf kostenlos zu erhalten. Später wurden geringe Gebühren verrechnet, um die Einkaufenden bewusst vor die Entscheidung zu stellen. Dokumentationen wie „Plastic Planet“ und weitere mediale Berichte bewegen immer mehr Menschen zum Umdenken und zum bewussteren Einkauf. Ein kunststofffreies Leben ist allerdings undenkbar, sind diese doch viel zu allgegenwertig. Jedoch ist es möglich sich besser zu informieren und unnötigen Plastikmüll zu vermeiden. Seit dem Jahr 2016 werden durch den Großkonzern REWE keine Kunststoffbeutel mehr an den Kassen vertrieben (8). Ersetzt wurden diese durch Tragetaschen aus Papier für den Einweggebrauch und durch Stoffbeutel. Weiters sind auch wesentlich robustere Mehrwegbeutel aus Kunststoff erhältlich. Auch andere Anbieter zogen bereits nach. Abbildung 3 zeigt eine Studie aus Deutschland, welche evaluierte wie Einkäufe befördert werden. Während man dies als eine erfreuliche Entwicklung betrachten kann, wird dies das Umweltproblem nicht lösen.

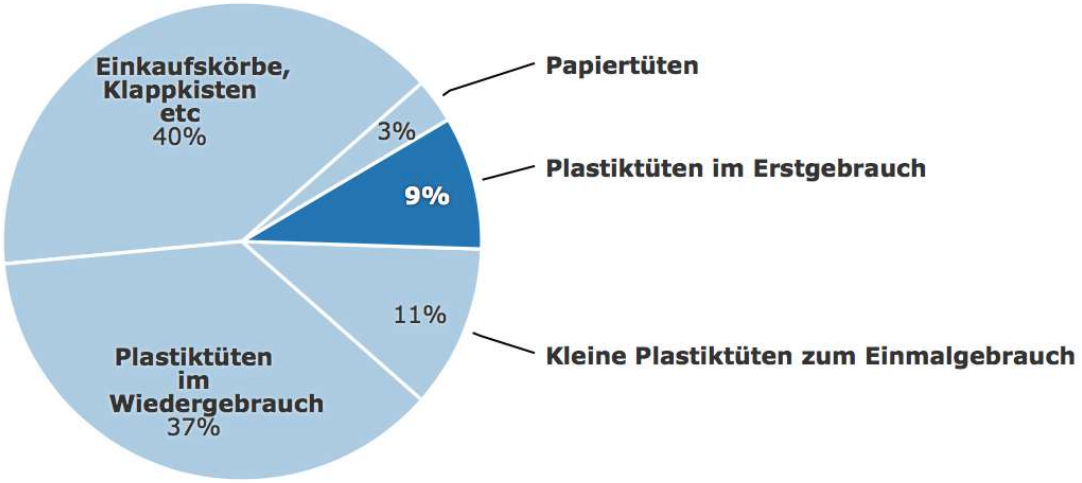


Abbildung 3: Nutzung verschiedenen Tragetaschen, Deutschland (8)

Ökologisch betrachtet steht die Papiertüte für den Einweggebrauch leider nicht besser da als ihr Pendant aus Kunststoff. Man kann dies allerdings auch als Weckruf an die Bevölkerung sehen, der eventuell die kritische Masse zum Umdenken bewegen kann. Um daran mitzuwirken ist es empfehlenswert als Einzelperson auf Einwegprodukte jeglicher Art nach Möglichkeit zu verzichten. Immer mehr Wasser- und Getränkelieferanten bieten ihre Produkte wieder in Glasflaschen an. Beispielsweise hat sich die Vöslauer AG aus genau diesen Gründen zu einer Wiederaufnahme der Glasflasche in das Sortiment entschieden. Wurst, Käse, Fleisch und ähnliche Artikel werden häufig in Verpackungen aus PET verkauft. Um den selbst verursachten Müllberg klein zu halten, können die Produkte beispielsweise am Stück in größeren Mengen gekauft werden. Je nach Möglichkeit ist auch der regionale Einkauf wünschenswert. Ergeben sich diese Möglichkeiten nicht sollte zumindest eine fehlerfreie Mülltrennung stattfinden. Der gelbe Sack ermöglicht es Kunststoffe wie PP, PET und PS durch Recycling erneut nutzbar zu machen.

Neben der umweltbelastenden Auswirkung gelten manche Kunststoffe auch als Krankmacher. Um die Eigenschaften den gewünschten Bedürfnissen anzupassen werden Additive hinzugefügt. Neben Farbpigmenten, Stabilisatoren, Flammschutzmittel und Treibmittel gelten Weichmacher als die am häufigsten zugemengten Stoffe bei der Herstellung von Kunststoffen (9). Viele Weichmacher bzw. Phthalate gelten als hormonell wirksame Chemikalien, krebserregend und toxisch. Pro Jahr werden in der EU etwa eine Million Tonnen Phthalate hergestellt. Neun Zehntel davon werden in der

Produktion für Weich-PVC verwendet. Phthalate werden benötigt, um besonders harte und spröde Kunststoffe weicher und flexibler einzustellen. Typische Beispiel für Produkte in denen Weich-PVC verwendet wird sind Kabelummantelungen, Rohre, Duschvorhänge, Schuhsohlen, KFZ-Bauteile und auch Kinderspielzeuge (10). Die EU-Kommission erteilte für die Herstellung von Kinderspielzeugen bereits ein Anwendungsverbot für die Phthalate DEHP, DBP und BBP, da diese als fortpflanzungsgefährdend eingestuft wurden. Da jedoch 80 Prozent der Spielzeuge importiert werden, muss man davon ausgehen, dass diese Stoffe weiterhin im Umlauf sind. Ersatzprodukte für Weich-PVC um diese zu umgehen sind meist teurer. Ein Beispiel hierfür sind Fußbodenbeläge aus Linoleum oder Holz. Durch wachsendes Gefahrenbewusstsein und Verbote setzt die Industrie nun immer häufiger auf wesentlich teurere, phthalatfreie Weichmacher. Diese gasen weniger aus und gelten als weniger gesundheitsschädlich. Ein weiterer Weg ist der Einsatz höhermolekularer Phthalate. Diese werden durch größere Teilchen schlechter von Körperzellen aufgenommen. Dadurch wird das Gefahrenpotential verringert.

Alle diese Entwicklungen und Probleme zeigen, dass innerhalb der Kunststofftechnik noch ausreichend Forschungsbedarf herrscht. Innerhalb dieser Arbeit wird sich mit der Orientierungsgradbestimmung kontinuierlich mittels Zugversuch verstreckter Polypropylen-Folien beschäftigt. Polypropylen bzw. PP ist einer der am häufigsten verarbeiteten Kunststoffe weltweit, beispielsweise wird dieser für Verpackungen und Gehäuse verwendet.

Diese Arbeit folgert sich aus dem Resümee einer Projektarbeit, die sich mit der Bestimmung des Orientierungsgrades von Polypropylen-Folien während eines kontinuierlichen Zugversuchs beschäftigte. Da die Ergebnisse der Projektarbeit vielversprechend waren, jedoch die Durchführung der Messungen durch ständig nötige, manuelle Eingriffe fehleranfällig war, sollte die Versuchsdurchführung weitgehend - jedoch möglichst kostengünstig - automatisiert werden. Die Versuche wurden zuvor gestuft ausgeführt. Das bedeutet, dass der Zugversuch nach bestimmten Dehnungswerten für die Messung mit dem Infrarotspektroskop angehalten wurde. Da während des Anhaltens relativ starke Relaxationseffekte auftreten, sollte als mögliche Verbesserungsmaßnahme die Messung des Orientierungsgrades während des kontinuierlichen Zugversuchs untersucht werden. Die Frage, warum die Messung zuvor gestuft durchgeführt wurde, ist vorerst völlig legitim und angebracht. Die Antwort

ist allerdings, mit Kenntnis über die Versuchsdurchführung einer Infrarotspektroskop-Messung (im Folgenden auch FTIR-Messung genannt), denkbar einfach. Um den Orientierungsgrad zu bestimmen, werden zwei FTIR-Messungen benötigt, welche mit zueinander senkrecht stehendem, polarisiertem Licht aufgenommen werden. Das bedeutet, dass die Probe beim kontinuierlichen Zugversuch während einer der beiden Messungen weiter verstreckt wird und die beiden zueinander gehörenden Messungen bei unterschiedlichen Verstreckungsstadien aufgenommen werden. Innerhalb der Projektarbeit wurde der Polarisationsfilter während des Zugversuchs nach jeder FTIR-Messung manuell (ca. im 3-Sekunden Takt) geschwenkt. Ist der/die Versuchs-durchführende nicht hochkonzentriert oder seine/ihre Aufmerksamkeit wird an anderer Stelle der Versuchsdurchführung benötigt, so werden die nachfolgenden FTIR-Messungen mit hoher Wahrscheinlichkeit mit falscher Polarisation aufgenommen, weil nicht zum richtigen Zeitpunkt geschwenkt wurde. Da die bereits erwähnten Relaxationsvorgänge innerhalb der Kunststoffproben im gestuften Versuch umgangen werden sollten, ist die kontinuierliche Prüfmethode unumgänglich. Zudem kann der Fehler durch langsame Zugprüfung vernachlässigbar klein gehalten werden. Die Versuche in der Projektarbeit erzielten gute und plausible Ergebnisse. Jedoch folgerte sich auch das Resümee, dass die Versuchsdurchführung durch die Notwendigkeit des manuellen Umschwenkens als fehleranfällig betrachtet werden muss. Das Schluss-wort der Projektarbeit schlägt vor das manuelle Umschwenken des Filters zu automatisieren, hält allerdings keine konkreten Lösungsvorschläge parat. Daraus ergab sich schließlich die Aufgabenstellung dieser Diplomarbeit. (11) (12)

Ziel der Arbeit ist die Entwicklung und der Aufbau einer autonomen Steuerungseinheit samt Schwenkmechanismus. Dabei musste der Mechanismus so konstruiert werden, dass keine Veränderungen an Spektroskop und Filter durchgeführt werden müssen. Außerdem dürfen dabei die eingeschränkten Platzverhältnisse, die sich durch Spektroskop und Zugprüfmaschine ergeben nicht außer Acht gelassen werden. Für Versuche bei denen kein Schwenkmechanismus benötigt wird, sollte dieser problemlos und rasch entfernt werden können. Wünschenswert war eine vom Computer unabhängige Lösung, da bereits je ein PC für Spektroskop und Zugprüf-einrichtung notwendig sind. Es sollte also nach einer adaptiven, autonomen und platz-sparenden Lösung gesucht werden.

Während der Entwicklung der Steuerung wurden Versuche durchgeführt. Dies geschah, um direkt auf Probleme reagieren zu können und um mögliche Verbesserungen direkt diskutieren und umsetzen zu können. Auch die bisherige Methode zur Auswertung der Messdaten sollte analysiert und gegebenenfalls verbessert werden. Während die manuelle Auswertung der Dehnungswerte anhand der Fotos nicht automatisiert werden kann, da diese meist ein geübtes Auge und eine professionelle Einschätzung des Auswerters erfordert, könnten Teile der Auswertung der Spektroskopiedaten automatisiert und optimiert werden. Zur Verarbeitung der Rohdaten des FTIR-Spektroskops wurde bisher ein bereits vorhandenes Tool verwendet. Der DPT-Auswerter wurde mit LabVIEW programmiert und stammt aus der Diplomarbeit von Herrn Dipl. Ing. Kerres (12). Dieses Tool benötigt eine Lizenz und die Installation der Software LabVIEW. Da es bei den Auswertungen mit Hilfe des Tools zu Problemen kam, wurde eine vom Grund auf neue Version erstellt. Dabei wurde auch darauf geachtet frei zugängliche und speicherplatzsparender Software zu verwenden. Das neue Tool wurde grundsätzlich inhaltlich auf Basis des bisherigen Tools entwickelt und in der textbasierten Programmiersprache Python geschrieben. Im Vergleich zur objektbasierten Programmierung in LabVIEW werden, wegen der übersichtlicheren Gestaltung spätere Anpassungen und Verbesserungen durch nachfolgende Nutzer vereinfacht.

Zur Überprüfung der Steuerungseinheit und der Auswertemethode wurde eine abschließende Messreihe durchgeführt und ausgewertet. Die Versuchsergebnisse werden nicht - wie sonst üblich - in einem eigenen Kapitel angeführt, sondern werden im Anschluss der Erklärung zur experimentellen Vorgehensweise aufgezeigt. Ein wesentlicher Teil dieser Arbeit ist die Entwicklung und Optimierung der möglichst weitgehend automatisierten Versuchsdurchführung. Der Entwicklungsprozess, mitsamt allen aufgetretenen Problemen und Schwierigkeiten, wird im Kapitel „Experimentelle Vorgehensweise und Ergebnisse“ beschrieben. Um dies übersichtlicher zu gestalten werden maßgebende Fortschritte in Unterkapiteln gegliedert. Die Versuchsdurchführung ist also als Teil des Ergebnisses zu betrachten und ist dadurch wesentlich mit den tatsächlichen numerischen Ergebnissen und Diagrammen verknüpft. Deshalb werden beide Komponenten zur besseren Übersicht nicht explizit getrennt dargestellt.

Die für diese Diplomarbeit entwickelte Polarisationsfiltersteuerung dient der automatisierten Aufnahme von FTIR-Messungen während des Zugversuchs. Diese spektroskopischen Aufnahmen sind zur Orientierungsgradbestimmung der PP-Folien nach dem Prinzip von *Michler et al* nötig (13). Das Wissen über den Verstreckungsgrad einer Kunststofffolie ist vor allem deshalb interessant, weil dadurch die Leistungsfähigkeit in Bezug auf die Maximalbelastung gesteigert werden kann. Dabei spielt es auch eine Rolle wie verstreckt wird. Während bei einer monoaxialen Verstreckung quer ausschließlich die Nebenvalenzbindungen überwunden werden müssen, können bei der biaxialen Verstreckung Kräfte in jedem Winkel verbessert aufgenommen werden (12).

Durch die spektroskopischen Messungen kann also ermittelt werden, wie weit verstreckt werden muss um einen bestimmten Orientierungsgrad zu erhalten. Dadurch können gewünschte technische Anforderungen gezielt umgesetzt und optimiert werden. Die zu Grunde liegende Vorgehensweise und die dazugehörigen Formeln stammen von *Michler et al* und werden im Folgenden gezeigt (13).

Zur Bestimmung des Orientierungsgrades nach *Michler et al* sind zwei mit senkrecht aufeinander stehenden Polarisationsfiltern aufgenommene Wellenspektren nötig. Dabei ist die erste Aufnahme parallel zur Probenachse und die Zweite quer zur Probenachse aufgenommen. Abbildung 4 zeigt ein typisch aufgenommenes Wellenspektrum aus den absolvierten Versuchen. Für die Ermittlung des Orientierungsgrades werden in weiterer Folge die Absorbtionsbanden innerhalb des Wellenzahlbereichs von 1050 bis 950 cm^{-1} interessant. Zur Errechnung des Orientierungsgrades sind die in Abbildung 4 deutlich erkennbaren Amplituden bedeutend. Laut Literatur treten diese Maxima bei den Wellenzahlen 998 und 974 cm^{-1} auf. Der Peak bei 998 cm^{-1} zeigt den Anteil der kristallinen Phase. Mit dem Maximum bei 974 cm^{-1} wird der Anteil der kristallinen und amorphen Phase angezeigt. Zur Bestimmung der Amplitudenhöhen ist des Weiteren eine Ausgleichgerade zu bestimmen. Diese folgert sich aus den benachbarten Minima der beiden Peaks im Bereich von 1050 bis 950 cm^{-1}, wobei das Minimum zwischen den beiden Maxima ausgeschlossen ist. Die Verbindung dieser beiden Minima ergibt die Ausgleichgerade, deren Höhe an der jeweiligen Stelle des Maximums von der Amplitude abgezogen wird.

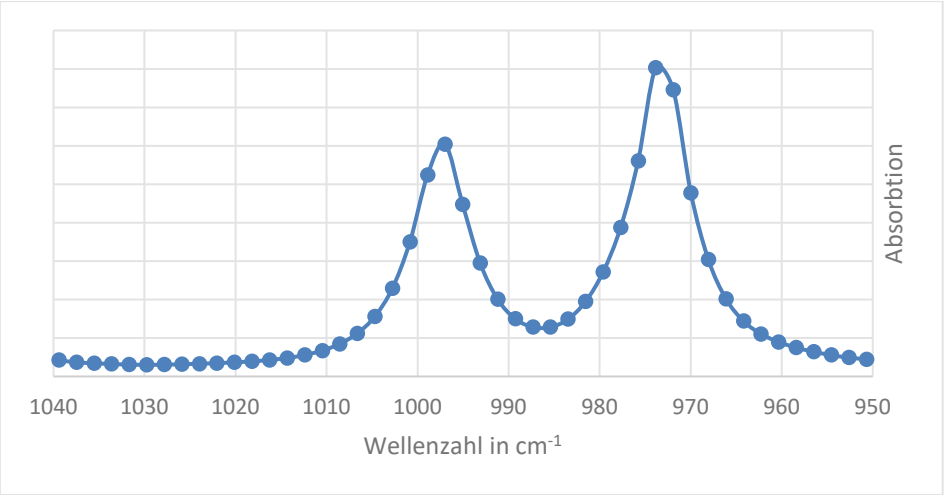


Abbildung 4: Typisches Wellenspektrum bei der Orientierungsgradbestimmung (aus Versuchen zur Probe 22)

Nach der Bestimmung der Amplituden werden mit folgenden formalen Zusammenhängen die Orientierungsgrade nach *Michler et al*. bestimmt.

$$f = \frac{R-1}{R+2} * \frac{R_0+2}{R_0-1}$$

Formel 1: Orientierungsgrad allgemein (13)

$$f_c = \frac{R_{(998)}-1}{R_{(998)}+2} * \frac{R_{0(998)}+2}{R_{0(998)}-1}$$

Formel 2: Orientierungsgrad kristalline Phase

f: Orientierungsgrad
R: Verhältnisse

$$f_{am} = \frac{(f_{av} - X_c f_c)}{(1-X_c)}$$

Formel 3: Orientierungsgrad amorphe Phase (14)

$$f_{av} = \frac{R_{(974)}-1}{R_{(974)}+2} * \frac{R_{0(974)}+2}{R_{0(974)}-1}$$

Formel 4: gemittelter Orientierungsgrad

Xc: kristalliner Anteil der Mischphase

$$R_{(998)} = \frac{A_{\parallel(998)}}{A_{\perp(998)}}$$

Formel 5: Verhältnis $R_{(998)}$ (13)

$$R_{(974)} = \frac{A_{\parallel(974)}}{A_{\perp(974)}}$$

Formel 6: Verhältnis $R_{(974)}$ (13)

A: Peakhöhe

$$R_0 = 2cot^2\psi$$

Formel 7: R_0

Ψ: Winkel zwischen der Achse der Polymerkette und des Übergangsmoments der zu unter-suchenden Absorptionsbanden der Gruppenschwingungen. Für PP wurde dieser Winkel mit 18° ermittelt (14).

$$X_c = \frac{\varepsilon_{0(974)}}{\varepsilon_{0(998)}} * \frac{A_{0(998)}}{A_{0(974)}}$$

Formel 8: Kristalliner Anteil (14)

$$\frac{\varepsilon_{0(974)}}{\varepsilon_{0(998)}} = 0{,}56$$

Formel 9: Verhältnis der Extinktionskoeffizienten (13)

ε: Extinktions-koeffizient

$$A_{0(9xx)} = \frac{A_{\parallel(9xx)} + 2A_{\perp(9xx)}}{3}$$

Formel 10: Strukturelle Absorption (14)

2. Experimentelle Vorgehensweise und Ergebnisse

Im folgenden Kapitel wird auf die Probenpräparation, den Versuchsaufbau, den stückweise verbesserten Versuchsablauf sowie die generelle experimentelle Vorgehensweise eingegangen. Wie schon zuvor erwähnt, werden die Ergebnisse sowie deren Interpretation zur besseren Übersicht in diesem Kapitel angeführt und nicht wie sonst üblich in einem separaten Kapitel.

Für nachfolgende Projekte, die die Verwendung der Zugprüfmaschine Conscindator vorsehen, sollen hier direkt einige Hinweise angeführt werden. Dadurch sollte gewährleistet sein, dass sich dieselben zeitaufwändigen Probleme in Zukunft nicht unnötiger Weise wiederholen müssen. Um die Prüfmaschine nutzen zu können, ist eine lizensierte Version des Programms LabVIEW nötig. In der vorliegenden Arbeit wurde LabVIEW 2012 verwendet. Die Nutzung der Maschine setzt außerdem zwei freie USB-Ports voraus. Die Installation der Software reicht nicht aus, um die Prüfmaschine betreiben zu können. Grund ist das Fehlen eines Motorcontroller-Treibers von Nanotec. Dieser Treiber für auf Windows basierende Rechner kann mittels des Suchbegriffs „SMCI_3x-1_USB“ im Internet gefunden werden, Weiters wird zum Download führender Link im Quellenverzeichnis angeführt (15). Bei der Installation des Treibers müssen beide USB-Anschlüsse der Maschine mit dem Computer verbunden sein. Andernfalls schlägt die Installation fehl. Außerdem muss die Prüfmaschine per 230V Steckdose mit Strom versorgt werden.

Ein weiterer Hinweis bezieht sich auf den Versuchsablauf selbst. Während die Maske des LabVIEW-Programms aktiv ist, kann es unvorhersehbar zu einer Fehlermeldung kommen. Die Meldung bietet zwei Optionen an, „Schließen“ und „Weiter“. Schließt man die Fehlermeldung, so wird in weiterer Folge das ausgeführte LabVIEW-Programm inaktiv. Dabei gehen alle nicht gesicherten, aufgenommen Daten verloren. Außerdem wird die am Beginn erforderliche Kalibrierung der Wägezelle zurückgesetzt. Die Maschine muss dann also erneut kalibriert werden. Um dies zu umgehen, kann die Fehlermeldung mit „weiter“ quittiert werden. Tritt die Meldung während eines Versuchs auf, ist es ratsam möglichst rasch „Weiter“ auszuwählen, da in der

verstreichenden Zeit keine Messwerte aufgenommen werden. Positiv hierbei anzumerken ist, dass die Maschine den Zugversuch ohne Unterbrechung weiterführt. Die besonders wichtigen Messdaten des Spektroskops sind davon also nicht betroffen. Der Ausfall der Messdatenaufnahme der Prüfmaschine resultiert bei der späteren Auswertung im Spannungs-Dehnungs-Diagramm als Stufe und ist deutlich erkennbar. Da diese Daten allerdings nicht weiterverarbeitet werden ist dies nicht weiter problematisch. Weiters ist anzumerken, dass die Auswahl „Schließen“ vorausgewählt ist. Daher darf der Fehler nicht mit Enter quittiert werden, da es sonst zu den bereits erwähnten Folgen kommt. Die Ursache des Fehlers konnte nicht eruiert werden, somit kann hier kein Lösungsvorschlag präsentiert werden und nur auf das oben Genannte hingewiesen werden. Es konnte beobachtet werden, dass die Meldung gewissermaßen zeitlich regelmäßig auftritt, ähnlich als würde ein Timer ablaufen. Daher ist es ratsam eine gegebenenfalls auftretende Fehlermeldung direkt vor dem Start des Versuchs zu quittieren. Bei dieser Vorgehensweise ist es unwahrscheinlich, dass die Meldung erneut und noch während der Versuchszeit auftritt. Erscheint diese danach erneut, so quittiert man diese erst nachdem alle anderen Komponenten des Versuchs vorbereitet sind.

Ein weiterer Hinweis in Bezug auf die Speicherung der Spannungs-Dehnungs-Messwerte wird am Ende des Kapitels 2.1 angeführt.

2.1. Probenherstellung

Zur Herstellung der Proben wird Polypropylen in Form eines Granulats verwendet. Das Granulat wird in eine Matrize gefüllt und diese geschlossen. Das vorbereitete Werkzeug samt Ausgangsmaterial wird danach in einer Heißpresse positioniert. Es folgt ein bereits zuvor erprobter Programmablauf der Presse. Dabei wird das Granulat unter bestimmtem Druck und Temperatur in Form gebracht. Nach dem ebenfalls vorprogrammierten Abkühlprozess wird die Matrize aus der Presse genommen und mittels eines weiteren Werkzeugs geöffnet. Dabei ist darauf zu achten, den Deckel der Matrize vorsichtig zu öffnen, um die entstandene dünne Kunststoffplatte nicht einzureißen. Es folgt eine Begutachtung der hergestellten Polypropylen-Platte. Optimal ist eine homogene, gleichmäßig starke, folienartige, transluzente Platte. Die Herstellung der Folien wurde bereits in der vorangehenden Projektarbeit erprobt und schrittweise verbessert. Anzumerken ist hier, dass die Herstellung zuvor noch zwischen zwei etwa

1 mm starken Metallplatten stattfand. Da dies jedoch nicht immer zu regelmäßig starken Folien führte, wird das Granulat nun mittels Matrize in Form gebracht. Aus der Projektarbeit folgen auch die in Tabelle 1 angeführten Einstellungen für die Heißpresse der Marke Collin.

Tabelle 1: Einstellungen der Heißpresse (11)

Temperatur in °C	30	200	200	200	30
Zeit in min	0	17	7	3	17
Druck in bar	0	5	100	150	100
Aufheiz/Abkühlrate in s/K	0	6	0	0	6

Aus der Tabelle geht folgendes hervor. Zuerst wird auf 30 °C erwärmt. Es folgt eine Aufheizphase auf 200 °C mit 6 s/K und mäßigem Druck bei 5 bar. Die Aufheizphase dauert 17 min. Da die Maschine die Temperatur der Heizplatten misst, muss davon ausgegangen werden, dass die massive Metallmatrize erst später die gewünschte Temperatur erreicht. Dies sollte mit ausreichender Sicherheit genügen, um alle Komponenten auf die voreingestellte Temperatur zu erwärmen. Im folgenden Schritt wird der Druck auf 100 bar erhöht. Diese Phase dauert weitere sieben Minuten. Danach wird der Druck erneut um 50 bar gesteigert und drei Minuten gehalten. Abschließend erfolgt der voreingestellte Abkühlvorgang, der durch die Kühlleitungen der Presse realisiert wird. Analog zur Aufheizphase wird nun mit 6 s/K auf 30 °C abgekühlt. Außerdem wird der Druck um 50 bar reduziert. Um alle Komponenten durchdringend zu kühlen wird erneut eine Gesamtzeit von 17 min für diese Phase eingestellt. Sobald diese Phase abgeschlossen wurde, senkt die Presse den Druck automatisch auf null und fährt in ihre Ausgangsposition zurück. Nun kann die noch warme Matrize mit Handschuhen aus der Presse entfernt werden. Die Tabelle zeigt auch, dass die Dauer innerhalb der Presse summiert 44 min beträgt. Zusätzlich muss die Matrize gereinigt werden, das Granulat mittels Präzisionswaage vorbereitet und die erzeugte Platte aus der Matrize entfernt werden. Gesamt muss ungefähr mit einem Zeitaufwand von 75

min pro erstellter Polypropylen-Platte gerechnet werden. Es empfiehlt sich, die aus der Presse genommene Matrize vor dem Öffnen noch einige Minuten auskühlen zu lassen. Ein erfolgreicher Herstellungsprozess führt zusammen mit 0,5 g PP-Granulat zu einer etwa 0,1 mm starken Folie. Wie bereits erwähnt, kann das unsachgemäße Öffnen der Matrize zu einer Beschädigung der Platte führen. Nach der erfolgreichen Abnahme des Deckels ist es häufig vorgekommen, dass sich das hergestellte Halbzeug nur mit großem Aufwand von der Bodenplatte lösen ließ. Man stelle sich einen oben offenen, hohlen Quader vor. Dieser wird nun mit flüssigem Kunststoff gefüllt bis eine dünne Schicht am Boden entsteht. Lässt man diese Schicht nun auskühlen, erkennt man sofort, dass eine zerstörungsfreie Herausnahme der Platte kaum möglich ist. Zurück zur Matrize. In seltenen Fällen haftet die Platte am Deckel und löst diese damit von der Unterseite. Ist dies nicht der Fall, so wird es nötig die Folie an einer Ecke mittels eines flachen Gegenstands, zum Beispiel mit Hilfe eines Schraubendrehers, zu lösen. Sobald man die Folie vom Untergrund ablösen konnte, wird dies durch die Luft unterhalb der Folie sichtbar. In weiterer Folge wird die Folie langsam und mit möglichst flachem Winkel abgelöst. Passiert dies zu schnell, so könnte die Folie einreißen und man muss neu beginnen. Da die Matrize aus Zeitgründen noch warm ist, verformt sich die Folie, wenn sie in zu steilem Winkel abgezogen wird. Diese rollt sich dann zigarrenförmig und erschwert die weitere Verarbeitung ungemein. Um das Herauslösen der Kunststofffolie wurde eine Idee überprüft. Man stelle sich erneut den Quader vor. Klebt man nun einen Etikettenstreifen in eine Ecke, und zwar so, dass Boden und zumindest eine Wandseite einige mm beklebt sind, so sollte das auf der Wand klebende Etikette als Ablösehilfe dienen. Dies wurde erprobt und stellte sich als zufriedenstellend heraus. Das Auftragen einer PTFE-Schicht mittels eines Sprays wurde bewusst unterlassen, um eine eventuelle Verunreinigung durch den Spray innerhalb der Proben zu vermeiden. In einer späteren Diskussion, jedoch schon nachdem eine ausreichende Anzahl an Proben hergestellt wurde, wurde empfohlen PTFE-Folien zu verwenden. Durch diese dürfte eine Verunreinigung keine Rolle spielen. Für zukünftige Arbeiten wäre dies also durchaus erprobenswert. Dabei sollte es reichen eine PTFE-Folie als Unterlage in die Matrize einzulegen. Die allseits bekannte Antihaftwirkung von PTFE könnte auch dazu führen, dass inhomogene Stellen eine untergeordnete Rolle spielen, da sich der Kunststoff durch wesentlich geringere Reibung besser durchmischen sollte.

Nachdem erfolgreich mehrere dieser Platten hergestellt wurden, werden aus diesen mittels eines scharfen Messers und eines Lineals etwa 5 mm breite und zumindest 80 mm lange Streifen präpariert. Obwohl später nur wenige Millimeter bei der Untersuchung eine Rolle spielen, ist diese Länge für die Einspannung in die Spannbacken der Zugprüfmaschine Conscindator nötig. In weiterer Folge wird mit einem Locheisen, welches einen Durchmesser von 13 mm aufweist, ein Steg bzw. eine Verjüngung der Probe gestanzt. Diese führt dazu, dass die Probe genau in diesem später durch das FTIR-Spektroskop beobachteten Bereich brechen wird. Um möglichst gleiche Verjüngungen der Proben zu realisieren, empfiehlt es sich vorab eine Schablone für das Stanzen anzufertigen. Eine erprobte Methode ist das Stanzen eines starken Blatt Papiers an zwei definierten Stellen. Liegen die beiden Stanzlöcher dann soweit auseinander, dass die erwünschte Stegbreite entstanden ist, so nutzt man diese in weiterer Folge unter der transluzenten Probe als Schablone für die Locheisen. Eine weitere Option ist das Nutzen einer Schablone, die auf die Probe gelegt wird. Diese wurde aus 2 mm starken Plexiglas gefertigt. Wie bei der Papierschablone, wird mittels zweier Stanzungen nebeneinander ein Steg, dessen Breite etwa 2 mm beträgt, angefertigt. Die beschriebene Schablone aus Plexiglas wird in Abbildung 5 gezeigt.
Obwohl diese Schablone professioneller wirkt, zeigte sich, dass diese weniger gut funktioniert. Problem ist hierbei das Verrutschen der Probe unterhalb der Stanzhilfe, zwischen dem ersten und zweiten Schlag. Da die Probe nach dem ersten Schlag verrutscht und nicht fixiert werden kann, muss die Probe erneut ausgerichtet werden. Dies hat sich als zu ungenau erwiesen, weshalb die erste Variante empfohlen wird.

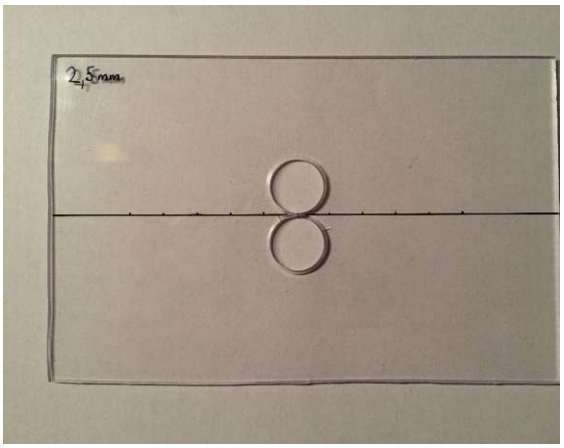


Abbildung 5: Schablone für Stanzwerkzeug, Steg in der Mitte

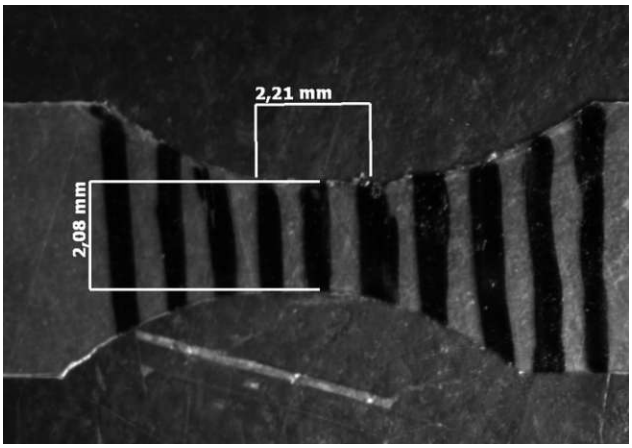


Abbildung 6: Probenmarkierung und verjüngter Bereich

Der nächste Schritt der Probenherstellung sieht das Auftragen einer Markierung im verjüngten Bereich vor. Diese dient später der Bestimmung der Dehnung, welche manuell mittels eines Programms ausgemessen wird. Abbildung 6 zeigt diese mittels wasserfestem, schwarzen Stift aufgebrachte Markierung. Sie zeigt auch, dass die Markierung nicht besonders regelmäßig aufgetragen ist. Es zeigte sich, dass sich unregelmäßige Markierungen später als besser auswertbar herausstellten. Der Grund hierfür ist, dass Unregelmäßigkeiten in der verstreckten Probe besser erkannt werden können. Im letzten Schritt der Probenpräparation werden die Prüfkörper links und rechts nummeriert und mit Hilfe des Programms AxioVision vermessen. Die Proben werden beidseitig nummeriert, um die Bruchteile nach dem Zugversuch einander zuordnen zu können.

Insgesamt wurden im Zuge dieser Arbeit achtundzwanzig Prüfkörper hergestellt. Bei der Herstellung der Kunststoffplatten muss auch mit Ausschuss gerechnet werden, da in manchen Fällen kein homogenes Erscheinungsbild des Erzeugnisses auftrat. Dies äußerte sich in einem trüben, milchigen und schleierartig durchzogenen Ergebnis. Es hat sich gezeigt, dass eine Konzentrierung der einzelnen Granulat-Körner in der Mitte der Matrize häufiger zu einer überwiegend homogenen Struktur führt. Weisen die erzeugten Platten trotz jeglicher Sorgfalt inhomogene Stellen auf, so wird empfohlen die Proben so aus der Folie zu schneiden, dass die Inhomogenität entweder als Rest ausscheidet oder so, dass diese später innerhalb der Einspannung der Zugprüfmaschine liegt. Neben dem Ausschuss der Platten, wurde die Erfahrung gemacht, dass auch mit Verschnitt und mit schlechten Stanzungen gerechnet werden muss. So

kann es durch Verschiebung der Probe unterhalb des Lineals zu ungleichmäßigen Breiten kommen. Es wird empfohlen, das Locheisen möglichst gerade anzusetzen und dann mittels eines kräftigen Hammerschlags die Stanzung durchzuführen. Dabei besteht Verletzungsgefahr, weshalb Schutzhandschuhe zu empfehlen sind. Wichtig ist, die Stanzung mit nur einem Hammerschlag auszuführen. Wird ein zweiter Schlag nötig, führt dies in der Regel zu einem ungleichmäßigem Stanzbild. Beim Stanzen wurden bei der Probenpräparation zwei Probleme beobachtet. Einerseits können, trotz Schablone, zu breite oder zu schmale Stege entstehen. Anderseits kam es auch zu ungleichmäßigen, ausgefransten Stanzungen. Vor allem Zweiteres ist als besonders kritisch zu betrachten, da diese Unregelmäßigkeiten später zu einem verfrühten Bruch der Probe führen können. Diese Störungen haben ähnliche Auswirkungen wie Risse und sollten vermieden werden. Um dies zu vermeiden ist eine möglichst harte Unterlage notwendig. Gleichzeitig sollte auch das Locheisen geschont werden, sodass dieses nicht zu schnell verschleißt. Eine Metallplatte als Untergrund ist also nicht empfehlenswert. Als Kompromiss zwischen möglichst hoher Härte und möglichst geringer Abnutzung des Werkzeugs, wurde eine Plexiglasplatte als Lösung gefunden. Da das Locheisen jedoch bei jedem Schlag auch die Plexiglasplatte beschädigt, ist es nötig die Position der Unterlage stetig geringfügig zu ändern. Damit wird vermieden, dass eine Furche entsteht die erneut zu Ausfransungen der Probe führen würde. Aus fünf ausreichend homogener Kunststofffolien konnten, wie bereits erwähnt, achtundzwanzig Versuchskörper hergestellt werden. Pro Platte konnten also etwa sechs Probekörper angefertigt werden. Beachtet man für zukünftige Probenpräparationen die oben angeführten Empfehlungen, so sollte eine höhere Ausbeute möglich sein. Weiters gäbe es die Möglichkeit fertige Stanzformen zu nutzen, die mit nur einer Stanzung den fertigen Prüfkörper inklusive paralleler Verjüngung erzeugen.

Abbildung 7: Stanzwerkzeug (16)

Abbildung 7 zeigt eines dieser Messer. Es wurde sich aus zwei Gründen gegen die Nutzung eines solchen Werkzeugs entschieden. Zum Ersten führen die breit ausgeführten Enden für die Einspannung zu einer deutlichen Erhöhung des Verschnitts. Dadurch werden deutlich weniger Proben pro Platte möglich. Zum Zweiten scheiterte dies auch an der Verfügbarkeit der passenden Probengeometrie. In Abbildung 7 ist erkennbar, dass der parallel verlaufende Prüfbereich, also jener der später zwischen den Einspannungen liegt, relativ lang ausgeführt ist. Dabei ist nicht vorherzusehen wo die Probe beginnen wird sich beim Zugversuch plastisch zu verformen. Für die Ermittlung des Orientierungsgrades, mittels der Aufnahmen des Infrarotspektroskops ist es allerdings wichtig, dass während der gesamten Messung keine lokale Veränderung stattfindet. Ansonsten könnte man nicht von der Änderung durch den Zugversuch oder der Änderung durch die Verschiebung des Messpunkts unterscheiden. Aus diesem Grund wurde eigens für diese Versuche in einer vorangehenden Diplomarbeit (12) eine Zugprüfmaschine entwickelt, welche die Probenmitte im Fokus des Spektroskops behält. Normalerweise ist einer der beiden Einspannköpfe einer Zugprüfmaschine statisch verankert. Bei diesem Aufbau gelingt es, durch ein Seilzugsystem, beide Spannklemmen parallel und mit gleicher Geschwindigkeit in entgegengesetzte Richtung zu bewegen. Findet man also ein Probenstanzwerkzeug, welches schmale Einspannenden, einen bis zu 5 mm langen parallelen Prüfbereich aufweist und eine Gesamtlänge des Prüfkörpers von 80 mm nicht unterschreitet, so ist dieses Werkzeug vermutlich der anderen Herstellungsmethode vorzuziehen. Obwohl nur wenige Millimeter für eine Untersuchung ausreichen würden, müssen die Spannbacken der Zugprüfmaschine zumindest 50 mm auseinander liegen. Der Grund hierfür sind die baulichen Begebenheiten des Spektroskops und der Zugprüfeinrichtung. Liegen die Spannbacken zu eng beieinander so ist es unmöglich die Prüfmaschine in Position für die FTIR-Messung zu bringen, da der Kondensor des IR-Mikroskops ansonsten den Spannbacken im Weg steht.

Die Dicke der Proben wird mittels einer Messschraube an drei unterschiedlichen Positionen in der Mitte der Probe gemessen und gemittelt. Die anderen beiden Messdaten folgen aus Messungen mittels des Programms AxioVision. Die in der Tabelle angeführte Breite bezieht sich auf die Weite des durch die Stanzung entstandenen Steges. Bei der Ermittlung der parallelen Probenlänge ist es nötig die Bereiche visuell

einzuschätzen. Die beiden gegenüberliegenden Stanzungen lassen durch ihre Kreisform nur näherungsweise einen parallelen Bereich entstehen. Nebenbei bemerkt ist allerdings genau dies für den Versuch auch zweckdienlich, da durch die Rundungen innerhalb der Verjüngung genau eine schmalste Stelle definiert wird. Für die Messung ist es allerdings erforderlich den Bereich mittels Augenmaß einzuschätzen. Da dies auch eine mögliche Fehlerquelle ist, ist es ratsam alle Proben nacheinander zu vermessen bzw. bei späteren Messungen die bereits vermessenen Proben erneut zu betrachten, um möglichst ähnliche Vorgehensweisen zu erzielen. In Abbildung 6 sind beide Schablonen abgebildet. Hier ist auch zu erkennen, dass die beiden Rundungen nicht direkt gegenüberliegen. Während dies für die Versuchsdurchführung keine Rolle spielt, erschwert es das Erkennen des als parallel zu betrachtenden Abschnitts erheblich. Tabelle 2 gibt die ermittelten Abmessungen der Prüfkörper in Millimeter an. Um die Ergebnisse vergleichbar zu halten, sollten die Abmessungen nicht zu weit voneinander differieren. Um dies besser einschätzen zu können wurden als Vergleichswert für künftige Projekte Mittelwerte und Standardabweichungen ermittelt. Diese werden hier gerundet angeführt. Für den näherungsweise parallelen Probenabschnitt errechnet sich der Mittelwert zu 2,23 mm mit einer Standardabweichung von 0,2 mm. Die Spalte der Breite ergibt einen Mittelwert von 2,13 mm mit einer Standardabweichung von 0,26 mm. Der Mittelwert der Probendicke ergibt sich zu 0,103 mm mit einer Standardabweichung von 0,0125 mm. Durch die Abmessungen werden die Proben auf den ersten Blick vergleichbar. Die parallele Probenlänge wird für die Ermittlung der Dehnung benötigt. Die Werte werden für die Versuchsdurchführung in das Bedienfeld der Zugprüfmaschine, welches mittels LabView realisiert wurde, eingegeben. Die Prüfmaschine visualisiert die ermittelten Spannungs- und Dehnungswerte simultan zum Versuch in der LabView-Maske als Diagramm. Nach Beendigung eines jeden Versuchsdurchgangs können bzw. müssen die aufgenommenen Werte in tabellarischer Form gesichert werden. Als Hinweis für nachfolgende Projekte sollte erwähnt werden, dass die Daten nicht automatisch, sondern nach jedem Versuch manuell gesichert werden, da diese ansonsten beim nachfolgenden Versuch überschrieben werden.

Tabelle 2: Probenabmessungen in mm

Proben-nummer	Parallele Länge	Breite	Dicke
1	2,51	2,15	0,103
2	2,52	2,23	0,091
3	2,21	2,08	0,118
4	2,37	2,03	0,104
5	2,3	2,18	0,093
6	2,26	2,14	0,106
7	2,23	1,93	0,106
8	2,25	2,53	0,11
9	1,85	2,01	0,102
10	2,2	2,15	0,101
11	1,98	1,83	0,099
12	2,54	2,65	0,121
13	2,03	1,73	0,122
14	2,34	2,41	0,091
15	2,2	1,75	0,089
16	2,41	2,23	0,077
17	2,43	1,98	0,117
18	2,39	2,21	0,119
19	2,25	1,88	0,082
20	2,08	2,45	0,102
21	1,72	2,23	0,082
22	2,28	2,01	0,114
23	2,03	2,27	0,114
24	2,12	2,32	0,103
25	2,3	2,47	0,105
26	1,95	1,53	0,087
27	2,27	2,14	0,111
28	2,43	2,25	0,11

2.2. Konzeptionierung und Version 1

Im folgenden Kapitel wird auf die grundsätzliche Konzeptionierung zur Umsetzung einer möglichst weitgehenden Automatisierung der Versuchsdurchführung eingegangen. Um die Herangehensweise schildern zu können, muss zuerst der Versuchsaufbau erklärt werden. Der bisherige Versuchsaufbau sieht vier Hauptkomponenten vor. Das FTIR-Spektroskop der Marke Bruker inklusive verbundenem

Computer mit Software. Die Zugprüfmaschine Conscindator mit Computer und LabVIEW zur Steuerung. Eine Spiegelreflexkamera mit passendem Stativ, ausreichender Vergrößerung und Blitz. Aus Platzgründen sollte der PC zur Steuerung der Prüfmaschine als Laptop ausgeführt sein. Eine Installation der Steuerungssoftware für den Conscindator auf dem Rechner des Spektroskops ist aus administrativen und sicherheitsrelevanten Gründen nicht möglich. Des Weiteren könnte das auf diesem Computer verwendete, seit längerem nicht mehr unterstützte, Betriebssystem Windows XP für Probleme mit Prüfmaschine und Software führen. Abbildung 8 zeigt den Versuchsaufbau.

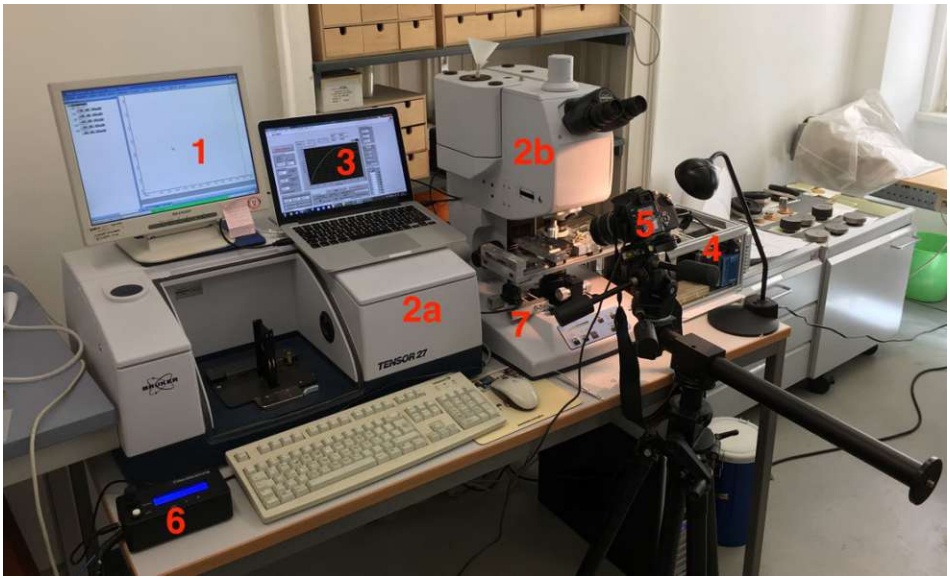


Abbildung 8: Versuchsaufbau

Die Markierung Nr. 1 zeigt das Display des Computers, der für die Steuerung des FTIR-Spektroskops notwendig ist. Hier werden sämtliche das Spektroskop betreffende Einstellungen mittels der Software OPUS 6.0 vorgenommen. Die verwendeten Settings wurden für spätere Versuche als Profile gespeichert und separat gesichert. Dies ist auch für künftige Versuche empfehlenswert. Die Markierung Nr. 2a und 2b zeigen das Spektroskop selbst. Nummer 3 zeigt das für die Zugprüfmaschine notwendige Notebook zu Steuerung. Die Prüfmaschine ist mit der Nr. 4 gekennzeichnet, während die Kamera samt massivem Stativ nachfolgend mit der Nr. 5 markiert ist. Die Markierung Nr. 6 zeigt bereits vorgreifend das in dieser Arbeit entwickelte Steuerungsgerät der automatischen Schwenkeinrichtung des Polarisationsfilters. Markierung Nr. 7 zeigt den Schwenkmechanismus. Wie im Bild leicht zu erkennen ist, musste die

Steuerung und die Schwenkeinrichtung aus platztechnischen Gründen voneinander getrennt realisiert werden. Ebenfalls im Bild erkennbar, allerdings nicht zwingend notwendig, ist eine Schreibtischlampe welche auf die Probe gerichtet ist. Um einer Verfälschung durch Erwärmung zu umgehen sollte diese ausschließlich während des Versuchs verwendet werden. Alternativ wäre auch die Nutzung einer Kaltlichtlampe möglich. Ist eine Lampe in Verwendung, so kann der Blitz der Kamera deaktiviert werden. Dies führte zu einem besseren Bildergebnis, da es zu keinen Reflexionen auf den umliegenden Metallteilen kam.

Die Bestimmung des Orientierungsgrades mittels Infrarotspektroskopie während des Zugversuchs, also ohne Stillstand während der Messung, wurde, wie bereits erwähnt, bereits in einer Projektarbeit untersucht. Da der Polarisationsfilter manuell für jede Messung des Spektroskops geschwenkt werden musste, sollte nun also eine praktischere, automatische Lösung gefunden werden.

Für die erste Version der Steuerungseinheit wurde angenommen, dass die Messungen des Spektroskops einem definierten Intervall zu Grunde liegen. Folglich würde einer Messung, die 3 Sekunden in Anspruch nimmt, eine weitere Messung mit derselben benötigten Zeit folgen. Hier sollte nochmals angemerkt werden, dass für die Bestimmung des Orientierungsgrades mehrere FTIR-Messungen hintereinander, während des Zugversuchs und mit schwenkendem Polarisationsfilter aufgenommen werden müssen. Um dieses Messintervall für die vorgenommenen Einstellungen des Spektroskops also bestimmen zu können, wurden mehrere Versuche durchgeführt und als Video für die spätere Auswertung aufgenommen. Dabei wurden die Voreinstellungen des Spektroskops variiert. Ziel der Variation ist das Finden einer möglichst kurzen Messdauer unter gleichzeitiger Berücksichtigung der Messqualität. Der Grund für die kleinzuhaltende Messdauer ist die Versuchsdurchführung während des Zugversuchs, bei der sich die Probe während der Messung weiterbewegt. Um den Fehler gering zu halten müssen zwei zueinander gehöhrende Messungen für die Bestimmung des Orientierungsgrades zeitnah aufgenommen werden. Im Idealfall würde dies gleichzeitig geschehen, dann würde sich der hiervon ausgehende Fehler auf null reduzieren.

Innerhalb der Einstellungsmaske des Infrarotspektroskops wurden vier Parameter abgeändert. Zum Ersten können die Messdauer oder die Anzahl der Scans festgelegt werden. Da in der Regel Messungen an statischen Objekten durchgeführt werden und

dabei zu Gunsten der Qualität länger gemessen wird, wird die Messdauer in Minuten eingegeben. Wird eine Messzeit festgelegt so führt die Maschine solange Scans hintereinander durch bis die vorgegebene Zeit verstrichen ist. Aus den einzelnen Scans einer Messung errechnet die Software des Spektroskops automatisch ein gemitteltes Ergebnis. Eine Steigerung der Anzahl der Scans erhöht die Messqualität für statische Versuche. Da mit steigender Scanzahl allerdings auch die Messdauer steigt und während des Zugversuchs dynamisch geprüft wird, kann die Anzahl nicht beliebig erhöht werden. Eine erhöhte Messdauer bedeutet auch einen höheren Verfahrensweg. Dadurch werden unterschiedliche Probenzustände aufgenommen, welche nicht als eine Messung zusammengefasst werden sollten, da dies der Genauigkeit entgegenwirkt. Als nächster Parameter wird die Dauer der Pausen zwischen den Messungen bestimmt. Bei den letzten beiden variierten Parametern handelt es sich um die obere und untere Schranke der Wellenzahl. Für die Auswertung ist der Bereich zwischen 950 und 1050 cm^{-1} interessant. Da jedoch die Parametervariation zeigte, dass sich die Aufnahme eines größeren Bereichs nicht messbar auf die Messdauer auswirkte, entschloss man sich den Messbereich zwischen 600 und 2000 cm^{-1} festzulegen. Auch wenn diese Daten zurzeit nicht verwendet werden, so ist es nicht abschätzbar, ob diese für neu erdachte Methoden sinnvoll wären. Die aufgenommene Datenmenge des FTIR-Spektroskops hält sich mit wenigen Megabyte pro Versuch in Grenzen und spricht damit nicht gegen die Aufnahme eines größeren Bereichs als eigentlich nötig. Insgesamt wurden zehn dieser Videos bzw. Einstellungsvariationen aufgenommen und ausgewertet.

Um den Text verständlicher zu gestalten folgt eine Erklärung rund um die Benutzung des Wortes „Messung“ im Zusammenhang mit dem Spektroskop. Die dadurch eingeführte Nomenklatur soll dem besseren Verständnis dienen. Wie aus dem vorigen Absatz hervorgeht, besteht eine Messung aus mindestens einem Scan. Zur Bestimmung des Orientierungsgrades werden zwei zeitlich direkt aufeinander folgende Messungen im 0 und 90 Grad Winkel benötigt. Im Folgenden werden diese beiden zusammengehörigen Messungen Orientierungsgradmessung genannt. Für eine Probe werden während des Zugversuchs mehrere Orientierungsgradmessungen hintereinander aufgenommen - dies wird als „Versuch“ bezeichnet.

Die Auswertung der Videos führte, neben der Festlegung des Bereichs der Wellenzahl, zu folgenden Erkenntnissen. Wie bereits erwähnt, kann für die Dauer pro Messung

zwischen der Angabe einer Messdauer und der Anzahl der Scans gewählt werden. Bei der Erstellung des Videomaterials wurde das Statusfeld der Software OPUS 6 aufgenommen. Dieses Feld zeigt die aktuelle Scannummer und beginnt bei der nächsten Messung wieder bei null. Durch diese Anzeige ist es möglich, die Messdauer pro Messung festzulegen, in dem man die Zeiten im Video zwischen zwei Starts von null ausgehend misst. Pro Video wurden 20 Messungen durchgeführt. Für die Auswertung wurden die gemessenen Zeiten gemittelt und verglichen. Dabei stellte sich heraus, dass die Zeiten pro Messung auch bei gleichbleibender Einstellung nicht konstant sind. Dies gilt auch für die Messungen innerhalb eines Versuchs. Die Einstellungen mit den geringsten zeitlichen Abweichungen wurden mit einer Messdauer von 0,05 Minuten (also 3 Sekunden) und einer Pause von 0 Sekunden ermittelt. Bei diesen Settings wurde eine Abweichung von 0,05 Sekunden ermittelt. Andere Einstellungen führten zu einer Abweichung von bis zu 0,8 Sekunden. Tabelle 3 zeigt die ermittelten Werte. Außerdem ist zu erwähnen, dass die erste Messung bei fast allen Messungen deutlich mehr Zeit in Anspruch nahm als die folgenden, weshalb dieser Wert für die Ermittlung des Durchschnittswertes ausgeschlossen wurde.

Tabelle 3: Auswertung des Videomaterials zur Bestimmung der Messintervalle, Werte in Sekunden

Zeile	Messzeit/ Anzahl der Scans	Pause	Wellen-zahl in cm^{-1}	Durchschn. Zeit-abstand	Abweichung zum Durch-schnittswert	1. Zeit-abstand
1	0,05 min	0	2000 - 600	3,14	0,05318	6,47
2	0,05 min	1	2000 - 600	5,91	0,16321	5,86
3	0,1 min	0	2000 - 600	6,30	0,22930	11,73
4	3 Scans	0	2000 - 600	3,13	0,24686	5,2
5	1 Scan	1	2000 - 600	3,20	0,51047	4,5
6	3 Scans	1	2000 - 600	5,41	0,58703	6,9
7	2 Scans	1	2000 - 600	5,30	0,67396	5,33
8	2 Scans	1	1200 - 800	4,84	0,78193	6,56
9	0,1 min	1	2000 - 600	9,80	0,78639	2,63
10	3 Scans	1	1100 - 900	5,38	0,81966	8,67

Die Tabelle ist nach der durchschnittlichen Abweichung von klein nach groß geordnet. Die Einstellungen mit 0,05 Minuten Messdauer und 0 Sekunden Pause führt also zum

Ergebnis mit geringster Messdauer und kleinster Abweichung. Die Tabelle offenbart außerdem, dass die eingestellten Messzeiten nicht exakt eingehalten werden. Für die Messungen, bei denen eine Messdauer eingestellt wurde fällt auf, dass die tatsächlichen Werte immer darüber liegen. So würde aus der ersten Zeile von Tabelle 3 hervorgehen, dass die Zeit pro Messung 3 Sekunden beträgt. Tatsächlich wird ein durchschnittlicher Wert von 3,14 Sekunden benötigt. Verdoppelt man nun die eingestellte Zeit, so verdoppelt sich auch die tatsächliche Zeit. Dies ist im Vergleich von Zeile 1 zu Zeile 3 erkennbar. Ebenfalls interessant ist der Vergleich zwischen Zeile 6 und 7. Obwohl für den Versuch aus Zeile 6 ein Scan mehr durchgeführt wurde, folgt eine nahezu idente Messdauer. Reduziert man die Anzahl der Scans auf 1 (siehe Zeile 5) so verringert sich die Messdauer um etwa 2 Sekunden im Vergleich zur Messung mit zwei Scans. Eine weitere mögliche Schlussfolgerung ist, dass Versuche mit definierter Messdauer zu Ergebnissen mit geringerer Abweichung vom Durchschnittswert führen, da diese die ersten drei Ränge der Tabelle belegen. Versuch Nummer 9 fällt hier allerdings aus der Reihe und stellt auch diese Annahme in Frage. Letztendlich zeigt der Vergleich zwischen Zeile 6 und 10, dass sich die Veränderung des Bereichs der Wellenzahl unmaßgeblich auf die Messdauer auswirkt.

Die Diskussion zeigt, dass die Vorhersage der Dauer für eine Messung schwierig ist, da die ermittelten Werte als nicht linear zu betrachten sind. Die besten Ergebnisse für geringe Messdauer bei gleichzeitig möglichst geringer Abweichung der Messzeit ergaben die Einstellungen aus Zeile 1 der Tabelle. Um allerdings nicht während des Schwenkvorgangs bereits eine neue Messung zu starten sollte eine Pause festgelegt werden. Das Drehen des Polarisationsfilters mittels Servomotor dauert etwa 0,5 Sekunden. Die definierte Pause sollte also nicht unter dieser Zeitspanne liegen. Offen bleibt aus welchem Grund eine voreingestellte Messdauer mit eventuell definierter Pause nicht zu einer konstanten und vorhersehbaren Gesamtzeit führt. Durch die Auswertung konnte ebenfalls erkannt werden, dass auch einzelne Scans nicht exakt gleich lange dauern. Die Abspeicherung der Daten im Hintergrund nach jeder Messung könnte eine mögliche Erklärung sein. Die Abweichung der tatsächlichen Messzeit von der voreingestellten Zeit könnte sich daraus ergeben, dass auch hier Scans hintereinander durchgeführt werden. Der letzte Scan würde somit in den meisten Fällen vor Ablauf der definierten Zeit beginnen. Geht man davon aus, dass die Software diesen Scan noch beendet bzw. nicht abbricht, so würde sich eine Überschreitung der Dauer

hiermit erklären. Die hiermit gewonnen Daten legen das Intervall des Schwenkmechanismus fest.

Zeitgleich wurde ein Grundkonzept zur Umsetzung des Mechanismus entwickelt. Ausgehend vom automatisch zu schwenkendem Polarisationsfilter sollte unter Berücksichtigung der eingeschränkten Platzverhältnisse ein Gerät entworfen werden, welches primär exakte Schwenkvorgänge realisiert. Weiters sollten die Anbringung und Demontage einfach und zeitsparend zu realisieren sein und darüber hinaus die vorhandenen Geräte nicht verändert werden. Eine Befestigung mittels Bohrlöcher oder Kleber ist damit untersagt. Da das Drehrad des Filters zur manuellen Bedienung als Zahnrad ausgeführt ist, liegt es nahe selbiges für die Antriebsseite zu nutzen. Die Suche nach einem passenden Kunststoffzahnrad aus dem Fachhandel schlug fehl. Deshalb entschloss man sich das Zahnrad des Filters als Ersatzteil anzufordern. Bei Nichtverfügbarkeit hätte man als weitere Ausweichoption den 3D-Druck eines Zahnrades in Erwägung gezogen. Der nächste Schritt bedarf der Auswahl eines Antriebes. Drei Motoren kommen für den Mechanismus in Frage. Die erste Option ist ein Gleichstrommotor mit Endschaltern. Die zweite Möglichkeit wäre die Nutzung eines Schrittmotors. Als dritte Variante könnte der Servomotor die richtige Wahl sein. Gegen den DC-Motor spricht die notwendige Umpolung für den Richtungswechsel beim Schwenken. Außerdem benötigt der Einsatz von Endschaltern mehr Raum und Kabel. Hierfür wären 8 Drähte notwendig. Die verlegten Kabel könnten sich durch den beschränkten Platz als störend auswirken. Durch die Endschalter wäre das System eventuell auch fehleranfälliger, da mehr Sensoren beteiligt wären als notwendig. Das Schwenken zwischen den beiden anzufahrenden Winkeln wäre mit dieser Variante nicht besonders genau. Nach dem Schwenkvorgang würde die Elektronik, durch den ausgelösten Endschalter, den Motor spannungsfrei schalten. Als Konsequenz könnte der Motor kein Moment aufnehmen, weshalb der Winkel und damit der Polarisationsfilter durch äußere Einflüsse verstellt werden könnte. Auch der Schrittmotor wurde als Option ausgeschlossen, obwohl dieser im Vergleich zum DC-Motor wesentlich weniger Nachteile mit sich bringt. Im Vergleich zum Servomotor werden beim Schrittmotor fünf statt nur drei Kabeladern benötigt. Da diese allerdings als Strang vom Motor wegführten, wäre dies kein Problem und dient als Notiz am Rande. Ein Argument gegen die Nutzung des Schrittmotors ist die Notwendigkeit eines Motortreibers, der als Platine zwischen Steuerung und Motor geschaltet werden muss. Generell ist die

Steuerung des Schrittmotors aus programmiertechnischer Sicht schwieriger als die Steuerung eines Servomotors. Außerdem müsste auch beim Schrittmotor ein Endschalter für die initiale Positionsbestimmung verwendet werden, da dieser technisch bedingt seine eigene Position nicht selbst bestimmen kann. Weiters kann es bei Unterversorgung durch etwaige Störungen zu Schrittverlusten kommen. In diesem Fall müssen alle Messungen nach dieser Störung als fehlerhaft betrachtet werden, da die Information der Position damit verloren ginge. Selbst wenn es danach zu keinen Schrittverlusten mehr käme, wären alle Messungen um die verlorene Schrittzahl verschoben. Die auf den Schrittmotor bezogenen Probleme können durch die Verwendung eines Servomotors umgangen werden. Durch die Bauart gibt es festgelegte Positionen. Das bedeutet null Grad sind auch noch nach einem Stromausfall oder einem Systemneustart null Grad ohne eine Kalibrierung durchführen zu müssen. Für die Steuerung sind nur drei Drähte notwendig, zwei Versorgungsleitungen und eine Signalleitung. Auch die Anforderung, die Kosten für den Schwenkapparat möglichst gering zu halten, sprechen für die Nutzung eines Modellbauservomotors. Zuletzt ist auch das geringe Gewicht für richtig dimensionierte Servomotoren als positiv zu betrachten.

Als Servomotoren bezeichnet man Elektromotoren, die mit einem Rotorlagesensor ausgestattet sind. In kleineren Modellbauservos kommen ausschließlich Gleichstrommotoren zum Einsatz. Doch auch Schrittmotoren und andere elektrische Antriebe bezeichnet man in Verbindung mit einem Drehwinkelgeber als Servoantrieb. Im Hobbybereich sind diese aus dem Modellbau bekannt. Dort werden sie beispielsweise für die Steuerung der Lenkung eines ferngesteuerten Autos oder für den Flugzeugmodellbau eingesetzt. Doch auch in der Industrie spielen Servomotoren eine wichtige Rolle. Dort werden sie zum Beispiel im Verpackungsbereich, in Industrierobotern und modernen CNC Fräsmaschinen eingesetzt.

Für die Positionsbestimmung gibt es zwei verschiedene Sensorarten, den Inkrementalgeber und den Absolutwertgeber. Beim Inkrementalgeber wird die Position indirekt, über das Anfahren einer Startposition bestimmt. Die Positionsbestimmung beim Absolutwertgeber erfolgt direkt ohne Anfangsposition. Diese Sensoren können optisch, magnetisch, induktiv, kapazitiv oder mittels Schleifkontakten realisiert werden (17).

Bei dem für den Schwenkmechanismus eingesetzten Modellbaumotor wird die Positionsbestimmung mit Hilfe eines Potentiometers auf der Ausgangwelle bestimmt. Die Regelstrecke vergleicht den Istwert des Potentiometers mit dem Sollwert der mit Hilfe der Signalleitung mittels Pulsweitenmodulation eingespeist wird. Die Breite der Pulse legt den Sollwert fest und steht damit in direktem Zusammenhang mit dem gewünschten Winkel. Liegen Soll- und Istwert auseinander, steuert die Regelstrecke den Gleichstrommotor solange in Richtung Sollwert, bis beide Werte ident sind. Kommt es durch äußeren Einfluss zu einer Verstellung des Winkels, so wird auch dies direkt durch die Regelstrecke korrigiert. Die meisten dieser Servomotoren weisen einen eingeschränkten Drehbereich von 180° auf. Da im vorliegenden Anwendungsfall um 90° geschwenkt werden muss, folgt daraus keine Einschränkung. Man sollte diese Motoren für genaue Anwendungen nicht im Grenzbereich verwenden, da diese dort meist nicht stabil laufen. Der Bereich von 10° bis 170° kann allerdings mit ausreichender Sicherheit als genau betrachtet werden. Bei dem ausgewählten Servomotor handelt es sich um einen Antrieb aus dem Modellbau. Sowohl Bauform als auch alle weiteren technischen Details eignen sich ausgezeichnet für den Einbau als Schwenkantrieb. Da der Motor nicht aus dem Profibereich angeschafft werden muss, können die Projektkosten weiterhin auf einem niedrigen Niveau gehalten werden.

Nachdem nun ein passender Antrieb gefunden wurde, musste in weiterer Folge eine Möglichkeit gefunden werden, diesen zu befestigen. Um das Zahnrad an der Antriebswelle des Motors zu montieren wurde ein Zubehörteil des Servos verwendet. Dieses dient, mit geringfügiger Modifikation des Zahnrades, als Adapter zwischen Motor und Zahnrad. Abbildung 9 zeigt das Zahnrad des Polarisationsfilters sowieso den Servomotor samt befestigtem Ersatzzahnrad. Da die umliegenden Bauteile und im speziellen auch das Gehäuse des Polarisationsfilters nicht durch Bohrungen, Schrauben oder Klebstoffe beschädigt werden sollten, lag es nahe, die Befestigung durch kraftschlüssige Verbindung zu realisieren. Der Polarisationsfilter und das Spektroskop geben nun die Position des Servomotors samt Zahnrad eindeutig vor. Abbildung 10 zeigt, dass der Motor nur oberhalb montiert werden kann, da das Spektroskop hier bauliche Grenzen setzt. Grundsätzlich wäre es möglich das Servogehäuse um 90° zu drehen. Dies hätte den Vorteil einer Verkürzung des Hebelarmes und damit eine Verringerung des Moments. Dadurch würde die Einspannung gering-

fügig weniger belastet werden. Man entschied sich jedoch dagegen, da die Ausrichtung des Motors in Abbildung 10 zeigt, dass diese die Zahnräder freilegt. Die vorhin genannte Variante würde diese verdecken. Da jedoch der Blick auf die Zahnräder besonders für die Montage des Aufbaus und zur Kontrolle während des Versuchs wichtig ist, entschied man sich für diese Orientierung. Abbildung 10 zeigt weiters die Befestigung des Aufbaus auf dem Filtergehäuse durch die Zuhilfenahme von Klemmkräften. Um die Konstruktion möglichst leicht zu gestalten, wurden Profile aus Aluminium verwendet. Wie auch beim Servomotor handelt es sich bei allen verwendeten Teilen um einfach zugängliche und kostengünstige Bauteile. Auf beiden Vierkant-Hohlprofilen wurden mittels Metallkleber Aluwinkel angebracht, welche die richtige und plane Ausrichtung der Konstruktion am Polarisationsfiltergehäuse gewährleisten. Der Motor wurde mittels Gummipuffer zur Schwingungsdämpfung und mittels Schrauben auf den Aluminumprofilen befestigt. Der vom Servomotor wegführende Kabelstrang wurde zur Zugentlastung Mittels Bohrung und Kabelbinder an der Konstruktion befestigt. Die Bohrungen für die beiden in länge gebrachten Gewindestangen sorgen mittels der Flügelmuttern für die benötigte Klemmkraft. Dieser Aufbau gewährleistet eine schnelle und einfach Montage sowie Demontage, ohne die umliegenden Bauteile zu gefährden.

Abbildung 9: Servomotor mit Zahnrad an Polarisationsfilter

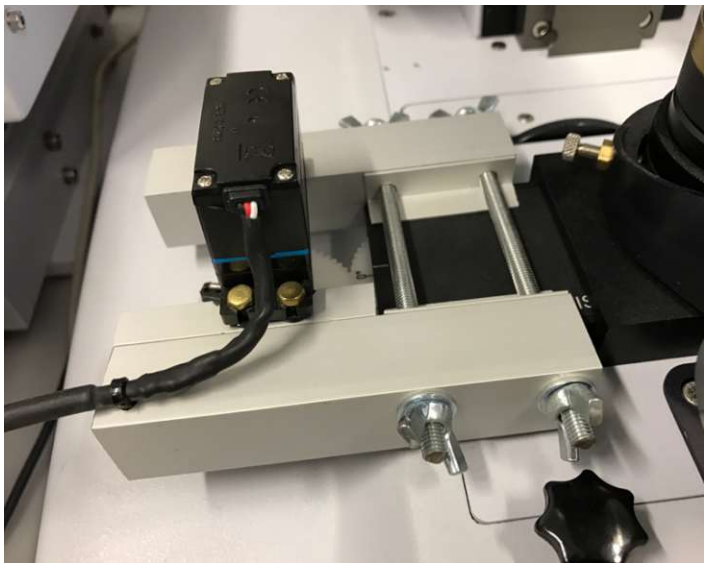


Abbildung 10: Motorhalterung am Polarisationsfilter

Um den Schwenkmechanismus zu montieren müssen die folgenden einfachen Schritte abgearbeitet werden. Wie üblich muss das Polarisationsfiltergehäuse in die vorgesehene Öffnung eingesetzt und richtig positioniert werden. Danach wird der Filter auf 0° eingestellt. Wird der Servomotor durch die Steuerelektronik angesteuert, so stellt sich dieser beim Systemstart automatisch auf die 0° Einstellung ein, falls diese nicht bereits vorliegt. Nachdem diese Kalibrierung vorgenommen wurde, kann die Schwenkkonstruktion angebracht werden. Dabei ist darauf zu achten, die Flügelmuttern so weit zu öffnen, dass die Konstruktion durch diese nicht unter Spannung steht. Im nächsten Schritt müssen die beiden Zahnräder zusammengeführt werden ohne die 0°-Einstellung des Filters zu verändern. Auf den Servomotor muss hier nicht geachtet werden, da dieser auch bei mechanischer Verstellung des Winkels automatisch wieder die richtige Position findet. Wenn Zahnräder passend in einander greifen und die beiden Winkel der Konstruktion plan am Filtergehäuse aufliegen, müssen die Flügelmuttern handfest angezogen werden. Die Flügelmuttern sollten abwechselnd festgezogen werden bis eine ausreichende Stabilität des Aufbaus gewährleistet werden kann.

Der Einsatz eines Servoantriebes setzt, wie bereits erwähnt, die Möglichkeit einer Signalübertragung mittels PWM voraus. Dieses Signal kann durch unterschiedliche Geräte generiert werden. Zum einen wäre eine ähnliche Steuerung, wie die der Zugprüfmaschine denkbar. Hier wird mittels der Software LabVIEW die Hardware der

Maschine mittels eines Computers angesprochen und gesteuert. Zum anderen wäre ein von einem Computer unabhängiges System wünschenswert. Deshalb wurde nach einer Möglichkeit gesucht, ein autarkes System zu entwickeln. Die Recherche ergab, dass die Steuerung mit Hilfe von Microcontrollern realisiert werden könnte. Hierbei wurden verschiedene Systeme analysiert. Besonderes Augenmerk wurde in weiterer Folge auf die Systeme rund um Raspberry Pi, PICmicro und Arduino gelegt. Wobei hier direkt angemerkt werden muss, dass es sich bei dem Raspberry Pi nicht um einen Microcontroller sondern um einen Einplatinencomputer handelt und dieser deshalb nicht zu den Microcontrollern zählt, er allerdings ähnlich verwendet werden kann. Hinter allen genannten Marken stecken immer größer werdende Gemeinschaften, die sogenannten Communities, die bei Problemen für Abhilfe sorgen. Vor allem jedoch sind viele der Projekte Open Source und damit für alle frei zugänglich. Dies erleichtert das Erlernen der zu verwendenden Programmiersprachen ungemein, zumal die Community mit kostenlosen Videos und How-tos dafür sorgt, dass auch Anfänger schnell den Einstieg in die Welt der Mikrocontroller finden.

Die Hardwarekomponente der Schwenksteuerung mit Hilfe eines Raspberry Pi's wäre grundsätzlich über die integrierte I/O (Input/Output) Schnittstelle des Minicomputers möglich. Da dieser jedoch für den vorgesehenen Einsatz als überqualifiziert einzustufen ist, fiel dieser aus der Auswahl. Diese auf Linux basierenden Mini-PC's mit integrierter WLAN-Anbindung sind dazu in der Lage, Hausautomatisierungen, NAS-Systeme, Server, Überwachungssysteme und weitere unzählige Projekte zu realisieren. Auch die Software für die aufgezählten Systeme sind in den allermeisten Fällen Open Source. Die noch relativ junge und kostengünstige Hardware bietet hiermit neue Möglichkeiten. Besonders interessant ist auch der kostengünstigste Raspberry Pi lautend auf den Namen Zero, mit einem Preis von nur rund fünf Euro. Dieser bietet sich grundsätzlich auch für kleinere Projekte an. Für die Elektronik des Schwenkmechanismus ist aber der Raspberry Pi Zero überdimensioniert und gegebenenfalls durch seine Verbindung mit dem Internet in Zukunft angreifbar.

Auch die beiden Systeme rund um PICmicro und Arduino unterscheiden sich, da auch der Arduino, obwohl oft so betitelt, streng genommen kein Mikrocontroller ist. Somit handelt es sich nur um die Geräte der Marke PICmicro um „echte" Mikrocontroller, da diese ohne Platine und weiterer Peripherie verkauft werden. Dies macht die Handhabung bedeutend komplizierter, da das Aufspielen des Codes bzw. der Software

einen Debugger bzw. Programmer benötigt. Da die Umsetzung mittels eines Arduinos dieses Gerät nicht benötigt und ein Debugger die Kosten des Projekts nicht unwesentlich in die Höhe schießen lassen würden, entschied man sich die Elektronik auf Basis des Arduinos aufzubauen. Hier ist nochmals besonders hervorzuheben, dass für die Verwendung der Steuerung keine Software, keine Treiber und damit auch keine Installationen notwendig sind. Auch eventuelle Probleme bezüglich Softwarekompatibilitäten mit unterschiedlichen Betriebssystemen werden hiermit direkt umgangen. Die Installation wird nur dann notwendig, wenn in den Code der Steuerung eingegriffen werden muss bzw. zusätzliche neue Optionen implementiert werden sollen. Die Software ist allerdings für alle gängigen Betriebssysteme frei zugänglich.

Für die Programmierung eines Arduinos ist kein externer Programmer nötig, da sich die notwendige Technik bereits auf der Platine des Arduinos befindet. Es werden mittlerweile verschiedene Modelle in unterschiedlichen Größen und Leistungsklassen angeboten. Außerdem spielt für die Auswahl das Wissen über die Anzahl der notwendigen Ein- bzw. Ausgänge eine wichtige und zielführende Rolle. Die Benennung der verschiedenen Modelle ist weitgehend intuitiv aufgebaut. Beispiele hierfür sind Micro, Nano, Uno, Mega. Diese sind der Größe und Leistung der Reihe nach angeführt. Während Micro, Nano und Mega dies durch den Namen verdeutlichen, fällt der Arduino Uno aus der Reihe. Der Grund hierfür liegt daran, dass es sich bei dem Uno um den ersten Arduino handelte. Dieser wurde schon mehrfach überarbeitet und liegt nun bereits in der dritten Version vor. Laut Datenblatt des Uno's befinden sich auf diesem 14 digitale Input/Output Pins, während sechs davon die Generierung eines PWM-Signals beherrschen. Außerdem sind sechs analoge Input Pins vorhanden (18). Diese sind vor allem für den Einsatz von Sensoren interessant und werden für die Schwenkelektronik nicht benötigt. Da die Ausmaße des Steuergeräts auf Grund der Platzverhältnisse klein gehalten werden sollten und nur wenige I/O Pins für Servo, Start/Stopp-Knopf und Display benötigt wurden, entschied man sich das kleinere Modell, den Arduino Nano, für die Umsetzung zu nutzen. Da es sich sowohl bei Soft- und Hardware im Bereich des Arduino's um Open Source Projekte handelt, ist die ohnehin bereits kostengünstige Hardware auch von Drittanbietern als Nachbau zu wesentlich günstigeren Preisen verfügbar. Der originale Arduino bzw. Genuino ist sinngemäß am oberen Ende der Preisskala unter den Arduino-Boards zu finden. Die Marke Arduino wird innerhalb Europas auf Grund namensrechtlicher Streitigkeiten der

Gründer auch als Genuino, eine Verschmelzung der Begriffe genuine (zu Deutsch echt) und Arduino, vermarktet (19). Da die meisten Nachbauten ohne Logo des Originals ausgeführt werden, handelt es sich hierbei keineswegs um Plagiate und sind diese damit völlig legal einzusetzen. Im Selbstversuch wurde ein originaler Arduino Uno mit einem Nachbau verglichen. Als Resultat wurde, neben dem geringfügig anderen Erscheinungsbild durch eine andere Farbe der Platinen, nur ein konkreter Unterschied in der Handhabung erkannt. Viele dieser Nachbauten setzen auf kostengünstigere Chips zur Umsetzung der Kommunikation zwischen Computer und Board. Da dieser von der Arduino-Entwicklungsumgebung nicht ohne weiteres erkannt wird, muss ein zusätzlicher Treiber installiert werden (20). Nach erfolgter Installation konnten keine weiteren Unterschiede festgestellt werden. Die untersuchten Nachbauten wirkten auch qualitativ nicht weniger hochwertig als das Original. Um die Kosten des Projekts also weiterhin möglichst gering zu halten, wurde ein Nachbaumodell des Nano's erworben. Trotz der kleineren baulichen Größe des Nano's im Vergleich zum Uno, stellt dieser mehr Pins bereit. 22 digitale I/O Pins stehen bereit, wobei auch hier sechs ein pulsweitenmoduliertes Signal erstellen können. Außerdem stellt der Nano acht analoge Input-Pins zur Verfügung. Der auf beiden Boards verwendete Microcontroller ATmega328 bietet 16 MHz Rechenleistung und einen Speicher von 32 kB. Für den Bootloader sind 2 kB in Verwendung, weshalb der hochzuladende Programmcode nicht größer als 30 kB sein kann.

Die Programmierung des Microcontroller-Boards erfolgt grundsätzlich über eine serielle Schnittstelle. Die Umsetzung dieser erfolgt über den bereits erwähnten Chip, welcher als USB-Seriell-Konverter dient. Der bereits am Microcontroller vorprogrammierte Bootloader sorgt zusammen mit dem Konverter für die Programmierbarkeit des Arduino-Boards ohne weiterem Programmiergerät. Genau dies macht diese Boards besonders praktikabel und einsteigerfreundlich, da es nun reicht Computer und Board mittels passendem USB-Kabel zu verbinden, um erste Schritte im Bereich der Microcontroller-Programmierung zu vollziehen (21). Sollte der Konverterchip durch Fehlbedienung, beispielsweise durch einen Kurzschluss, zerstört werden, so muss nicht zwangsweise das gesamte Board aufgegeben werden. Auf den meisten Arduino-Boards befindet sich eine ISP-Schnittstelle. Mittels eines Programmiergeräts und passendem Adapter kann der Microcontroller damit weiterhin programmiert werden. Außerdem wird dies notwendig, falls der Bootloader beschädigt

wird. In diesem Fall muss über diese Schnittstelle der Bootloader erneut geflasht werden. Der Bootloader kann beispielsweise durch ein Ziehen des Kabels während des Uploads beschädigt werden. Diese Programmiergeräte sind im Gegensatz zu denen der Marke PICmicro kostengünstig zu erwerben, da auch diese auf dem Open Source Prinzip bestehen. Diese Geräte sind bereits unter zehn Euro zu erwerben. Für die Anschaffung der Programmer für die PICmicro-Chips müsste zum Vergleich mit etwa der zehnfachen Summe gerechnet werden. Da die nachgebauten Arduinos, besonders im Fall des Nano's, bereits ab etwa drei Euro zu erwerben sind, lohnt sich die Fehlersuche bei einem möglichen Defekt des Geräts meist nicht.

Nachdem der Schwenkmechanismus realisiert und ein Steuerungssystem in Form des Microcontroller-Boards gefunden wurde, konnte man sich nun mit weiteren Anforderungen, benötigten Tastern und um Formen der Visualisierung beschäftigen. In der ersten Version der Schwenkelektronik sollte der Polarisationsfilter durch die bereits zuvor ermittelten Werte der Videoauswertung voreingestellt geschwenkt werden. Die einfachste Weise dies umzusetzen wäre, die ermittelte bzw. erwünschte Konstante innerhalb des Codes festzulegen. Abbildung 11 zeigt die Entwicklungsumgebung, die sogenannte Arduino IDE und den Code, auf dem die Schwenkelektronik aufbaut. Mit diesen wenigen Zeilen gelingt es bereits den Servomotor kontrolliert alle 3 Sekunden zu schwenken.

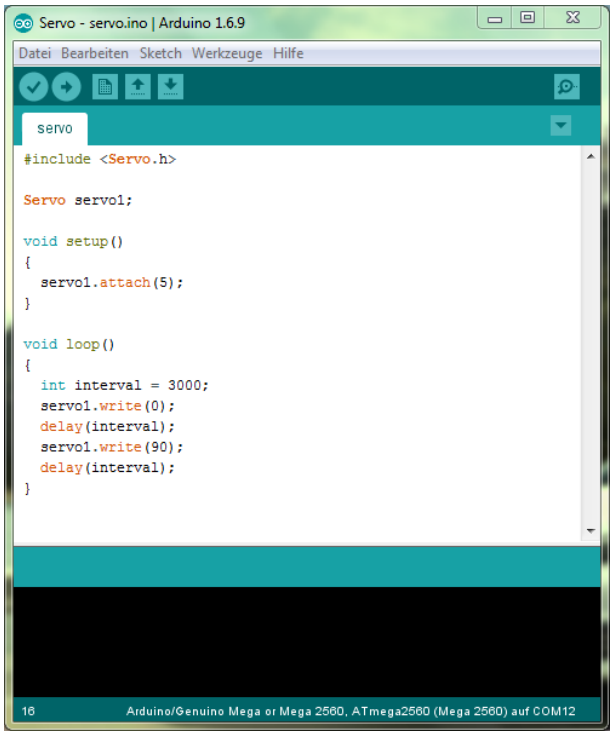


Abbildung 11: Arduino IDE, Kerncode zur Steuerung

Bei der innerhalb der Arduino IDE genutzten Programmiersprache handelt es sich um eine vereinfachte Form von C++. Die Programme sind immer, wie in Abbildung 11 zu erkennen, aufgebaut. Wie auch bei anderen Programmiersprachen erfolgt, falls notwendig, in den ersten Zeilen die Einbindung von Bibliotheken. Diese ermöglichen die Nutzung bereits vorhandener Befehle. Im Beispiel wird die Bibliothek „Servo“ aufgerufen. Im Bereich „void setup“ erfolgt unter anderem die Zuweisung der genutzten I/O Pins. Das Beispiel zeigt, dass die Signalleitung des bereits zuvor zugewiesenen „servo1“ am digitalen I/O Pin 5 angeschlossen sein muss. Dabei muss darauf geachtet werden, dass dieser Pin die Generierung eines PWM Signals unterstützt. In den meisten Fällen sind diese Pins auf der Platine gesondert gekennzeichnet. Sollte dies nicht der Fall sein, genügt ein Blick in das Datenblatt des Microcontroller-Boards. Im Abschnitt „void loop“ werden die ausführenden Befehle und Berechnungen festgelegt. Im Gegensatz zu den anderen Abschnitten des Programms wird der Code in „void loop“ wiederholt. Dafür ist es nötig zu wissen, dass der Microcontroller den Code Zeile für Zeile von oben nach unten abarbeitet. Wurde die letzte Zeile des Bereichs „void loop“ ausgeführt, so wird im nächsten Schritt wieder mit der ersten Zeile von „void loop“ begonnen. Im Beispiel wird zuerst die Variable „interval“ mit dem Wert 3000 belegt. Da dieser Wert innerhalb dieses Codes konstant ist, wäre es ebenfalls möglich diese Zeile im ersten Bereich, also außerhalb der „void loop“, zu setzen. Die Einführung einer Variable macht besonders für eine mehrfache Belegung des Wertes innerhalb des Codes Sinn. Muss der Wert nachträglich verändert werden, reicht es den Wert der Variable zu ändern. Anderenfalls müsste der komplette Code nach dem Wert durchsucht und geändert werden. Besonders bei größeren Programmen ist dies für bessere Übersichtlichkeit oft sinnvoll. In der nächsten Zeile wird der Winkel des Servoantriebs auf null Grad festgelegt. Die Funktion „delay(x)“ legt mit x die gewünschte Wartezeit in Millisekunden fest. Nach der Wartezeit von 3000 ms im Beispielcode, wird der Winkel mit der nächsten Zeile auf 90 Grad geändert. Im nächsten Schritt folgt wieder eine Pause von drei Sekunden. Wie bereits beschrieben beginnt der Ablauf danach erneut, wodurch der Winkel wieder auf null Grad verstellt wird.

Nach dem Upload des Programms mittels der IDE auf das Arduino-Board, verrichtet der Microcontroller umgehend seine Arbeit. Unter der Voraussetzung der richtigen Verkabelung wird der Servomotor im 3 Sekunden-Takt schwenken. Da die Boards nur bei sehr schwachen Motoren ausreichend Leistung bieten, wird in den meisten Fällen ein

zusätzliches Netzteil notwendig sein, so wie auch im Fall des Schwenkmechanismus. Der Servomotor wird mit drei Kabeln gesteuert bzw. versorgt. Ein Massekabel (GND), eine Versorgungsleitung (PLUS), und ein Signalkabel. Bei der Nutzung eines zusätzlichen Netzteils muss darauf geachtet werden, die GND-Leitung mit Netzteil und Microcontrollerboard zu verbinden. Wird die Masseverbindung zum Arduino-Board vergessen, so kann durch den fehlenden Stromkreis kein PWM-Signal übertragen werden. Wird der Arduino vom Strom getrennt und nach unbestimmter Zeit wieder versorgt, so wird der hochgeladene Code erneut von der ersten Zeile beginnend abgearbeitet. Eine Trennung des Stroms ist also wie ein Neustart bzw. ein Reset des Systems zu betrachten.

Dass der Code aus Abbildung 11 für die Steuerung des Schwenkmechanismus nicht ausreichend ist, ist naheliegend. Um den Start des Schwenkvorgangs kontrolliert durchführen zu können wird damit ein Startknopf notwendig, welcher gleichzeitig auch als Stoppknopf dienen soll. Außerdem sollte das gewünschte Schwenkintervall einstellbar sein, um mögliche Einstellungsänderungen komfortabel anpassen zu können. Um das Intervall einstellen zu können, wurde ein Potentiometer für die Justierung in einem vorgegebenen Bereich verwendet. Damit ist es möglich die Schwenkzeit zwischen 2000 und 4000 ms anzupassen. Für die genaue Justierung der eingestellten Zeit mittels des Potentiometers wird ein Display für die Darstellung notwendig. Um auch weitere Informationen zu visualisieren wurde auf ein 4-Zeilen Display gesetzt, welches 20 Zeichen pro Zeile darstellen kann. Wie auch die bis jetzt genannten Komponenten wurde auch hier darauf geachtet die Kosten gering zu halten. Das Display, welches auf Grund eines Controllers auf der Rückseite mit nur 4 Kabeladern betrieben wird, ist mit Kosten von etwa 15 Euro etwa doppelt so teuer wie der Servoantrieb. Neben den beiden Versorgungsleitungen Plus und Ground kommen zwei I^2C-Datenbusleitungen zum Einsatz. Um die mittels Poti eingestellte Zeit zu bestätigen wird ein Taster notwendig. Dieser kann, abhängig vom Systemzustand, mehrfach belegt werden. So dient dieser als Bestätigungstaste, Start- und Stopptaste. Des weiteren wurde ein Taster für die Durchführung eines Resets implementiert. Dieser muss allerdings nicht wie die anderen Eingabekomponenten softwareseitig im Code festgelegt werden. Auf jedem Arduino-Board befindet sich ein Reset-Pin. Wird dieser mit der Ground kurzgeschlossen, so wird der Reset durchgeführt. Anders als möglicherweise erwartet, wird hierdurch nichts gelöscht bzw. auf Werkseinstellungen

zurückgesetzt. Der Reset führt, wie die kurzfristige Trennung des Stroms, zu einer erneuten Ausführung des auf dem Microcontroler befindlichen Codes. Dabei geht die zuvor angezeigte Information am Display verloren, da der Controller wieder mit der ersten Zeile des Codes startet. Die Speicherung der Daten wäre mit Hilfe von Zwischenspeichern grundsätzlich möglich, ist aber für die Umsetzung der Schwenkelektronik nicht notwendig, da diese ohnehin für jede erneute Versuchsdurchführung erneut gestartet werden muss und die Daten für die Auswertung irrelevant sind.

Neben der Visualisierung der einstellbaren Zeit, die sogenannte „Steptime“, wurden die weiteren Zeilen des Displays für die Darstellung des „Counters“, der „Runtime“ und der “Pause“ genutzt. Der „Counter“ zählt die einzelnen Schwenkvorgänge bzw. FTIR-Messungen. Er beginnt mit null und zählt hoch nachdem eine Positionsänderung fertiggestellt wurde. Der Counter gibt also die Nummer der aktuellen FTIR-Messungen an. Abbildung 12 zeigt einen der ersten Prototypen der Schwenkelektronik. Die Zeile „Runtime“ gibt die Zeit ab Auslösen des Startknopfes in Sekunden an. Sollte der Versuch angehalten werden und später fortgesetzt werden, gibt die Zeile „Pause“ die verstrichene Zeit in Sekunden seit Auslösen des Stoppknopfes an.

Um mehr der verfügbaren, digitalen Input/Output-Pins des Microcontroller-Boards zu nutzen und die Bedienung intuitiver zu gestalten wurden LED’s verwendet. Zwei dieser Leuchtdioden zeigen den aktuellen Winkel des Polarisationsfilters bzw. des Schwenkmechanismus an. Ist der Filter auf der 0°-Position, so leuchtet die dazugehörige mit der Aufschrift 0° gekennzeichnete LED und die 90°-LED bleibt dunkel. Analog wird mit der Anzeige der 90°-Position vorgegangen. Eine dritte LED zeigt in der Farbe Grün den Zustand des Servos bzw. den aktiven Programmablauf an. Leuchtet diese und ist der der Motor richtig angeschlossen so wird der Servo von der Elektronik gesteuert.

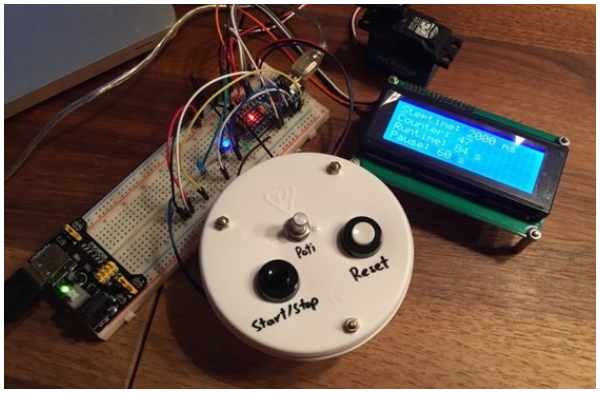


Abbildung 12: Konzeptaufbau mittels Steckbrett

Beim Auslösen der Tasten kommt es technisch bedingt zum sogenannten „Prellen“. Der Microcontroller überwacht den binären Zustand des Tasters. Im nicht ausgelösten Zustand entspricht dieser dem Wert 0. Im Gegensatz dazu nimmt der ausgelöste Taster für den Controller den Wert 1 an. Da der Microcontroller allerdings mit 16 MHz arbeitet, können 16 Millionen Rechenoperationen pro Sekunde abgearbeitet werden. Wird die Taste nun manuell ausgelöst so kommt es beim Übergang vom Aus zum Ein Zustand mechanisch bedingt zum mehrfachen Aus- und Einschalten, dem Prellen. Mechanische Vibrationen innerhalb des Tasters führen dazu, dass der Schaltzustand für wenige Millisekunden schwankt. Doch auch optoelektrische, chemische und Flüssigkeitsschalter haben dieses Problem, welches in Abbildung 13 verdeutlicht wird (22).

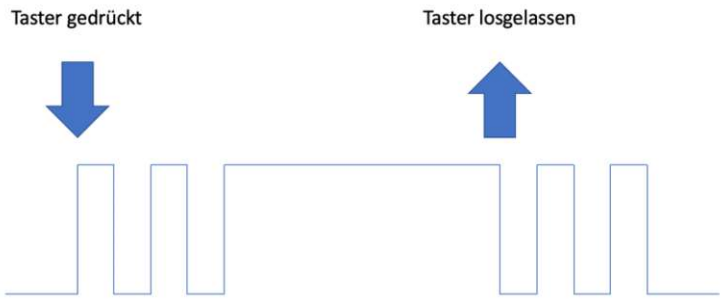


Abbildung 13: Schaltzustand eines prellenden Schalters (22)

Dieses kurzzeitige Auftreten von mehreren Einschaltimpulsen muss für die Umsetzung der Schwenkelektronik umgangen werden, um einen exakten Startzeitpunkt zu definieren, der direkt mit dem ersten Einschaltsignal festgelegt wird. Da außerdem der Startknopf auch als Stoppknopf benutzt werden soll, würde während des Prellens mehrfach Start und Stopp hintereinander ausgelöst werden. Der letzte Schaltimpuls würde dann nach dem Zufallsprinzip entweder Start oder Stopp bedeuten und damit entweder einige Millisekunden verzögert starten oder mit Stopp die Pause einleiten und auf einen erneuten Start warten. Damit ist klar, dass dieses Problem umgangen werden muss. Dies kann sowohl hardwareseitig als auch softwareseitig realisiert werden. Die Elektronikindustrie hält für Spezialfälle hardwaretechnisch prellfreie Schalter bereit, die entweder mittels Feder-Dämpfer-Mechanismus die auslösenden Schwingungen tilgen oder mittels eingebauter elektronischer Schaltverzögerung das Problem umgehen. Diese speziellen Schalter sind jedoch bedeutend teurer als ihre

Pendants. Die Schaltverzögerung kann aber auch relativ einfach softwareseitig umgesetzt werden. Vereinfacht ausgedrückt wurde dies für die Schwenkelektronik wie folgt umgesetzt. Durch eine IF-Bedingung wird der Zustandswert des Tasters überwacht. Diese wird dann aktiv, wenn dem Wert 0 der Wert 1 folgt. Findet dieses Ereignis statt, so wird, neben dem weiteren auszuführenden Code in der Bedingung, eine Verzögerung eingeleitet, welche das erneute aktiv werden innerhalb dieser Zeit unterbindet. Da eine Prellung etwa 10 ms andauert, sollte der Wert der Verzögerung unbedingt darüber liegen. Da jedoch im Fall der Steuerung eine direkte Aktivmachung des Tasters nicht notwendig ist, wird die Verzögerung großzügig auf 500 ms ausgedehnt. Das bedeutet, dass in dieser Zeit der Schalter inaktiv ist und auch gewollt ausgelöste Schaltbefehle nicht von der Software erkannt werden.

Abbildung 12 zeigt neben dem Konzeptaufbau mittels Steckbrett auch das notwendig Werden einer Platine und eines geeigneten Gehäuses. Für die Realisierung der Platine gibt es verschiedene Optionen. Die Platine kann beispielsweise mittels Software (z.B. Autodesk EAGLE) designt und dann professionell von einem Anbieter geätzt werden. Dies hat den Vorteil, dass Leiterbahnen auf beiden Seiten der Platine verwendet werden könnten, wodurch kleinere Baugrößen möglich werden.

Des weiteren ist es möglich das Design der Platine einseitig auszulegen und selbst zu ätzen. Hierfür gibt es verschiedene Möglichkeiten. Eine günstige und einfache Methode wäre die Herstellung mittels fotobeschichteter Platinen. Hierbei wird ein Bild der Leiterbahnen auf eine durchsichtige Folie in schwarzer Farbe gedruckt, auf der fotoreaktiven Platine befestigt und belichtet. Im nächsten Schritt wird der nicht belichtete Teil der Platine mittels einer Chemikalie entfernt. Dadurch wird die Kupferschicht freigelegt und kann mittels Natriumpersulfat oder Eisen-(III)-Chlorid geätzt werden. Die Leiterbahnen werden durch die belichtete Fotobeschichtung nicht angegriffen und es entsteht eine DoItYourself-Platine. Im letzten Schritt müssen noch die Bohrungen für die zu verlötenden Pins der Bauelemente durchgeführt werden. Diese können sich durch die kleinen Lötflächen, deren Mitte genau getroffen werden muss, und den zu verlötenden Pins als besonders schwierig herausstellen. Durch die Nutzung von SMD-Bauteilen könnten einige Bohrungen umgangen werden. Doch auch der Einsatz dieser Elektronikteile ist im Zuge der Lötarbeit mit Schwierigkeiten verbunden, da diese meist besonders klein ausgeführt sind (23).

Gegen die Produktion einer Platine, in welcher Art auch immer, spricht aber, dass diese später nicht mehr abänderbar ist. Da sich die Schwenkelektronik während der Versuchsdurchführungen im Stadium der Entwicklung befindet und auch zukünftig Erweiterungen möglich sein sollen, ist von einer statischen Methode abzusehen. Da jedoch die Nutzung von Steckbrettern während der Versuche als zu fehleranfällig und für den Transport als zu unsicher eingestuft wurde, musste eine Lösung im Zwischenbereich gesucht werden.

Daraus ergab sich der Einsatz von Lochplatinen. Diese bieten sich ausgezeichnet für den Prototypenbau an, da diese zu jeder Zeit anpassbar und erweiterbar sind. Der Handel bietet unterschiedliche Varianten. Es sind Platinen mit bereits zusammengelegten Leiterbahnen, beispielsweise zeilenweise ausgeführt, sowie Platinen mit nicht zusammenhängenden, einzelnen lötbaren Bohrungen verfügbar. Für das Gerät wurde eine Variante mit jeweils drei zusammenhängenden, leitenden Bohrungen ausgesucht, da diese viele Verbindungsmöglichkeiten bietet ohne Leiterbahnen für nebeneinanderliege Verbindungsstellen ziehen zu müssen. Nachteil hieran ist, dass nicht ohne weiters Versorgungsbahnen eingerichtet werden können. In weiterer Folge der Entwicklung erwies sich die Auswahl der Lochplatinen als eine gute Entscheidung, da hierdurch eine Auslösesteuerung der Kamera hinzugefügt werden konnte. Abbildung 14 zeigt den vom Steckbrett zum Lochplatinen-Design transformierten Prototypen.

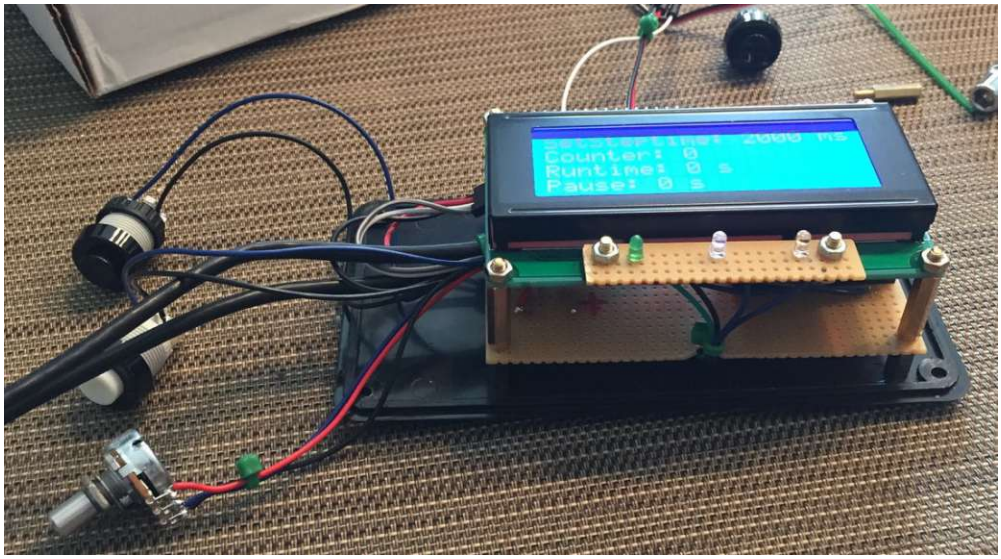


Abbildung 14: Prototyp mit Lochplatine

Außerdem wird der Aufbau des Steuergeräts gezeigt. Der Microcontroller und zusätzliche Elektronik, wie Widerstände, Transistoren und Steuerdrähte wurden auf der unteren Platine installiert. Abbildung 15 zeigt einen Blick von unten auf die Hauptplatine. Bei den beiden sichtbar größer ausgeführten Lötstellen handelt es sich um Versorgungsstellen, also Plus und Minus. Der Arduino Nano wird mit Hilfe zweier Sockelleisten mit der Platine verbunden. Die Lötstellen dieser Verbindungen sind ebenfalls deutlich im Bild sichtbar. Die Sockelleisten, und nicht das Microcontroller-Board selbst, werden mit der Platine verlötet. Dies hat den Vorteil der einfachen Austauschbarkeit, sollte der Microcontroller einen Defekt aufweisen. Ein umfunktioniertes, kostengünstiges, aus dem Zubehörhandel erhältliches Kunststoffgehäuse wurde für die Nutzung als Außenhülle der Elektronik umfunktioniert. Der Deckel des Gehäuses wird als Bodenplatte des elektronischen Aufbaus verwendet. Mittels passender Abstandshalter und dazugehöriger Schrauben wurde die Hauptplatine an der Bodenplatte befestigt. Durch weitere Abstandhalter gelingt es, das Display, welches die Größe der Hauptplatine vorgibt, oberhalb zu montieren. Eine weitere kleinere Leiterplatte wird auf der Platine des Displays befestigt um dort als LED-Leiste zu dienen. Das geschlossene Gehäuseunterteil muss nun in weiterer Folge passend bearbeitet werden. Dafür muss ein Sichtfenster für das Display, drei Bohrungen für die Leuchtdioden, Bohrungen für die Taster und das Potentiometer und Kabelausgänge geschaffen werden. Um das Gehäuse optisch anspruchsvoller zu gestalten, wurde dieses mattschwarz lackiert. Die Funktion der eingebauten Leuchtdioden, sowie das Gerät an sich wurden mittels eines Etikettiergeräts beschriftet. Da das Display bei Verstellung des Motorwinkels flackert, wurde eine zweite Versorgungsleitung alleinig für den Servo eingerichtet. Für die Versorgung wird bewusst bei beiden Kreisläufen eine Spannung von fünf Volt benötigt. Damit wird der Einsatz von USB-Steckern möglich.

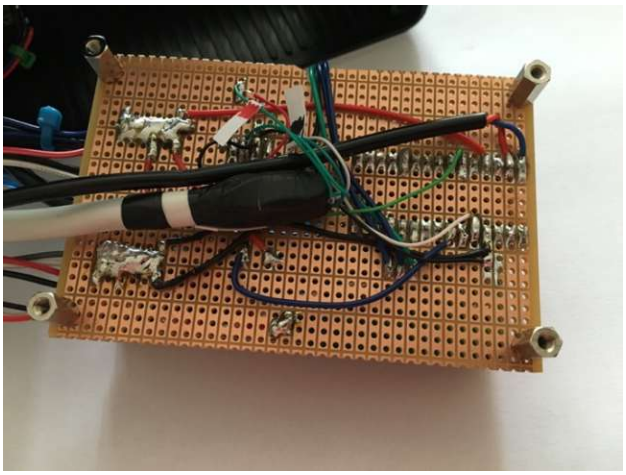


Abbildung 15: Prototyp der Hauptplatine von unten

Der Strom kann also aus beliebigen USB-Netzteilen oder von PC's mit verfügbaren Plätzen bezogen werden. Die bei USB 2.0 verfügbare Obergrenze von 500 mA wird dabei nicht überschritten. Falls es in Zukunft zu Problemen mit der Steuerung kommt, kann dadurch schnell und einfach ein möglicher Defekt eines Netzteils eruiert werden. Der dreipolige Anschluss des Servomotors wurde zum Zwecke der besseren Verstaumöglichkeit abschließbar ausgeführt. Im Falle eines defekten Servoantriebes kann dieser dadurch problemlos ausgetauscht werden. Der Anschluss wurde mittels eines dreipoligen 3,5mm-Klinkensteckers ausgeführt. Die Buchse hierfür wird im Gehäuse verbaut. Sollte eine Verlängerung notwendig sein, kann dadurch auf ein handelsübliches Audio-Verlängerungskabel zurückgegriffen werden. Alle nicht abschließbaren, nach außen führende Kabel sind mittels einer Zugentlastung vor zu starker Belastung auf deren Lötstellen gesichert. Des weiteren werden die außen liegenden Kabel durch die Nutzung eines Spiralschlauchs gebündelt. Dadurch wird der Aufbau übersichtlicher und aufgeräumter.

Auf den Stromlaufplan der bislang beschriebenen Version des Schenkmechanismus wird, wegen einer maßgeblichen Änderung, welche in Kapitel 2.3 beschrieben wird, hier nicht näher eingegangen. Selbiges gilt für den Programmcode des Prototypen.

Der Programmcode wird im Anhang 9.1 angeführt. Eine genauere Diskussion wird der finalen Version des Codes vorbehalten. Schematisch umrissen beinhaltet der in der Arduino IDE geschriebene Code folgendes. Die Tasten, das Potentiometer, der Servoantrieb, das Display, sowie die LED's werden implementiert. Die Entprellung der Tasten findet softwareseitig durch Verzögerungen statt. Zuerst findet die Abfrage des Widerstandswerts des Potentiometers statt. Der einstellbare Bereich des Poti' wird

durch Interpolation auf den gewünschten zeitlichen Bereich umgelegt. Das Auslösen des Startknopfes bestätigt vorerst die Eingabe des Poti‘s, welcher sich durch die Programmierung zwischen 2000 und 4000 ms verstellen lässt. Natürlich ist dieser Bereich beliebig erweiter- bzw. verkürzbar, unterliegt allerdings gewissen Einschränkungen. Die analogen Inputs der Arduino-Microcontrollerboards verfügen standardmäßig über eine 10 Bit Auflösung. Eine angelegte Spannung von 5 V kann also in 1023 Teilen aufgelöst werden. Der kleinste Teilschritt ergibt sich für dieses Beispiel zu etwa 5 mV. Selbiges gilt nun auch für den mathematisch umgerechneten Zeitbereich. Der gewählte Bereich wäre damit theoretisch gerundet in 2 Millisekunden-Schritten einstellbar. Da diese präzise Steuerung eines Potis praktisch nicht möglich bzw. nicht nutzerfreundlich zu bewerkstelligen ist, wurden die Schritte auf 10 Millisekunden festgelegt. Dadurch sind 200 verschiedene Einstellungen im festgelegten Bereich problemlos und bedienerfreundlich auswählbar. Nachdem das ausgewählte Zeitintervall bestätigt ist, wird der Startknopf seinem Namen gerecht und ist für das Auslösen des Hauptprogramms startbereit. Dieses Prozedere beinhaltet verschiedene kleinere Unterprogramme, wie das Auslösen des Timers, des Counters, der Statusleuchtdioden, sowie die Anzeige dieser Werte am Display. Das wichtigste Element dieses Codes bleibt jedoch der bereits gezeigte Abschnitt in Abbildung 11, die Stellung des Motorwinkels per PWM-Signal. Die vorab mittels Potentiometer eingestellte Variable definiert nun den Schwenkrhythmus des Motors.

Die Schwenksteuerung ist damit zusammen mit dem Schwenkmechanismus bereit für die ersten tatsächlichen Versuche. Für die Inbetriebnahme der Version 1 wurde eine Kurzanleitung erstellt, welche zur Vollständigkeit im Anhang 9.2 beigefügt ist. Da sich die Versuche aus den folgend angeführten Gründen als wenig zufriedenstellend herausstellten, wird hier nur die Auswertung einer Probe gezeigt. Diese soll stellvertretend für alle sechs Proben dieser Versuchsreihe zeigen, dass eine Weiterentwicklung der Steuerung notwendig wurde. Generell ist anzumerken, dass die Güte der Ergebnisse mit Version 1 der Schwenksteuerung schwankt. Unter den ausgewerteten Ergebnissen der einzelnen Proben befinden sich sowohl den Erwartungen aus der Literatur entsprechende Ergebnisse als auch stark davon abweichende. Die Auswertung des Orientierungsgrad über den Verstreckungsgrad der Probe zeigt deutliche Analogien zu den aus der Literatur stammenden Diagrammen.

Abbildung 16 zeigt die Ergebnisse von Probe 1. Auf die Darstellung des zugehörigen Spannungs-Dehnungs-Diagramms wird bewusst verzichtet, da dieses zu keinen weiteren Erkenntnissen führt und für die weitere Erklärung irrelevant ist.

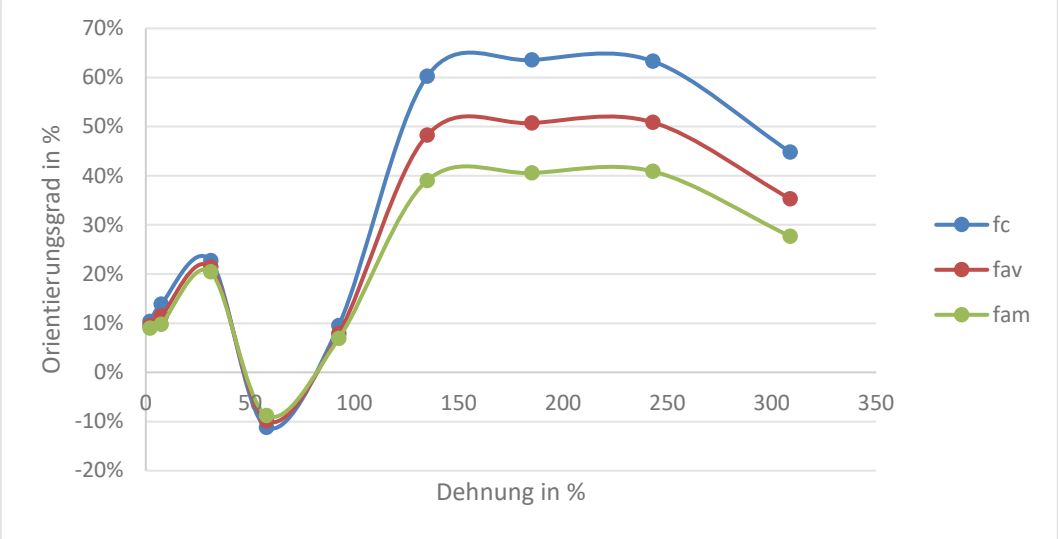


Abbildung 16: Orientierungsgrad Probe 1

Abbildung 17 zeigt zum Vergleich die aus der Literatur stammenden, zu erwartenden Ergebnisse. Daraus ergibt sich, dass ausschließlich die Proben 1 und 6 zu plausiblen Ergebnissen führen.

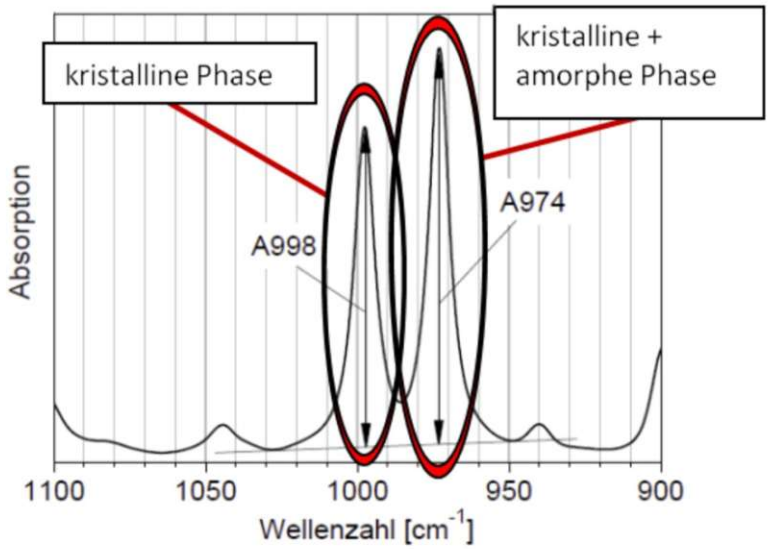


Abbildung 17: Beispielbild Absorption über Wellenzahl einer PP-Folie (12)

Abbildung 18 zeigt das Ergebnis der Auswertung zu Probe 3. Die Kurvenverläufe sind wechselhaft und deshalb als unglaubwürdig anzusehen. Zusammen mit der Tatsache, dass die Ergebnisse der anderen Proben keine Analogien untereinander aufweisen, müssen alle Auswertungsergebnisse in Frage gestellt werden. Nach eingehender Untersuchung wurde festgestellt, dass die Einführung eines festen Schwenkintervalls im Versuchsablauf fehlschlägt. Grund hierfür sind die unvorhersehbar schwankenden Messzeiten des Spektroskops.

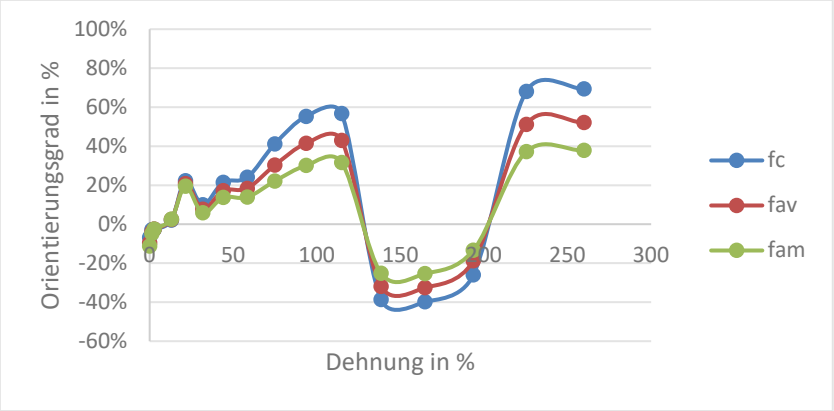


Abbildung 18: Orientierungsgrad Probe 3

Zwar wurden die zuvor ermittelten Einstellungen des Spektroskops mit möglichst geringfügig schwankenden Messzeiten angewendet, doch auch die hier auftretenden kleinen Abweichungen führen im Verlauf des Versuchs bei Auftreten von nicht dem Durchschnitt entsprechenden Werten zu einem immer größer werdenden Fehler. Die in Abbildung 19 gezeigten parallelen Zeitstrahlen sollen dieses Problem verdeutlichen. Während die Schwenkzeiten der Steuerung durch konstante Abstände dargestellt werden, zeigt der Strahl des FTIR die Verschiebung der Intervalle bei unregelmäßigen Messzeiten. Die Abbildung ist zum besseren Verständnis überspitzt dargestellt. Im Beispiel ist die dritte Messung des FTIR deutlich länger als die durchschnittlichen Messzeiten. Aus diesem Grund findet der nächste Schwenkvorgang bereits in der Messung des Spektroskops statt. Zum einen verfälscht dies das Messergebnis, zum anderen kann sich der Fehler soweit aufsummieren, dass eine 0° oder 90° Messung übersprungen wird. Dies wird im rechten Teil von Abbildung 19 deutlich erkennbar dargestellt. Als Folge für die Auswertung würden entweder zwei Messungen mit gleichem Winkel ausgewertet, oder die Reihenfolge der Messungen umgekehrt werden. Dies würde bedeuten, dass einer 0° bis 90° Messung eine 90° bis 0° Messung

folgt. Da das Auswertungstool dies nicht erkennen kann, würden die Daten weiterhin als 0° bis 90° Messungen ausgewertet werden und zu falschen Ergebnissen führen. Dies könnte die in Abbildung 18 auftretenden Schwankungen erklären, da während weiterer Auswertungsarbeiten beobachtet werden konnte, dass sich das Vorzeichen des Orientierungsgrades ändert, wenn die Reihenfolge der Messungen vertauscht wird.

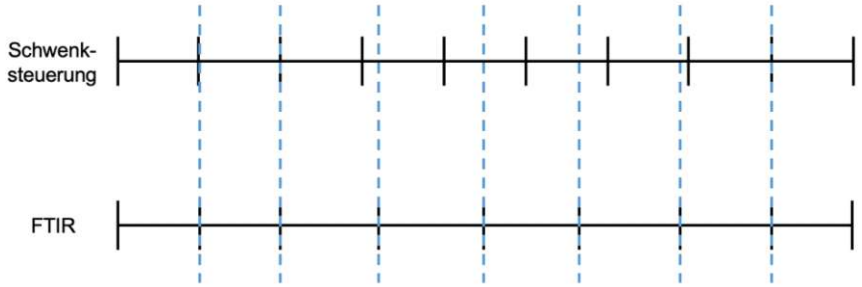


Abbildung 19: Zahlenstrahl FTIR und Steuerung

Aus diesem Grund musste der Steuerungsmechanismus neu überdacht und den neu erlangten Erkenntnissen angepasst werden. Die Überarbeitung wird in Kapitel 2.3 erläutert. Da diese auch hardwareseitige Veränderungen und Zusätze erfordert, wird in weiterer Folge von Version 2 der Steuerung gesprochen.

2.3. Version 2 der Filtersteuerung

Durch die, wie in Kapitel 2.2 beschrieben, nicht zueinander passenden Zeiten der Filtersteuerung und des Spektroskops wird eine Kommunikation der beiden Geräte notwendig. Da die Messzeiten des Spektroskops nicht konstant sind, gelingt es nicht die Schwenkintervalle mit der Einstellung eines fixen Intervalls synchron mit den Messungen des FTIR-Spektroskops zu halten. Dieses Problem kann ausschließlich über Kenntnis der Messzeiten des Spektroskops gelöst werden. Da die vorherige Ermittlung über einen Durchschnittswert fehlschlug, muss die Kenntnis während des Versuchs in Echtzeit an die Steuerung weitergegeben werden. Gelingt dies nicht so würde das Projekt scheitern.

Durch Kontakt mit dem Hersteller des Spektroskops, Firma Bruker, wurde in Erfahrung gebracht, dass grundsätzlich ein Trigger-Signal abgreifbar ist. Während die Kenntnis darüber für das Projekt neue Hoffnungen schuf, eröffneten sich dadurch auch neue

Probleme. Für die Umsetzung wird ein Adapterkabel notwendig, welches den finanziellen Rahmen des Projekts sprengen könnte. Abgesehen hiervon ist die Schnittstelle des Spektrometers TENSOR bereits mit dem Kabel des FTIR-Mikroskops HYPERION belegt. Da beide Geräte für die Versuche benötigt werden, ist es nicht möglich auf das HYPERION zu verzichten. Aus diesem Grund müsste bei der Anschaffung des Adapters dafür gesorgt werden, dass die Signale an beide Geräte, die Schwenksteuerung und an das Mikroskop, gelangen können. Dadurch würde ein zusätzlicher Adapter notwendig werden, der als Y-Weiche dient.

Durch die Kommunikation mit dem Hersteller wurde bekannt, dass durch den Adapter zwei Signale des Spektrometers übertragen werden. Pin Nummer 7 der seriellen Schnittstelle des TENSOR‘s, welche mittels einer DSUB-25 Buchse ausgeführt wurde, dient mit Hilfe eines Pullup-Widerstands der Überwachung eines von außen zugeführten Triggersignals. Um ein TRIG-Signal auszulösen, wird die anliegende Spannung kurzzeitig unterbrochen. Diese Unterbrechung muss größer als 1 ms sein, um von der Elektronik erkannt werden zu können. Das bedeutet auch, dass, falls kein Messauslöser gewünscht wird, ununterbrochen ein Spannungspotential anliegen muss. Grundsätzlich handelt es sich hierbei also um einen Sensor. Das unterbrechungsfreie Anlegen einer Spannung mit dem bereits in Version 1 verwendeten Arduino Nano ist problemlos umsetzbar.

Bei dem zweiten Signal handelt es sich um einen Aktor. Startet nun eine Messung, so teilt das ACK-Signal (Acknowledge) dies mittels PIN 19 der Schnittstelle mit. Um das hochpreisige Gerät vor Fehlbedienung, Kurzschlüssen, Überspannung und anderen äußeren Einwirkungen zu schützen, sollten die Pins des Spektroskops keinesfalls direkt abgegriffen werden. Aus diesem Grund muss das Auslesen bzw. Beschalten der Pins mittels galvanischer Entkopplung durchgeführt werden. Hierdurch gelingt es die Stromkreise des Spektroskops und der Steuerung getrennt voneinander zu realisieren, während trotzdem eine Kommunikation der beiden Systeme untereinander möglich ist. Galvanische Trennung kann mittels verschiedener Bauteile umgesetzt werden. Das Relais ist wohl das bekannteste Bauteil zur galvanischen Entkopplung. Die Trennung gelingt hier durch die Bestromung einer Spule, welche dadurch magnetisiert und in weiterer Folge den Kontakt des zweiten Stromkreises schließt. In beiden Stromkreisen können hierdurch verschiedene Potentiale und Ströme geführt werden. Dabei können

sowohl Wechsel- als auch Gleichstromkreise getrennt ausgeführt und geschalten werden.

Für die Entkopplung des Spektroskops vom Stromkreis der Filtersteuerung wird ein Optokoppler verwendet. Wie der Name schon vermuten lässt trennt dieser die Systeme durch den Einsatz optischer Hilfsmittel. Konkret wird im Inneren des Optokopplers eine Leuchtdiode aktiv, sobald diese mit Strom versorgt wird. Auf der gegenüberliegenden Seite liegt galvanisch getrennt ein Phototransistor, welcher leitfähig bzw. niederohmig wird sobald er von der LED bestrahlt wird. Dieser Prozess ist von außen visuell nicht sichtbar. Wichtig ist hierbei, dass sich ansonsten keine weiteren Bauteile im Optokoppler befinden. So müssen eventuell nötige Spannungsteiler, Pullup-Widerstände und der Vorwiderstand für die LED in der Schaltung berücksichtigt werden. Die Funktion des Optokopplers kann einfach getestet werden indem die LED zusammen mit einem der angelegten Spannung entsprechenden Widerstand bestromt wird. Im nächsten Schritt wird der Widerstand an den beiden Pins des Phototransistors mit Hilfe eines Ohmmeters gemessen. Ist die LED aktiv so wird das Messergebnis etwa null Ohm betragen. Im Gegensatz dazu fällt das Ergebnis für die inaktive LED hochohmig aus. Um die beiden Pins der Buchse des Spektroskops, welche für das TRIG- und ACK-Signal verwendet werden weiter verarbeiten zu können, werden also zwei Optokoppler notwendig. Das elektronische Bauteil MCT6 beinhaltet zwei Optokoppler in einem DIP-8 Gehäuse. Abbildung 20 zeigt den schematischen Aufbau des Bauteils aus dem Datenblatt. Sie zeigt, dass die beiden Leuchtdioden auf einer Seite des Bauteils positioniert sind und zwar von Pin 1 bis 4. Auf der gegenüberliegenden Seite befinden sich die beiden Phototransistoren. Mit Hilfe dieses Bauteils und einigen wenigen Widerständen gelingt es nun, die Stromkreise für beide Signale galvanisch zu trennen.

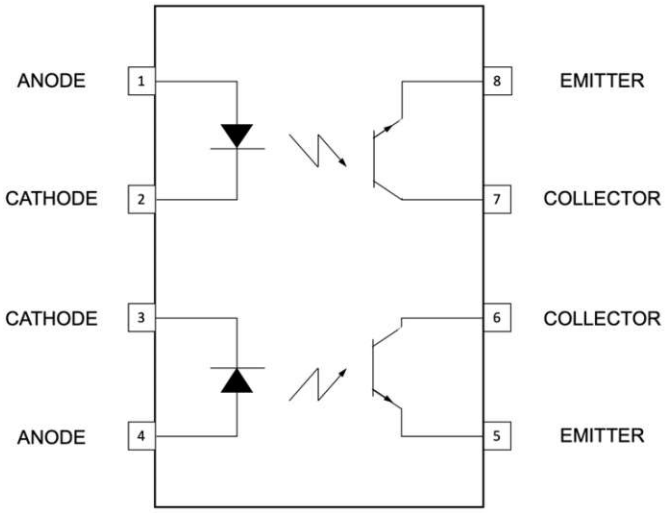


Abbildung 20: Schematischer Aufbau Optokoppler MCT6 (24)

Abbildung 21 zeigt den schematischen Aufbau der galvanisch getrennten Signalleitungen. Leuchtdiode Nr. 1, welche zwischen Pin 1 und 2 liegt, wird mittels eines 300 Ω Widerstands zwischen den ACK-Pin Nr. 19 und dem 5 Volt Pin mit der Nr. 17 verbunden. Beim Anschluss der LED ist auf die Polung zu achten, so muss die Anode (Pin 1) mit dem 5 Volt Pin verbunden werden. Wird also das ACK-Signal aktiv, so leuchtet LED1 des Optokopplers und schaltet den Phototransistor 1, welcher sich zwischen Pin 8 und 7 befindet, niederohmig. Durch einen Spannungsteiler kann dieses Umschalten des Phototransistors durch den Arduino überwacht werden. Der Collector des Phototransistors muss mit dem GND Anschluss des Microcontrollers verbunden werden. Besonders wichtig ist, die beiden Masseanschlüsse der Stromkreise nicht zu verbinden, da sonst keine galvanische Trennung besteht. Der Emitter-Pin des Phototransistors wird mit einem analogen Inputpin des Arduinos verbunden und mittels eines 10 kΩ Pullup-Widerstands mit dem 5 Volt Pin des Boards verbunden. Bei der tatsächlichen Umsetzung wurde der analoge Pin A6 für das Acknowledge-Signal verwendet. Durch diesen Aufbau kann nun die Information über das Stattfinden einer Messung an den Microcontroller weitergegeben werden.

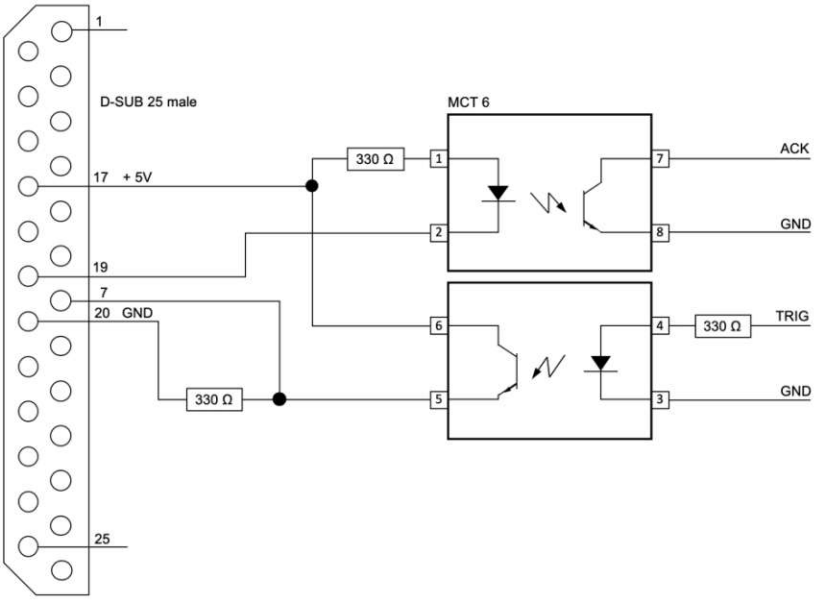


Abbildung 21: Adapterkabel Schema mit Optokoppler

Der zweite Optokoppler wird in die andere Richtung verwendet, sodass nun der Microcontroller ein Signal an die Leuchtdiode sendet. Auch diese LED muss mittels eines Vorwiderstands mit 330 Ω versorgt werden. Das Triggersignal wird mittels des digitalen I/O Pins D2 des Arduino Nano umgesetzt. Auf der Seite des Phototransistors 2, welcher zwischen Pin 5 und 6 liegt, wird auch mittels eines Spannungsteilers der Zustand überwacht. Der Collector-Pin wird mit dem 5 Volt Versorgungspin der Buchse des Spektroskops verbunden. Der Emitter-Pin mit dem Pin 7 der Buchse. Der Spannungsteiler wird mittels eines 330 Ω Widerstands realisiert. Durch diese Schaltung wird das durch den Arduino generierte Triggersignal galvanisch getrennt an das Spektroskop weitergegeben.

Nachdem nun der grundsätzliche Aufbau der galvanisch getrennten Verbindung zwischen Steuerung und Spektroskop verdeutlicht wurde, wird nun auf den Stromlaufplan der Steuerung eingegangen. Im Mittelpunkt des Aufbaus steht das Arduinoboard mit dem ATmega328P Microcontroller. Die Platine des Mikrocontrollers dient im Grunde dem besseren Zugang der Pins des Chips. Jedoch bietet das Komplettpaket Arduino noch weitere praktische Komponenten, die den Prototypenbau vereinfachen. Wie schon erwähnt, wird dadurch die Programmierung per USB-Kabel möglich. Ein weiteres Beispiel ist der Spannungswandler, der eine Eingangsspannung zwischen 7 und 12 Volt zulässt und damit Eingänge die deutlich über der Arbeitsspannung des Controllers, welcher mit 5 Volt betrieben werden muss, ermöglicht. Dadurch werden

diese Controller auch gerne für mobile Einsätze ohne kabelgebundene Stromversorgung, zum Beispiel mit 9 Volt Blockbatterien, betrieben.
Grundsätzlich wäre es auch möglich den Microcontroller ohne das Arduinoboard einzusetzen. Da Microcontroller, vor allem jene mit mehreren Ein- und Ausgängen, allerdings meist als SMD-Bauteile vorliegen, würde eine eigens angefertigte Platine notwendig werden. Generell würde das Ergebnis deutlich professioneller aussehen, allerdings würde dies die Kosten des Projekts, wie vorab beschrieben, deutlich steigern, zumal die bisherigen Kosten sehr niedrig ausfielen.

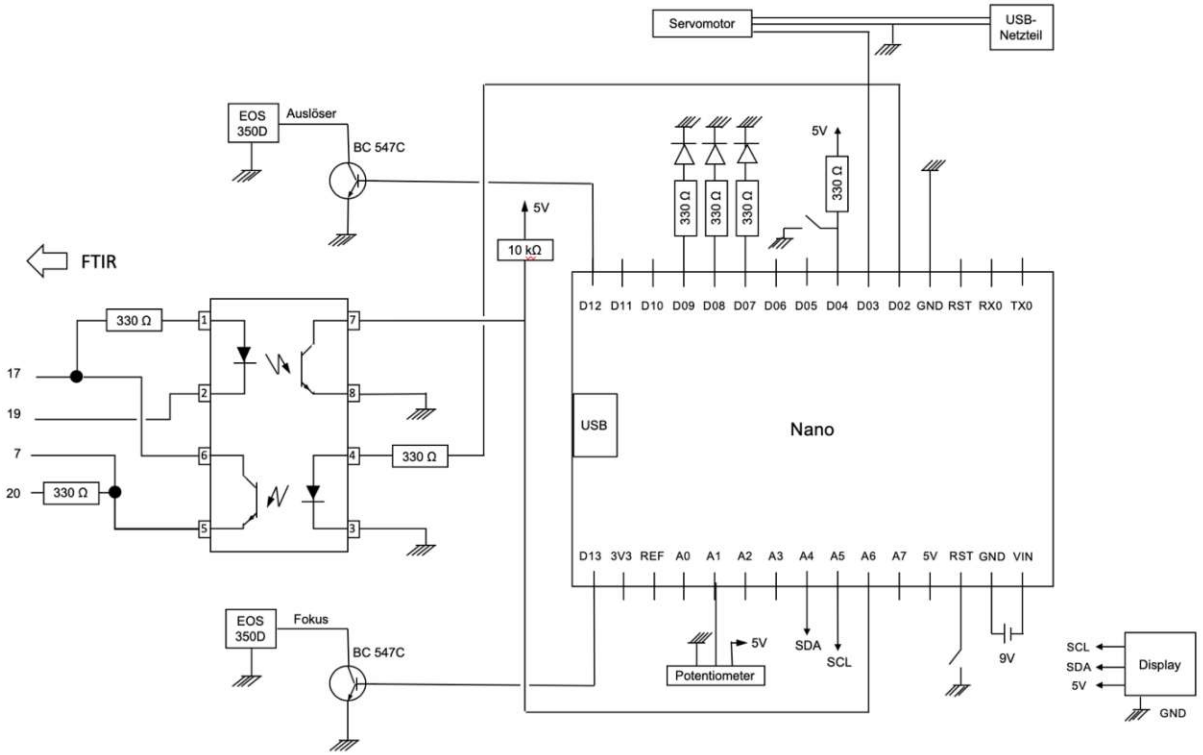


Abbildung 22: Stromlaufplan der Filtersteuerung

Abbildung 22 zeigt nun den Stromlaufplan der Steuerung. Der stromführende Aufbau der neuen Version ist weitgehend ident mit dem der ersten Version. Die wesentlichen Unterschiede sind zum einen das Hinzufügen des Optokopplers zur Kommunikation mit dem Spektroskop. Zum anderen ist durch die gegenseitigen Triggersignale keine Einstellung der gewünschten Schwenkzeit mehr notwendig. Dadurch ist das Potentiometer zur Einstellung des Schwenkintervalls obsolet und dient in der aktuellen Version keinem Zweck. Die Anschlüsse und das Potentiometer selbst bleiben jedoch erhalten, um für eine eventuell später auftretende Einsatzmöglichkeit dienen zu können. Im

Code scheint der verstellbare Widerstand nicht mehr auf, damit ist dieser funktionslos. Die Funktionsweise wird trotz wegfallender Notwendigkeit im Folgenden kurz beschrieben. Der mittlere Pin des Potis ist mit dem analogen Input A1 verbunden. Die äußeren beiden Pins werden mit den GND- und 5V-Pins des Boards verbunden. Funktionstechnisch misst der Controller die Spannung zwischen GND und mittleren Pins des Potis. Wird dieser verstellt so ändert sich der Widerstand zwischen diesen beiden Pins. Wird der Widerstand zwischen Pin 1 und 2 kleiner, so wird der Widerstand zwischen 2 und 3 größer. Der Gesamtwiderstand des Potis an den beiden äußeren Polen bleibt also gleich. Damit bleibt auch die Spannung zwischen den beiden äußeren Pins bei 5 Volt. Der Arduino misst also die potentielle Änderung zwischen GND und A1 auf Basis der Spannung von 5V. Dies zeigt die Wichtigkeit einer möglichst konstanten Spannungsquelle bzw. eines passenden Spannungswandlers. Dies wird durch die Elektronik des Arduinoboards gewährleistet.

Wie schon vorab erwähnt wird das Display durch einen hardwareseitigen Treiber durch nur vier Leitungen betrieben. Die beiden Versorgungsleitungen werden entsprechend mit dem Board verbunden. Die Datenbusleitungen SDA und SCL werden mit den analogen Pins A4 und A5 verbunden. Eine zugehörige Library im Programmcode vereinfacht die Steuerung auch softwareseitig. Die Reset-Taste wird zwischen den RST-Pin und GND geschalten. Die Spannungsversorgung des Arduino-Boards wird mit einem 9 Volt Netzteil realisiert. Die zulässige Eingangsspannung am VIN-Pin des Boards darf zwischen 7 und 12 Volt liegen. Pin D2 wird für das Senden des Triggersignals verwendet. Mittels Optokoppler wird dieses galvanisch getrennt übermittelt.

Die Signalleitung des Servos, welche oft orange oder gelb ausgeführt ist, wird am digitalen I/O-Pin D3 angeschlossen. Dieser unterstützt die für die Steuerung des Servos notwendige Generierung des PWM-Signals. Um ein Flackern des Displays bei Schwenkvorgängen zu umgehen wird der Servoantrieb getrennt mit Spannung versorgt. Da fünf Volt ausreichend sind, wird die Stromzufuhr über einen handelsüblichen USB-Stecker umgesetzt. Dadurch kann jedes USB-Netzteil bzw. jede USB-Schnittstelle eines Computers zur Versorgung des Servos herangezogen werden. Der Pluspol der Quelle wird mit dem Pluspol des Motors verbunden. Wichtig ist es die Massepole des Boards, des Servoantriebs und der zusätzlichen Spannungsquelle zu verbinden. Ohne diese Verbindung ist einer der beiden Stromkreise unterbrochen, wodurch der Motor nicht gesteuert werden kann.

An Pin D4 des Arduinoboards wird die Start-/Stopp-Taste implementiert. Der Status der Taste wird mit Hilfe eines 10 kΩ Pullup-Widerstands ausgelesen. Die Pins D7, D8 und D9 werden für die Statusleuchtdioden verwendet. In Version 1 wurde der Pin D7 zur Anzeige des Status des Motors genutzt. Da dies jedoch deutlich sicht- und hörbar ist, muss dies nicht durch eine eigene Anzeige visualisiert werden. Ab Version 2 wird diese Leuchtdiode zur Anzeige des gesendeten Triggersignals verwendet. Schickt die Steuerung also ein Signal an das Spektroskop, so wird dies durch das kurze Aufleuchten der grünen Statusleuchtdiode sichtbar gemacht. Alle eingebauten Leuchtdioden werden mit einem 330 Ω Vorwiderstand betrieben. Mit Pin D8 wird die 0°-Position des Filters angezeigt. Die LED an Pin 9 zeigt im aktiven Zustand an, dass sich der Polarisationsfilter auf der 90°-Stellung befindet.

Der Stromlaufplan in Abbildung 22 zeigt des weiteren eine Entwicklung der Steuerung, die sich erst im späteren Verlauf der Versuche mit Version 2 ergab. Die Gründe für diese Erweiterung und die Erklärung zum Stromlauf werden in Kapitel 2.4 angeführt.

Um das Spektroskop während der Entwicklung des Programmcodes für Version 2 nicht zu besetzen und damit andere Experimente einzuschränken, wurde die Seite des Spektroskops simuliert. Außerdem, damit eine Beschädigung des Spektroskops durch den experimentellen Aufbau ausgeschlossen werden kann. Hierfür wurde ein zweites Arduino-Nano-Board verwendet, welche dazu dienen sollte, dass von der Steuerung gesendete Triggersignal zu verarbeiten und entsprechend das Acknowledge-Signal zurückzugeben. Der Stromlaufplan der Simulationsseite wird in Abbildung 23 dargestellt. Die nummerierten eingehenden Kabel aus der Steuerungsseite von Abbildung 22 sind nach dem entsprechenden Pin des Spektroskops benannt. Wie bereits beschrieben, dienen diese vier Leitungen der Kommunikation.

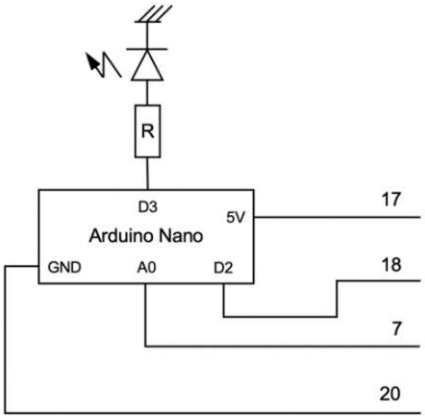


Abbildung 23: Stromlaufplan der Simulationsseite

Kabel Nr. 17 wird am 5-Volt-Anschluss des Arduinos auf der Simulationsseite angeschlossen und dient der Versorgung der LED im Optokoppler und damit der Generierung des ACK-Signals. Abbildung 24 zeigt den Versuchsaufbau rund um den Optokoppler zwischen Steuer- und Simulationsseite.

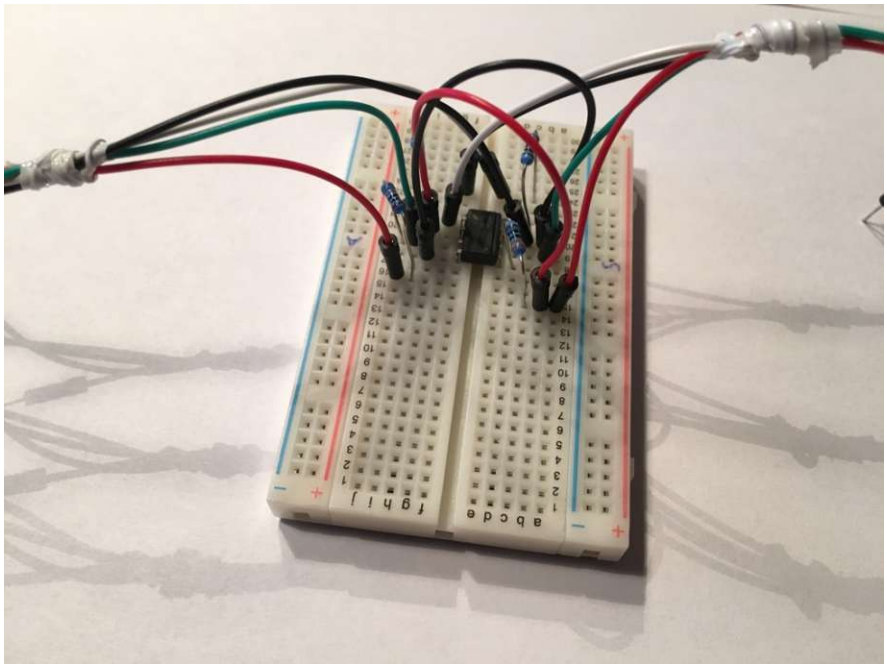


Abbildung 24: Galvanische Trennung mittels Optokoppler, Versuchsaufbau

Außerdem dient Pin Nummer 17 der Überwachung des Status des Trigger-Phototransistors. Die Überwachung des Status wird am Eingangs-Pin A0 umgesetzt. Über das Kabel Nr. 20 wird hierfür ein Pulldown-Widerstand mit 330 Ω mit dem Masseanschluss des Boards verbunden. Um das Testen der Programmierung einfacher zu gestalten, wurde an Pin D3 ein LED samt Vorwiderstand hinzugefügt. Diese zeigt den Status des ACK-Signals an. Leuchtet diese so wird simuliert angezeigt, dass gerade eine Messung stattfindet.

Um die durch den Optokoppler realisierte galvanische Trennung beizubehalten, wurden für den simulationsseitigen und den steuerungsseitigen Microcontroller gesonderte Stromversorgungen genutzt. Es sei erwähnt, dass die Verwendung der gleichen Stromquelle für beide Microcontroller nicht überprüft wurde. Für Nachahmungen wird also das gleiche Setup empfohlen. Eventuelle Schäden, durch die nicht vorhandene galvanische Trennung, können sonst nicht ausgeschlossen werden.

Da der Programmcode der Simulation kompakt und übersichtlich gehalten werden konnte, kann dieser in Abbildung 25 gezeigt werden.

SimulationSpektroskop §

```
const int ACKout = 2;
const int ACKled= 3;
int oldTRIGin = 0;
int TRIG;

void setup() {
  Serial.begin(9600);
  pinMode(ACKout, OUTPUT);
  pinMode(ACKled, OUTPUT);
}

void loop() {
  digitalWrite(ACKout, LOW); //HIGH
  digitalWrite(ACKled, LOW);
  int TRIGin = analogRead(0);
  if (TRIGin <= 500) {
    TRIG = 1;
    }
  else {
    TRIG = 0;
  }

  Serial.println(TRIGin);

  if ((TRIG == 1) && (oldTRIGin == 0)) {
    digitalWrite(ACKout, HIGH);
    digitalWrite(ACKled, HIGH);
    delay(3000);
    }

oldTRIGin = TRIGin;
}
```

Abbildung 25: Programmcode der Simulationsseite

Die ersten vier Zeilen des Codes dienen der Zuweisung der genutzten Pins auf der Platine und der Festlegung einer Variablen für das eingehende Triggersignal. Danach wird die Baudrate der seriellen Schnittstelle auf 9600 festgelegt. Dadurch ist es möglich, mit Hilfe des seriellen Monitors der Arduino IDE beliebige Stati anzuzeigen. Im Loop-Bereich des Codes werden vorerst die beiden ACK-Pins inaktiv gesetzt. Mit dem Befehl analogRead(0) wird der Status des Triggerkabels überwacht und die Variable mit dem ausgelesenen Wert belegt. In den nachfolgenden IF-Bedingungen wird überprüft, ob der gemessene Wert einem aktiven oder inaktiven Triggersignal entspricht. Wird ein Trigger erkannt, so wird die Variable TRIG mit dem Wert 1 belegt, andernfalls mit 0. Mit der nächsten IF-Bedingung wird nun die Variable TRIG überwacht. Die Bedingung wurde so implementiert, dass diese nur zutrifft, wenn eine aufsteigende Flanke erkannt wird. Nur so kann gewährleistet werden, dass der Anfang

des Triggersignals als tatsächlicher Auslöser dient. Ansonsten würden auch nachfolgende Werte mit 1 die Bedingungen als wahr setzen. Wird die IF-Bedingung aktiv, so werden die beiden ACK-Pins aktiv gesetzt. Die Simulation sendet damit den Status „aktive Messung“ an die Steuerung und erleuchtet zusätzlich die ACK-LED. Der Programmcode der Steuerungsseite wird nachfolgend umrissen erklärt. Die Steuerungsseite funktioniert grundsätzlich ähnlich. Hier wird auf die absteigende Flanke des ACK-Signals gewartet, um dann erneut ein Triggersignal auszusenden. Dieses Ping-Pong sorgt für die gewünschten synchronisierten Messungen.

Nachdem nun die Steuerung mit Hilfe der Simulation entwickelt und angepasst wurde, sollte diese am Spektroskop einen ersten Testlauf absolvieren. Hierfür wurden die vier Steuerleitungen mit den zugehörigen Pins eines DSUB25-Steckers verbunden. Die verwendeten Anschlüsse wurden bereits in der Erklärung zu Abbildung 21 näher erläutert. Wie bereits zuvor erwähnt, ist es nicht ohne weiteres möglich, zwei Geräte an der seriellen Schnittstelle des Spektroskops anzuschließen. Da noch kein entsprechender Adapter zur Verfügung stand, wurde die Verbindung des HYPERION getrennt, um die Schnittstelle für die Steuerung frei zu machen. Der erste Test verlief weitergehend vielversprechend, sodass nur wenige Unterschiede im Vergleich zur Simulation bestanden. Es ergab sich jedoch eine besonders wichtige Erkenntnis. In der Theorie kann direkt nach der Messung des Spektroskops ein erneuter Trigger, ausgelöst durch die Steuerung, gesendet werden. Softwareseitig würde man hier auf die absteigende Flanke des ACK-Signals warten und erneut einen Trigger auslösen. In der Praxis stellte sich heraus, dass der Trigger nicht direkt nach der absteigenden Flanke des ACK-Signals gesendet werden kann, da das Spektroskop oder der Computer noch nicht bereit sind das Triggersignal zu verarbeiten. Die Messung ist also bereits abgeschlossen und die Maschine meldet dies auch, doch das Triggersignal kann erst geschickt werden, sobald ein kleines Fenster im Programm erscheint, welches mitteilt, dass auf das Triggersignal gewartet wird. Wird der Trigger geschickt bevor dieses Bestätigungsfenster erscheint, so wird dieser nicht erkannt. In weiterer Folge stoppt damit die Kommunikation der beiden Geräte, da die Steuerungssoftware nach dem Aussenden des Triggers erneut auf die absteigende Flanke des ACK-Signals wartet. Da jedoch mit dem Trigger keine erneute Messung angestoßen wurde, bleibt das ACK-Signal im inaktiven Status hängen.

Grundsätzlich wäre es also besser, das Erscheinen des Fensters als Beendigung der Messung detektieren zu können. Die technische Umsetzung hierfür würde sich allerdings als kostspielig und schwierig herausstellen. Außerdem wären erneut größere Umbaumaßnahmen und ggf. eine komplette Umstellung der Hardware notwendig, da für das Erkennen des Fensters vermutlich eine visuelle Auswertung notwendig werden würde. Da das Fenster innerhalb der Software OPUS erscheint und es sich dabei nicht um Open-Source-Software handelt, dürfte ein Abgreifen dieses Signals nicht einfach umzusetzen sein.

Glücklicherweise erscheint das Bestätigungsfenster im zeitlich vorhersehbaren Rahmen nach der absteigenden Flanke des ACK-Signals. Mit dieser Information kann innerhalb der Steuerungssoftware eine entsprechende zeitliche Verzögerung implementiert werden, welche dafür sorgt, dass das Triggersignal zeitlich versetzt gesendet wird. Im Versuch wurde herausgefunden, dass zwischen Messsignalende und dem Auftreten der Meldung relativ konstant eine Sekunde verstreicht. Zur Sicherheit wird der Offset um 20 % erhöht. Damit sinkt das Risiko, dass das Triggersignal versendet wird bevor die Software des Spektroskops bereit ist. Innerhalb des Programmcodes wird dies mit Hilfe der delay-Funktion umgesetzt, welche den Programmablauf nach Feststellen der abfallenden ACK-Flanke für 1200 ms anhält.

Da die Messungen während dem Zugversuch durchgeführt werden, soll die Zeit, welche zwischen zwei Messungen verstreicht, möglichst gering sein. Die gesamte erfindliche Zeit kann in folgende Terme aufgeteilt werden:

Gesamtzeit/Messung = FTIR-Messzeit + Datei-Verarbeitung/Speicherung + Trigger-Offset

Der Term der FTIR-Messzeit wurde bereits diskutiert. Dieser kann durch Verringerung der Scans auf Kosten der Qualität verringert werden und wirkt sich im unteren, einzahligen Bereich nicht maßgebend aus. Die beiden letzteren Terme können vermutlich nur noch durch Geräteneuanschaffungen beeinflusst werden. Der Vollständigkeit halber sei darauf hingewiesen, dass ein schnellerer Computer evtl. zu einer verkürzten Speicherzeit führen könnte. Allerdings könnte dies auch alleinig vom Spektroskop und dessen analogen Anschlüssen abhängen. Dadurch könnte, falls eine kürzere Gesamt-

zeit zu Gunsten einer höheren Auflösung notwendig wird, ein neues Spektroskop unumgänglich sein. Diese Aussagen liegen allerdings im Bereich der Vermutungen und sind gegenwärtig nicht überprüfbar.

Nachdem dieses Problem erfolgreich und simpel gelöst werden konnte, wurde nun der bereits erwähnte Adapter für den Anschluss zweier Geräte an das Spektroskop nötig. Unerwarteter Weise zog auch der Adapterbau nicht unerhebliche Probleme mit sich.

Das grundsätzliche erste Konzept für den Bau des Adapters sieht vor, ein handelsübliches Verlängerungskabel so umzubauen, dass die benötigten Leitungen gesplittet in ein weiteres Kabel mit Buchse enden. In der Praxis würde das 25-polige RS232-Kabel, welches als Verlängerung Buchse und Stecker bietet, in der Mitte durchtrennt werden. Im nächsten Schritt würden die vier benötigten Kabelstränge jeweils durch eine Zuleitung mittels Lötverbindung erweitert werden. Die restlichen Leitungen müssten, wegen der Durchtrennung der Leitungen ebenfalls mittels Löten verbunden werden. Auf Grund der fünfundzwanzig verdrillten Leitungen ist es nicht möglich nur die vier benötigten Kabel zu durchtrennen, da die Lötarbeit mehr Platz benötigt, als im verdrillten Zustand vorhanden ist.

Vor dem Umbau des Kabels zum Adapter wurde dieses als Verlängerung zwischen den beiden Geräten des Spektroskops, dem Tensor und dem Hyperion getestet. Im ersten Versuch wurde das vorhandene Verbindungskabel durch das neue Kabel verlängert. Diese Art der Verbindung schlug fehl, da keine Kommunikation zwischen den beiden verbundenen Geräten aufgebaut werden konnte. Deshalb wurde als weiterer Versuch das vorhandene Kabel durch das Verlängerungskabel ersetzt. Auch dieser Versuch scheiterte. Mit der Annahme ein defektes Kabel erlangt zu haben, wurde ein weiteres Verlängerungskabel eines anderen Anbieters verwendet. Dieses konnte ebenfalls nicht als Verlängerung, wohl aber als Ersatz des Kabels dienen. Durch diese Erkenntnis ergab sich die Vermutung, dass der Innenwiderstand der benutzen Kabel und Anschlüsse die Ursachen des Problems seien. Das Durchmessen mittels Ohmmeter ergab einen deutlichen Unterschied der jeweilig gemessenen Widerstandswerte der verfügbaren Kabel. Während das originale Kabel einen ungefähren Widerstand von 0,1 Ω aufwies, wurde bei dem Verlängerungskabel, welches als Ersatz dienen konnte, ein Innenwiderstand von 0,7 Ω ermittelt. Da die meisten Multimeter im niedrigen Bereich der Messung von Widerständen allerdings als ungenau gelten, können diese Werte nur zur Einschätzung der vorliegenden Größenordnung

verwendet werden. Für die Ermittlung genauer Messwerte könnten hochwertige Multimeter bzw. andere Methoden herangezogen werden. Die angezeigten Messwerte deuten allerdings auch in dieser Art schon unverkennbar auf ein Widerstandsproblem hin.

Aus dieser Erkenntnis folgerte die Suche nach einem Kabel mit möglichst geringem Innenwiderstand aller Komponenten. Da dieser Wert bei keiner der verfügbaren und finanziell akzeptablen Verlängerungen angeführt wurde, entschied man sich den kompletten Adapter selbst zu fertigen. Dabei ist es möglich die Komponenten, also die Anschlüsse und Leitungen, gezielt auszuwählen, da hier üblicherweise die Widerstandswerte angegeben werden.

Sowohl Kabel als auch die benötigen Anschlüsse - zwei Buchsen und ein Stecker - wurden für den Adapterbau in Hinblick auf einen möglichst geringen Widerstand ausgewählt. Die 25-poligen RS232 Anschlüsse weisen Pins mit Vergoldung auf und zielen damit auf einen geringen Übergangswiderstand ab. Auch das 25-adrige Kabel weißt einen geringen Innenwiderstand auf, sodass ein Gesamtwiderstand ähnlich des originalen Kabels zu erwarten ist. Als Abzweigung zur Filtersteuerung wird weiters ein 4-adriges Kabel mit ansonsten gleichen Eigenschaften verwendet. Gegenüber dem vorherigen Konzept ergibt der eigenständige Adapterbau einen deutlichen Mehraufwand, da die Leitungen an die Lötösen der Anschlüsse verlötet werden müssen. Auch die sorgfältig durchgeführten Lötverbindungen dienen der Verringerung des Widerstands. Möglicherweise sind schlecht ausgeführte Verbindungsstellen der Grund für die erhöhten Widerstandswerte der gekauften Verlängerungen.

Für den Adapterbau sind auf den Anschlussbauteilen mittels der Lötösen 54 Lötstellen nötig. Innerhalb des Kabels werden weitere vier Lötstellen als Abzweigungen nötig. Um auch hier den Widerstand der Leitungen möglichst gering zu halten, wurde die Hauptisolierung großzügig mittels Spezialwerkzeug entfernt, um in weiterer Folge ausschließlich die vier benötigen Kabeladern auftrennen zu können und die restlichen Kabel unberührt zu lassen. Abbildung 26 zeigt links einen geöffneten Anschluss mit den bereits verlöteten Ösen. Hier sei angemerkt, dass dieser Aufbau höchster Konzentration bedarf, da sehr beengte Platzverhältnisse vorliegen. Darüber hinaus muss die Kabeladernlänge je nach Position im Anschluss angepasst und auf einwandfrei durchgeführte Lötverbindungen geachtet werden. Des weiteren muss auf der Gegenüberseite darauf geachtet werden, die selbe Pinbelegung auszuführen. Die

zweite Seite ist dadurch deutlich schwieriger zu fertigen, da die 25 Leitungen vor dem Verlöten geordnet werden müssen, um nicht durch überkreuzte Leitungen zu wenig Platz im Gehäuse zu erhalten. Ansonsten wird das Schließen des Gehäuses später unmöglich.
Auf der rechten Seite der Abbildung 26 ist der fertig ausgeführte Adapter zu sehen. Die aufgetrennte Ummantelung des Kabels wurde mittels Schrumpfschlauch und Spiralschlauch nach dem Umbau geschützt.

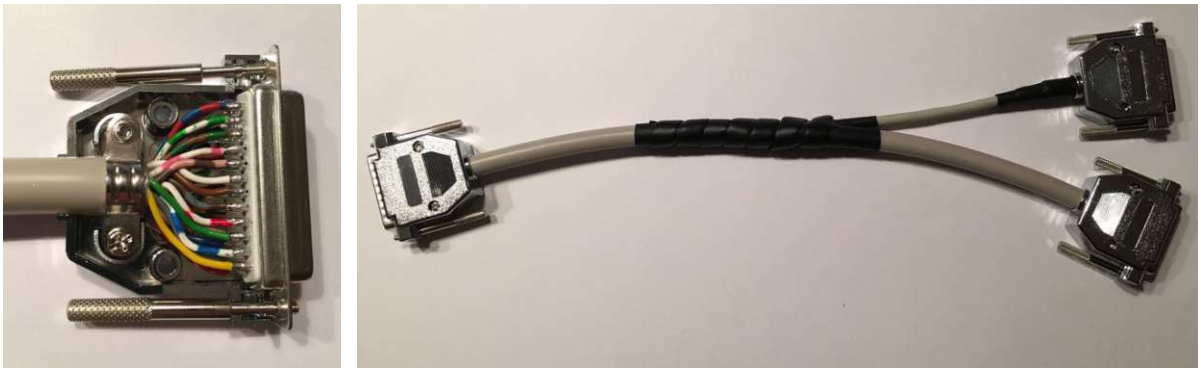


Abbildung 26: Adapter zur Integration der Filtersteuerung

Im nächsten Schritt wurde ein Testlauf in der Praxis mit dem Adapter vollzogen, welcher durchwegs positiv ausfiel. Daraus kann gefolgert werden, dass die Annahme der Problemursache richtig erkannt wurde. Mit diesem Adapter wurde es nun erstmals möglich die Messungen durch Trigger gesteuert durchzuführen. Neben den positiven Ergebnissen, wurde erkannt, dass die Masse des Adapters, vor allem die Steifigkeit und Unhandlichkeit in Zukunft zu Problemen führen könnten. So könnte der Adapter dadurch die serielle Buchse des Spektroskops mit der Zeit abnutzen. Außerdem wurde erkannt, dass diese Art von Schnittstellen nicht für ständige An- und Abschluss-vorgänge konzipiert sind. Aus diesem Grund wurde erneut recherchiert und eine potenziell bessere Lösung gefunden, bei welcher es möglich sein sollte, den Adapter dauerhaft am Spektroskop zu belassen. Abbildung 27 zeigt die äußerst elegant und kompakt ausgeführte, zweite Version des Adapters ohne Gehäuse. Bei dem erworbenen Bauteil können alle Leitungen selbstbestimmt angelegt werden. Dem Kunststoffgehäuse wurde für das nach außen führende Kabel eine Bohrung hin-zugefügt.

Die Widerstandswerte liegen durch die verkürzten Leitungen nochmals unter denen der Vorgängerversion.

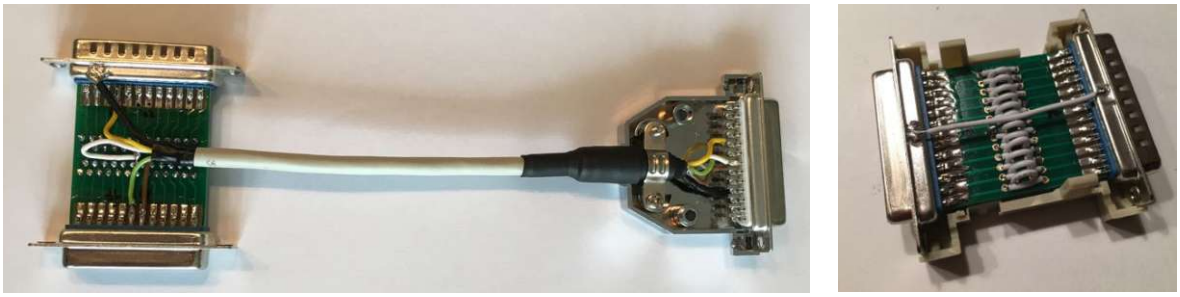


Abbildung 27: Kompakte Version des Adapters

Es sei erwähnt, dass für die Triggersteuerung des Spektroskops eine bestimmte Einstellung innerhalb der Software OPUS 6 vorgenommen werden muss. Für die Konfiguration muss das Spektroskop eingeschaltet und die Software aktiv sein. Im nächsten Schritt muss in OPUS der Menüpunkt Measure/Enhanced Measurement gefunden werden. Dann muss der Ordner „Optics parameters“ ausgewählt werden. Der nächste Schritt sieht vor, den Punkt „External synchronization“ aktiv zu setzen. Wird nun eine Messung gestartet, so erscheint ein Fenster, welches anzeigt, dass die Triggersensorik aktiv ist.

Während der Testarbeiten im Zuge des Adapterbaus wurde eines der beiden Netzteile ausgetauscht. Es wurde ein leistungsstärkeres Netzteil ausgewählt, um gegebenenfalls das zweite Netzteil, welches den Servomotor versorgt, zu umgehen. Der Austausch führte zu erheblichen Problemen, da die komplette Steuerung unverständlicherweise ausfiel. Nach eingehender Untersuchung stellte sich mit Hilfe eines Multimeters heraus, dass das NoName-Netzteil von Werk aus verpolt ausgeliefert wurde. Eine zusätzliche Kontrolle durch Öffnen des Netzteils bestätigte dies, da die Leitungen falsch herum mit der Platine verlötet wurden. Diese Verpolung führte zu einem Defekt des Arduinoboards, wodurch dieses für die weitere Benutzung innerhalb der Steuerung unbrauchbar wurde. Der Controller kann weiterhin über die ICSP-Schnittstelle programmiert werden, dies ist allerdings mit dem gewählten Aufbau nicht ohne Weiteres zugänglich, weshalb ein neues Board eingesetzt werden musste. Dadurch, dass nicht der Arduino selbst, sondern Sockelleisten mit der Platine verlötet wurden, war der Austausch der Boards problemlos möglich.

Wie bereits im vorigen Kapitel erwähnt, soll nun der Programmcode der finalen Version der Filtersteuerung näher diskutiert werden. Der Code wird in voller Länger zur besseren Übersicht im Anhang 9.3 angefügt. Einzelne Passagen des Codes werden zur Erklärung als Bildausschnitt angeführt.

Wie in fast jedem Arduino-Code werden zu Beginn die benötigten Bibliotheken geladen. Für die Software der Steuerung wurden drei „libraries“ genutzt. Die Bibliothek „wire.h“ dient der Implementierung der Kommunikation zwischen I^2C-Geräten (25). Diese ist für die Datenbusverbindung zwischen LCD und Microcontroller-Board nötig. Die Einbindung von „Servo.h“ sorgt für vorgefertigte Befehle zur Steuerung des Servoantriebs. Die Library „LiquidCrystal_I2C.h“ dient der Ansteuerung des Displays. Hierdurch müssen nicht für jedes Zeichen einzelne Pixel mittels des Programmcodes angesteuert werden, sondern es wird durch einfache Befehle möglich, Wörter und Zahlen im Programmcode als Klartext dargestellt, umzusetzen.

Abbildung 28 zeigt die Pinbelegung und Festlegung der Variablen. An Pin 2 des Nano’s wird der Trigger zur Steuerung des Spektroskops generiert, weshalb die zugehörige Variable im Code zur besseren Lesbarkeit „TRIGout“ genannt wurde. Die Variable „tasterpin“ ist eine der wenigen als Input definierten Pins und dient der Abfrage des Zustandes des Start/Stopp-Knopfes. Über „motorpin“ wird der Servoantrieb gesteuert. Die Zeile Rund um Pin 1 ist auskommentiert, da es sich hierbei um das deaktivierte Potentiometer aus Version 1 handelt. Mit „TRIGled“, „nullled“ und „neunzigled“ werden die drei Leuchtdioden angesteuert. Die Variablen „kamera“ und „kamerafokus“ werden genutzt, um in der finalen Version auch das Auslösen der Kamera bei Auslösen des Triggers zu steuern.

```
const int TRIGout = 2;
const int tasterpin = 4; // mit Spannungsteiler
const int motorpin = 3;
// const int potpin = 1; // Analog A1
const int TRIGled = 7;
const int nullled = 8;
const int neunzigled = 9;
const int kamera = 12;
const int kamerafokus = 13;
```

Abbildung 28: Ausschnitt aus Programmcode – Pinbelegung

In der nachfolgenden Passage, welche in Abbildung 29 gezeigt wird, werden einige weitere Variablen definiert, die den Status verschiedener Vorgänge zwischenspeichern. Mit der Variable „ACKin“ wird der Zustand des vom Spektroskop gesendeten Ackknowledge-Signals auf dem analogen Pin A6 des Microcontroller-Boards überwacht. Hierfür wird erneut ein 10kΩ Pullup-Widerstand notwendig. Mit Hilfe einer If-Bedingung und eines Schwellenwerts wird der analoge Wert gewissermaßen digitalisiert und liegt in binärer Form vor. Dieser übersetzte Wert wird in der Variable „ACK“ gesichert. Mit der „ACKold“ wird der Wert aus dem vorherigen Loop des Programmcodes gespeichert. Durch die Sicherung vorheriger Werte wird es möglich, auf- und absteigende Flanken softwaretechnisch zu erfassen. Folgt also auf den Wert 0 der Wert 1 so kann dies als aufsteigende Flanke erkannt werden. Setzt man außerdem die dafür notwendige Bedingung bei Erkennen der gesuchten Flanke für einige Millisekunden außer Kraft, so kann im selben Zug der analysierte Wert entprellt werden. Erklärungen und weitere Möglichkeiten zum Thema Entprellen wurden bereits in Kapitel 2.2 erläutert. Die Variablen „tasterwert“ und „oldtasterwert“ dienen selbsterklärend der Zwischenspeicherung des Status des Start/Stoppknopfes. Alle Variablen die das Präfix „old“ enthalten, sind nötig um die Flanken zu erkennen und werden nun nicht weiter berücksichtigt. Um den Startwinkel des Servomotors zu definieren wird die Variable „anfangspos“ eingeführt. Der Wert wurde hier mit 30° festgelegt, wobei dies der 0°-Position des Polarisationsfilters entspricht. Der Grund hierfür wurde bereits erläutert. Der Drehwinkel zwischen den beiden Polarisationsfilter-Positionen wird mit „winkel“ festgelegt. In der Praxis zeigte sich, dass der Wert 91 besser den tatsächlichen 90° entspricht. Mit der Variable „count“ wird der Zähler der Messdurchgänge realisiert. Der Status für die beiden Positions-LEDs wird mit den Variablen „nullledState“ und „neunzigledState“ gespeichert. Die Variable „startbutton“ wird für zwei Aufgaben verwendet. Sie verhindert einerseits, dass die Startbedingung während der Messung erneut ausgeführt werden kann, da ansonsten alle Parameter zurückgesetzt werden würden. Andererseits folgern aus dem Status dieser Variable andere wahr werdende Bedingungen, welche nach Erledigung ihres Zwecks mitunter wiederrum den Status von „startbutton“ invertieren.

```
int ACKin;
int ACK;
int oldACK;
int tasterwert; // Tasterstatusabfrage
int oldtasterwert;
int anfangspos = 30; // Anfangswinkel um mechanisches Anfahren des Servos zu verhindern
int winkel = 91; // Winkel switcht jeden Durchgang zwischen 0 und 91
int count = 0; // zählt Durchgaenge
int nullledState = HIGH; // Winkelanzeige LED 0 Grad
int neunzigledState = LOW; // Winkelanzeige LED 90 Grad
int startbutton = 0;
```

Abbildung 29: Ausschnitt aus Programmcode – weitere Variablen

Der darauffolgende Abschnitt des Codes dient der Implementierung verschiedener Variablen, die für die Zeitmessung und das Entprellen genutzt werden.
In „void setup()“ werden die verwendeten Pins den Variablen zugewiesen und der I/O-Status festgelegt. Außerdem wird der Servo eingebunden. Die benutzen LED's erfahren ihren Anfangsstatus. Abbildung 30 zeigt das Setup des LCD-Displays. Die verwendeten Befehle werden durch die bereits zuvor erwähnte, zugehörige Bibliothek möglich. Mit „lcd.begin(20, 4)“ wird angegeben, dass es sich um ein vierzeiliges Display mit jeweils 20 Zeichen handelt. Hier sei erwähnt, dass zu Beginn des Codes, unterhalb der Inkludierung der Library die I^2C-Adresse angegeben ist. Diese muss bekannt sein, um das Display ansprechen zu können. Die Adresse ist entweder durch den Hersteller bekannt oder kann mittels eines Codes innerhalb der Arduino-IDE seriell ausgelesen werden. Mit „lcd.setCursor“ wird festgelegt wo der darauffolgende Print-Befehl dargestellt werden soll.

```
lcd.begin(20, 4);  // LCD hat 4 Zeilen à 20 Zeichen
lcd.setCursor(0, 0); //Start in Zeile 0, Zeichen 0
lcd.print("ACK: X");
lcd.setCursor(0, 1);
lcd.print("Counter: 0");
lcd.setCursor(0, 2);
lcd.print("Runtime: 0 s");
```

Abbildung 30: Ausschnitt aus Programmcode – Startcode für LCD-Display

In „void loop“ folgt nun der Hauptteil des Programms. Zu Beginn wird der aktuelle Status des Start/Stopp-Knopfes in der Variable „tasterwert“ abgelegt. Da das Programm in der Endlosschleife durchgeführt wird, erfolgt die Abfrage tausende Male pro Minute. In der Passage, welche in Abbildung 31 gezeigt wird, wird die bereits

vorher näher gebrachte Digitalisierung des vom Spektroskop ausgehenden Messsignals unternommen. Liegt das eingehende Messsignal über dem Schwellenwert 1000, was in Realität im Messaufbau einer ungefähren Spannung von 5 Volt entspricht, so setzt die Bedingung die Variable „ACK“ auf den Wert 1, andernfalls auf 0. Um den momentan ermittelten Status zu visualisieren, wird dieser am Display mit den nachfolgenden Zeilen des Codes ausgegeben.

```
int ACKin = analogRead(6);
if (ACKin >= 1000) {
  ACK = 1; // Messung aktiv
  }
else {
  ACK = 0; // keine Messung
  }

lcd.setCursor(0, 0); //Start in Zeile 0, Zeichen 0
lcd.print("ACK: ");
lcd.print(ACK);
lcd.print("    ");
```

Abbildung 31: Ausschnitt aus Programmcode – ACK-Signaleingang

Abbildung 32 zeigt die if-Bedingung für das Auslösen des Versuchsprozederes. Mit „currentMillis“ und „verzoegerung“ wird das Prellen verhindert ohne auf die Funktion „delay“ zurückzugreifen. Dies hat den entscheidenden Vorteil, dass der Programmablauf nicht wie bei der delay-Funktion im entprellten Zeitraum angehalten wird. Weitere simultane Abläufe werden also nicht behindert. Für das Ausführen der Befehle innerhalb der if-Bedingung müssen vier Gleichungen zutreffen. Zuerst kann die Bedingung nur wahr werden, wenn die Entprell-Verzögerung nicht ausgelöst ist. Des Weiteren muss eine aufsteigende Flanke des Startknopfes registriert werden. Dies gelingt in dem der aktuelle Tasterwert auf den Wert 0 und der vorherige auf 1 geprüft wird. Außerdem muss die Variable „startbutton“ 0 sein, da dies gewährleistet, dass aktuell keine Messung läuft. Diese Art von Variablen werden auch „flag“ genannt. Werden alle vier Gleichungen wahr, so wird zuerst die flag auf 1 gesetzt und ermöglicht damit das Aktivwerden weiterer if-Bedingungen. Mit „starttime“ wird die Zeitmessung begonnen. Die beiden Funktionen TRIGCAMsend() und counter() werden am Ende des Codes implementiert und dienen der Kamerasteuerung und dem Zählen der Schwenkvorgänge. Diese Bedingung kann nur zu Messbeginn wahr werden und

aktiviert somit das Prozedere. Für die Kommunikation zwischen Steuerung und Spektroskop wird die Bedingung aus Abbildung 34 nötig.

```
currentMillis = millis();
if ((currentMillis - verzoegerung  >= 500) && (tasterwert == 0) && (oldtasterwert == 1) && (startbutton == 0)) {
  verzoegerung = millis();
  startbutton = 1;
  starttime = millis();
  TRIGCAMsend();
  counter();
}
```

Abbildung 32: Ausschnitt aus Programmcode – Startbedingung

Mit der nächsten Bedingung, zu sehen in Abbildung 33, wird die Starttaste während des Versuchs als Stoppknopf umfunktioniert. Wieder müssen vier Gleichungen erfüllt sein. Wie zuvor wird die Entprellung überprüft. Außerdem muss wieder eine aufsteigende Flanke, also das Drücken des Knopfes, erkannt werden. Zuletzt muss die flag „startbutton“ 1 sein. Treffen alle Bedingungen zu, so wird die flag auf den Wert 2 gesetzt. Diese löst in weiterer Folge das Stoppen des Prozesses aus, da die Bedingung aus Abbildung 34 nicht mehr erfüllt ist. Mit „oldtime“ wird die Zeit für das Einlegen einer Messpause zwischengespeichert.

```
if ((currentMillis - verzoegerung  >= 500) && (tasterwert == 0) && (oldtasterwert == 1) && (startbutton == 1)) {
  verzoegerung = millis();
  startbutton = 2; // 0 um startknopf zu reaktivieren
  oldtime = time;
}
```

Abbildung 33: Ausschnitt aus Programmcode – Stoppknopf

Der Steuerungsmechanismus bzw. die Kommunikationslösung werden in Abbildung 34 gezeigt. Wird das Ende einer Messung durch das Erkennen einer absteigenden Flanke des ACK-Signals detektiert und ist auch die Software gerade im aktiven Modus, also „startbutton“ mit dem Wert 1, so werden folgende Befehle ausgeführt. Zuerst wird um 1300 ms verzögert, da ansonsten das gesendete Triggersignal nicht erkannt werden kann. Dies wurde bereits näher diskutiert. Danach wird der Servoantrieb geschwenkt, der Winkel wird durch die Funktion switcher() periodisch zwischen 0 und 90° getauscht. In den nächsten Zeilen werden die Positionsleuchtdioden umgeschaltet. Der Status der LED’s wird ebenfalls durch die Funktion switcher() pro Schwenkvorgang geändert. Es folgt ein weiterer delay um dem Motor Zeit zu geben

seine vorgegebene Position zu finden. Danach findet mit der Funktion TRIGCAMsend() die Aussendung des Triggers statt.

```
// Steuerung
if ((ACK == 0) && (oldACK == 1) && (startbutton == 1)) { // keine Messung, Start gedrückt
    delay(1300); // verzögerung, da datei auf spektroskop noch nicht erscheint
    servo1.write(anfangspos + winkel); // Servostellung 0°
    counter();
    switcher();
    digitalWrite(nullled, nullledState);
    digitalWrite(neunzigled, neunzigledState);

    delay(1200); // Zeit um Motor zu stellen
    // ack=0 ABER Messung SYNC Fenster noch nicht bereit; 2000ms funktioniert nicht

    TRIGCAMsend();
}
```

Abbildung 34: Ausschnitt aus Programmcode – Steuerung

In Abbildung 35 wird die Implementierung der Zeitmessung gezeigt. Durch die Gleichung „currentMillisT – previousMillisT >= 100“ wird die Zeitmessung alle 100 ms aktualisiert, wenn die flag „startbutton“ eine aktive Messung anzeigt. Die Formel aus der vierten Zeile der Abbildung berechnet die am Display anzuzeigende Zeit. Für die Berechnung wird die Funktion millis(), welche auch bereits für die Entprellung verwendet wurde, verwendet. Die vorgefertigte Funktion aus der Arduino IDE gibt die vergangene Zeit in Millisekunden wieder, welche seit Start des Programms vergangen ist (26). Mit einem Reset bzw. durch eine Stromtrennung wird dieser Wert zurückgesetzt. Beim Auslösen des Startknopfes wird aus diesem Grund der zu diesem Zeitpunkt aktuelle Wert der Funktion in der Variable „starttime“ gesichert. Zieht man diese nun von dem aktuellen Wert der Funktion ab, so erhält man die vergangene Zeit in Millisekunden seit Auslösen der Taste. Falls eine Messung mit Pause durchgeführt wird, speichert „oldtime“ die Zeit des vorherigen Durchgangs. Durch Hinzuaddieren wird damit die Gesamtzeit ermittelt. Ohne Pause bleibt der Wert von „oldtime“ null. Um den ermittelten Wert in Sekunden darzustellen wird dieser in der nachfolgenden Codezeile umgerechnet. Die darauffolgenden Zeilen des Programms bilden die gemessene Zeit in der dritten Zeile des Displays ab.

```
// Zeit
currentMillisT = millis();
if ((currentMillisT - previousMillisT >= 100) && (startbutton == 1)) {
  previousMillisT = currentMillisT;
  time = oldtime + millis() - starttime;
  timedisplay = (double)time / 1000;

  lcd.setCursor(0, 2);
  lcd.print("Runtime: ");
  lcd.print(timedisplay, 1); // 3 nachkommastellen
  lcd.print(" s  ");
   }
```

Abbildung 35: Ausschnitt aus Programmcode – Zeitmessung

Abbildung 36 zeigt die Funktion counter(), welche dafür sorgt die Schwenkvorgänge der Steuerung zu zählen. Die Schritte werden in der Variable „count“ gezählt. Bei jedem Aufruf der Funktion wird der Wert der Variable durch „count++“ um einen Zähler erhöht. Die weiteren Befehle dienen erneut der Darstellung am Display, in diesem Fall in der zweiten Zeile in Spalte 10.

```
void counter() { // Schrittzähler
  count++;
  lcd.setCursor(9, 1);
  lcd.print(count);
}
```

Abbildung 36: Ausschnitt aus Programmcode – Funktion counter()

Mit der Funktion switcher(), welche in Abbildung 37 gezeigt wird, werden Zustände der wechselnden Variablen geändert. In der oberen Passage des Codes wird der Winkel des Servoantriebs getauscht. Der Wechsel zwischen den beiden Winkeln wird mittels einer if-Bedingung vollzogen. Dabei wird überprüft, ob der Variablenwert aktuell 0 entspricht. Stimmt dies zu, so wird der Wert auf 91 geändert. Andernfalls wird der Wert auf 0 geändert, da der Wert in diesem Fall aktuell 91 sein muss. Die Variable kann codetechnisch keinen anderen Wert annehmen als 0 oder 91.
Analog erfolgt die Umstellung der LED-Zustände. Ist der Status der 0°-LED aktuell LOW so wird dieser auf HIGH geändert. Trifft dies nicht zu, so ist dieser aktuell HIGH und wird auf LOW getauscht. Selbiges findet für den Tausch des Zustandes der 90°-LED statt.

Für den synchronen Ablauf der Zustände sind alleinig die Anfangszustände wichtig. Diese müssen passend zueinander gewählt werden. Der Anfangswert für den Servo muss also 0 sein, während die 0°-LED aktiv und die 90°-LED inaktiv sein muss. Die Funktion switcher() regelt das Restliche.

```
void switcher() {
    if (winkel == 0) // switch winkel
      winkel = 91; // entspricht 90° auf zahnrad
    else
      winkel = 0;

    if (nullledState == LOW) // switch LEDstate
      nullledState = HIGH;
    else
      nullledState = LOW;

    if (neunzigledState == LOW) // switch LEDstate
      neunzigledState = HIGH;
    else
      neunzigledState = LOW;
  }
```

Abbildung 37: Ausschnitt aus Programmcode – Funktion switcher()

Mit der am Ende des Codes stehenden Funktion TRIGCAMsend(), welche in Abbildung 38 dargestellt wird, erfolgt die Steuerung des Kamera und die Aussendung der Triggers zur Steuerung des Spektroskops. Es wurde beobachtet, dass der Fokus vor Auslösen der Kamera aktiviert werden muss, um ein Fehlverhalten, welches zu fehlenden Bildern führen würde, bei der Aktivierung des Auslösers zu verhindern. Deshalb wird im Code 100 ms vor Auslösen der Fokus deaktiviert. Um die Kamera für die nächste Aufnahme bereit zu machen, wird der Fokus nach dem Auslösen wieder aktiviert und erst wieder beim nächsten Aufruf der Funktion deaktiviert. Mit dieser Vorgehensweise funktioniert die Kamerasteuerung zuverlässig. Die Auslösezeit wurde mit 10 ms festgelegt.

```
void TRIGCAMsend() { // Trigger und Kamera
  digitalWrite(kamerafokus, LOW);
  delay(100);
  digitalWrite(kamera, HIGH);
  delay(10);
  digitalWrite(kamera, LOW);
  delay(10);
  digitalWrite(kamerafokus, HIGH);

  digitalWrite(TRIGout, LOW); // Trigger gesendet
  digitalWrite(TRIGled, HIGH);
  delay(10); // Triggersendedauer
  digitalWrite(TRIGout, HIGH);
  digitalWrite(TRIGled, LOW);
  }
```

Abbildung 38: Ausschnitt aus Programmcode – Funktion TRIGCAMsend()

Im zweiten Abschnitt der Funktion liegt das eigentliche Herzstück des Codes, die Zeilen zur Aussendung des spektroskopsteuernden Triggers. Da das Spektroskop einen Spannungsabfall als Triggerimpuls erkennt, wird die Variable „TRIGout“ für 10 ms auf LOW geschaltet. Das Spektroskop erkennt diesen Abfall über die galvanisch getrennte Verbindung und löst die Messung aus. Gleichzeitig wird auch die Status-LED der Steuerung zur Anzeige des ausgesendeten Triggers für 10 ms angesteuert und blinkt damit kurz auf. Damit wird das erfolgreiche Aussenden des Triggers visualisiert.

Bevor die Messergebnisse dargestellt werden, folgt vorerst eine Erklärung zur Auswertung der mit Version 2 der Filtersteuerung aufgenommen Rohdaten.

Zur Auswertung stehen nach erfolgreicher Versuchsdurchführung die Messdaten des Spektroskops und der Zugprüfmaschine sowie die Bilder der Proben zur Bestimmung der Dehnung zur Verfügung.

Da die Messdaten der Zugprüfeinrichtung als Klartext innerhalb eines Textfiles ausgegeben werden, fällt es nicht schwer diese weiter zu verarbeiten. Um die erforderlichen Spannungs-Dehnungsdiagramme aus den Rohdaten zu extrahieren wurde Microsoft Excel genutzt. Zur Bestimmung der Dehnung werden die aufgenommenen Bilder des Versuchsbereichs der Probenkörper mit der Software AxioVision ausgemessen und katalogisiert. Dabei wird zuerst der parallele Probenbereich gemessen, welcher die Ausgangslänge zur Berechnung der Dehnung angibt. Vorab auf den Probenkörpern aufgebrachte schwarze Markierungen sollen das Messen vereinfachen und dazu dienen, dieselben Messpunkte einzuhalten. Mittels der Software wird die Pixelanzahl zwischen den Messpunkten gemessen. Die manuelle

Auswertung der Bilder ergibt erneut eine Excel-Liste mit den aufgenommenen Längen. Mit der Formel

$$(Aktuelle\ Länge - Ausgangslänge) / Ausgangslänge * 100$$

ergibt sich die Dehnung in Prozent und steht damit zur weiteren Auswertung bereit.
Die gewonnenen Daten aus den FTIR-Messungen liegen roh nicht im Klartext vor und müssen deshalb innerhalb der Software OPUS 6.0 in Datenpunkttabellen übersetzt werden. Innerhalb des Programms ist es nicht möglich, mehrere Dateien simultan zu konvertieren, weshalb manuell jede einzelne Datei als DPT-Datei gesichert werden muss. Um nicht für jede Dateisicherung einen neuen Namen vergeben zu müssen, empfiehlt es sich die Rohdateien vorerst mit einem Tool, wie z.B. Joe 4.0, umzubenennen. Die Spektroskopsoftware sichert die Rohdaten mit fortlaufender Nummer als Dateiendung. Diese fortlaufende Zahl wird bei der direkten Konvertierung zur Datenpunkttabelle überschrieben. Aus diesem Grund muss der Dateiname geändert werden. Um den Konvertierungsprozess zu verkürzen, empfiehlt es sich aus ProbeX.4 die Datei ProbeX_4.4 zu erstellen. Dadurch geht die Nummerierung bei der zügigen manuellen Konvertierung zu DPT nicht verloren. Umbenennungstools wie Joe 4.0 können dabei die vorherige Dateiendung erkennen und in den neuen Dateinamen schreiben. Nachdem die Daten als Datenpunkttabellen vorliegen können diese mit dem in Kapitel 2.5 beschriebenen Tool automatisiert ausgewertet werden. Dieses berechnet die Orientierungsgrade und gibt diese als xls-Datei aus. Die Orientierungsgrade werden in weiterer Folge den zugehörigen ermittelten Dehnungen zugeordnet und dargestellt. Ohne Kamerasteuerung erfolgt die Zuordnung von Orientierungsgrad zu Dehnung über die Zeitachse. Mittels der Kamerasteuerung ist dies nicht mehr notwendig, da das zugehörige Foto zur Bestimmung der Dehnung zum Zeitpunkt der FTIR-Messung aufgenommen wird.
Grundsätzlich liefert die beschriebene Mess- und Auswertemethode überwiegend plausible Ergebnisse. Innerhalb dieser Versuchsreihe, welche bei Probe 7 startet und bei Probe 14 endet, wurden neun Proben geprüft. Die Versuche zu Probe 8, Probe 13 und Probe 15 scheiterten aufgrund von Problemen mit der Zugprüfeinrichtung. Es sei anzumerken, dass es bei den Messungen sporadisch zu Ausfällen der Kamera kam. Dadurch wurde es nötig, die Auswertungen manuell nachzubessern bzw. die fehlenden Werte zu interpolieren. Durch die Kamerasteuerung, welche im folgenden

Kapitel erläutert wird, entfällt diese Unsicherheit. Abbildung 39 und Abbildung 40 zeigen typische Ergebnisse aus den Messreihen mit Schwenkautomatik und ohne Kamerasteuerung. Die Kurvenverläufe folgen weitgehend jenen, welche aus der Literatur bekannt sind (siehe Abbildung 17).

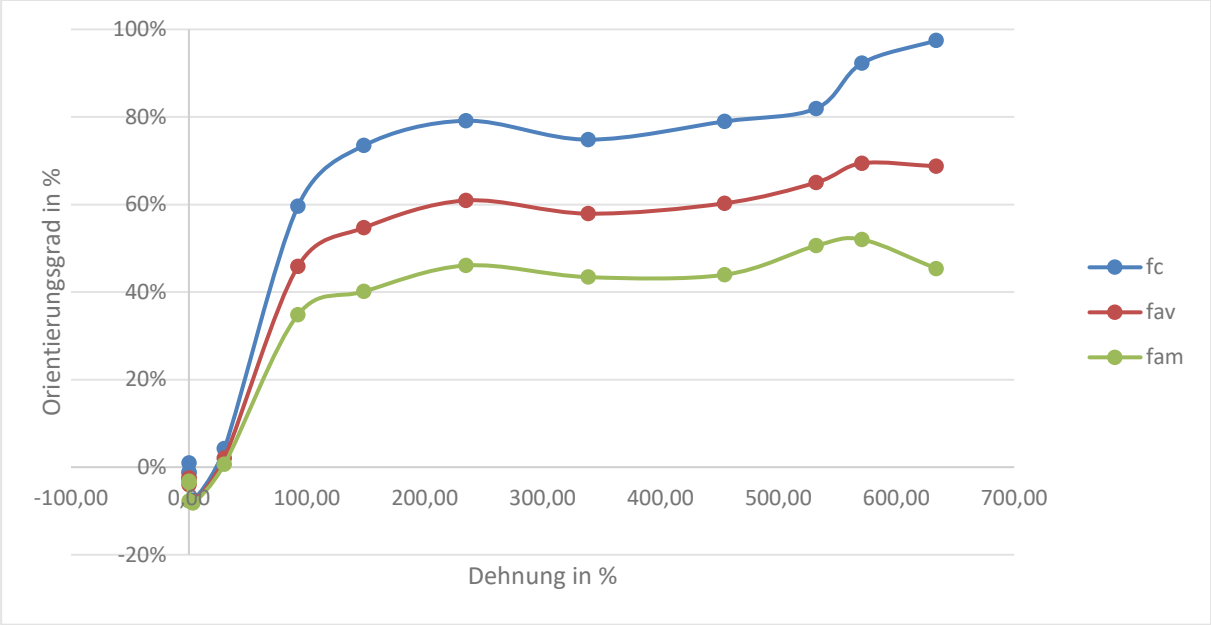


Abbildung 39: Orientierungsgrad über Dehnung - Probe 7

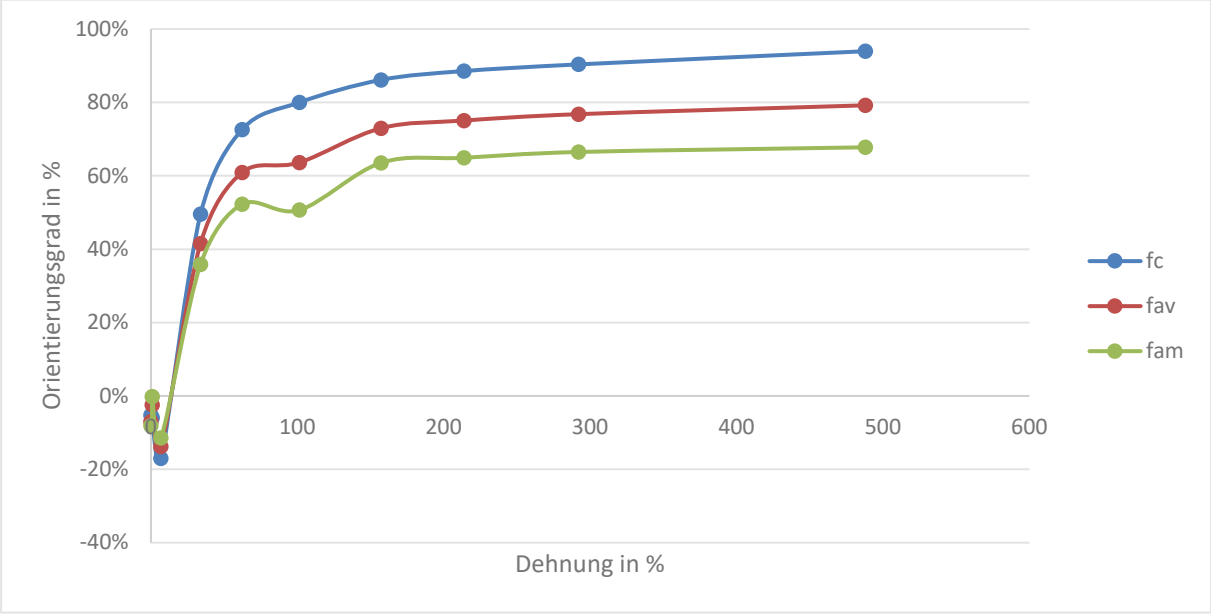


Abbildung 40: Orientierungsgrad über Dehnung - Probe 9

Nach der Realisierung der Kamerasteuerung entstand die Idee für eine neue Auswertemethode. Bei der bisherigen Auswertung wurden zwei aneinander folgende

FTIR-Messergebnisse, welche durch die Steuerung mit 0° und 90° Polarisationsfilter aufgenommen wurden, verwendet um den Orientierungsgrad zu bestimmen. Bei der Auswertung fiel auf, dass die Auslösung der aufgenommenen Punkte vor allem im Anfangsbereich gering ausfällt, wodurch für die Darstellung des steilen und interessanten Anstiegs zu Beginn nur wenige Datenpunkte zur Verfügung stehen. Die Versuchsgeschwindigkeit wurde nicht weiter reduziert, um einerseits die bisher aufgenommen Daten vergleichbar zu halten und diese andererseits ohnehin bereits auf ein Minimum reduziert wurden. Auch die Erhöhung der Messgeschwindigkeit des Spektroskops ist keine Option, da eine weitere Reduzierung der Scananzahl zu keiner schnelleren Messrate führte. Dies wurde bereits näher diskutiert.

Aus diesen Gründen werden für die verbesserte Auswertemethode die Spektroskopiedaten interpoliert bzw. für die Randwerte extrapoliert. Das bedeutet, dass für eine 0°-Messung eine interpolierte 90°-Messung zur Bestimmung des Orientierungsgrades berechnet wird. Dasselbe gilt für 90°-Messungen. Der jeweilige interpolierte Wert ergibt sich aus den vorigen bzw. nachherigen ermittelten Daten. Zur Bestimmung wird außerdem die Dehnung ε zum Zeitpunkt der zugehörigen, tatsächlichen Messung benötigt. Start bzw. Endwerte werden durch Extrapolation ermittelt. Die Formeln werden mit Abbildung 41 hergeleitet:

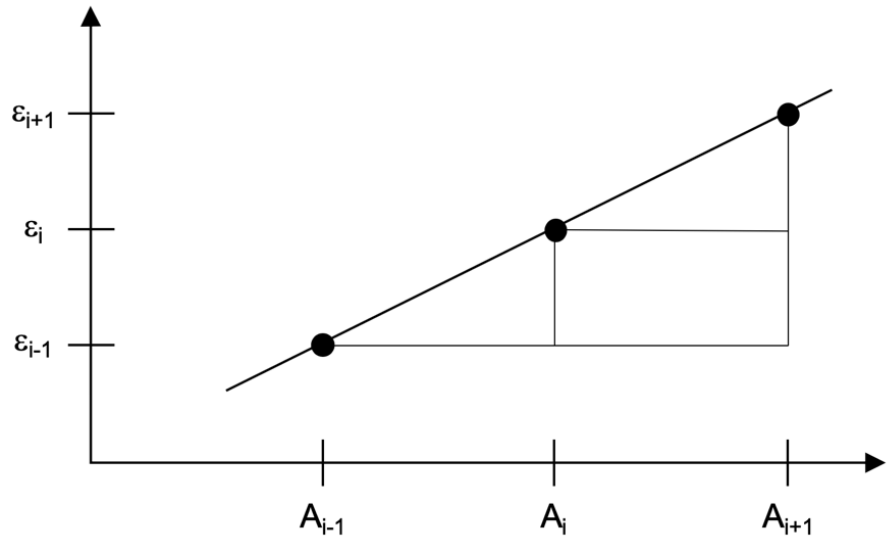


Abbildung 41: Darstellung der Variablen zur Bestimmung der Formeln (Interpolation, Extrapolation)

Aus Abbildung 41 ergeben sich folgender Zusammenhänge:

$$\frac{\varepsilon_{i+1} - \varepsilon_{i-1}}{A_{i+1} - A_{i-1}} = \frac{\varepsilon_i - \varepsilon_{i-1}}{A_i - A_{i-1}}$$

Formel 11: Formaler Zusammenhang für Interpolation/Extrapolation

$$\frac{\varepsilon_{i+1} - \varepsilon_{i-1}}{A_{i+1} - A_{i-1}} = \frac{\varepsilon_{i+1} - \varepsilon_i}{A_{i+1} - A_i}$$

Formel 12: Formaler Zusammenhang für Extrapolation des Anfangswerts

Durch Umformung von Formel 11 erhält man den durch Interpolation bestimmten Amplitudenwert A_i:

$$A_i = \frac{A_{i+1} - A_{i-1}}{\varepsilon_{i+1} - \varepsilon_{i-1}} (\varepsilon_i - \varepsilon_{i-1}) + A_{i-1}$$

Formel 13: Interpolation der Spektroskopiedaten

Die extrapolierten Amplitudenwerte, welche zur Bestimmung der Anfangs- oder Endwerte benötigt werden, werden mit den unten angeführten Formeln berechnet. Der Zusammenhang aus Formel 12 dient hier der einfacheren Umformung nach A_{i-1}.
Aus Formel 11 ergibt sich für den Endwert A_{i+1}:

$$A_{i+1} = \frac{A_i - A_{i-1}}{\varepsilon_i - \varepsilon_{i-1}} (\varepsilon_{i+1} - \varepsilon_{i-1}) + A_{i-1}$$

Formel 14: Extrapolation Endwert der Spektroskopiedaten

Mit Formel 12 erhält man direkt den Anfangswert A_{i-1}:

$$A_{i-1} = \frac{A_i - A_{i+1}}{\varepsilon_{i+1} - \varepsilon_i} (\varepsilon_{i+1} - \varepsilon_{i-1}) + A_{i+1}$$

Formel 15: Extrapolation Anfangswert der Spektroskopiedaten

Die Indizes *i-1*, *i*, und *i+1* sind für die obigen Formeln chronologisch festgelegt. Auf *i-1* folgt also *i*. Auf *i* folgt *i+1*. Zur Bestimmung des Endwerts mit Formel 14 müssen die beiden bekannten vorherigen Werte *i-1* und *i* herangezogen werden. Analog dazu müssen für einen unbekannten Anfangswert mit Formel 15 die nachfolgenden Werte mit den Indizes *i* und *i+1* herangezogen werden. Für die Interpolation müssen die beiden Randwerte *i-1* und *i+1* zur Berechnung mit Formel 13 herangezogen werden. Abbildung 42 und Abbildung 43 zeigen die bereits gezeigten Ergebnisse zu den Proben 7 und 9 mit der verbesserten Auswertemethode. Die Erhöhung der Messauflösung ist deutlich zu erkennen. Durch die Interpolation wird die Anzahl der Messpunkte verdoppelt. In Abbildung 42 fällt der außergewöhnliche Anstieg am Ende der Kurven auf. Während die Extrapolation in Abbildung 43 zu plausiblen verwertbaren Ergebnissen führt, scheitert diese in Abbildung 42. In diesem Fall sollten die letzten Werte für die Darstellung manuell entfernt werden, da diese nicht der Realität entsprechen. Für diese Erläuterung wurden die Werte nicht entfernt.

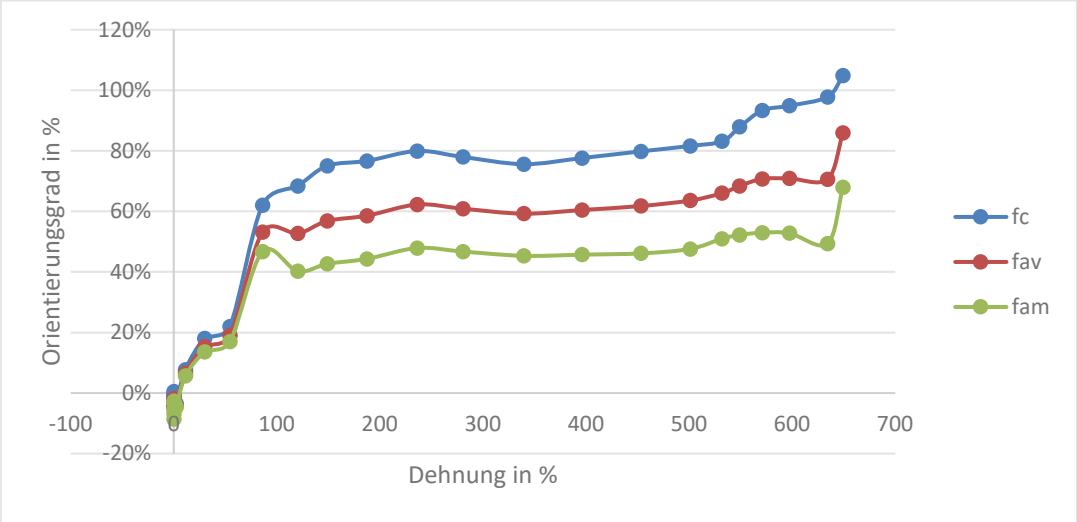


Abbildung 42: Orientierungsgrad über Dehnung - Probe 7 - Methode 2

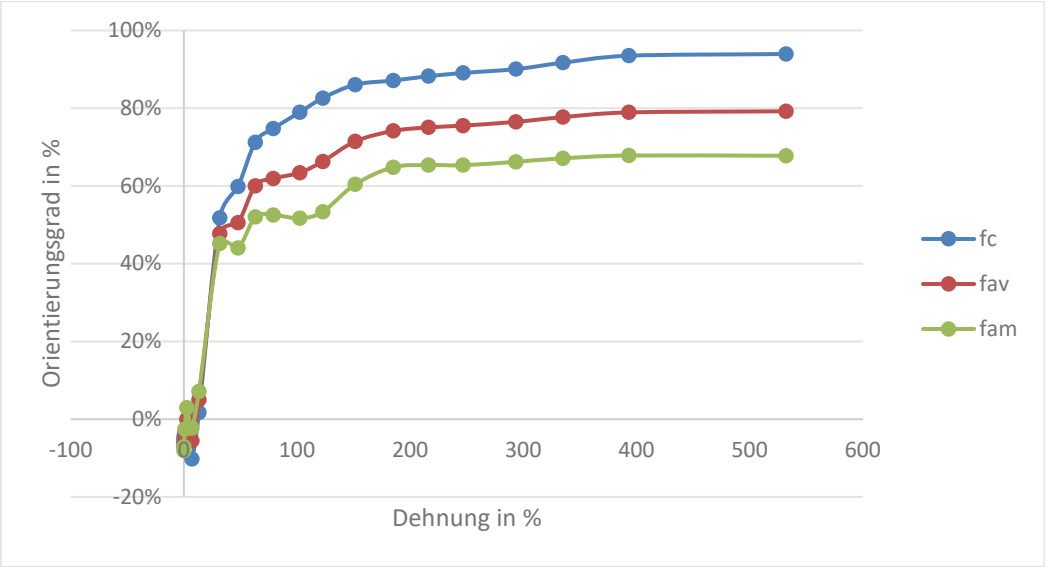


Abbildung 43: Orientierungsgrad über Dehnung - Probe 9 - Methode 2

Nachfolgend werden in Abbildung 44 und Abbildung 45 die zu den Proben 7 und 9 zugehörigen Spannungsverläufe dargestellt.

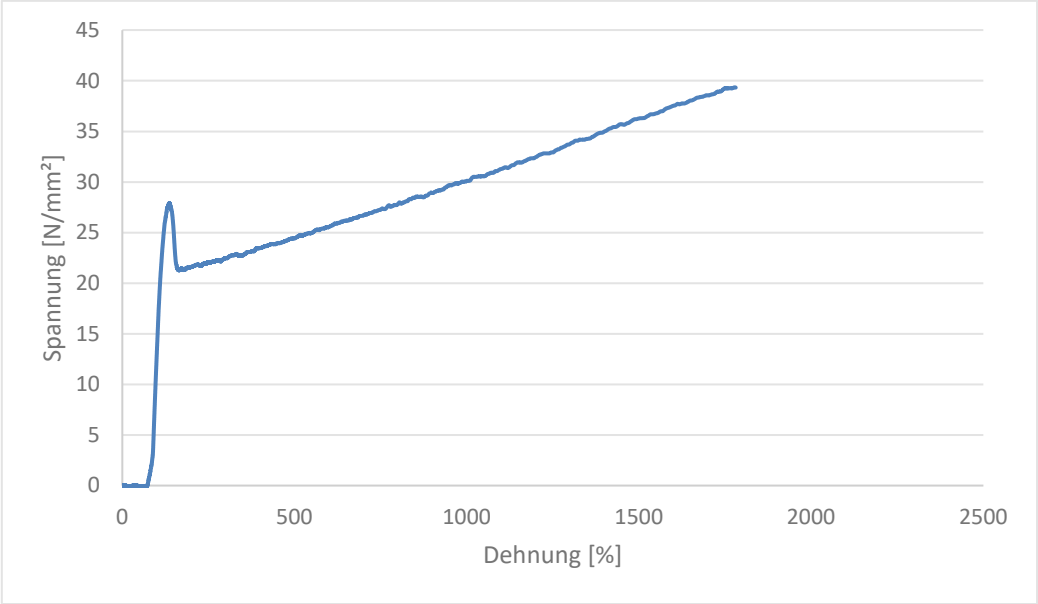


Abbildung 44: Spannungs-Dehnungsdiagramm - Probe 7

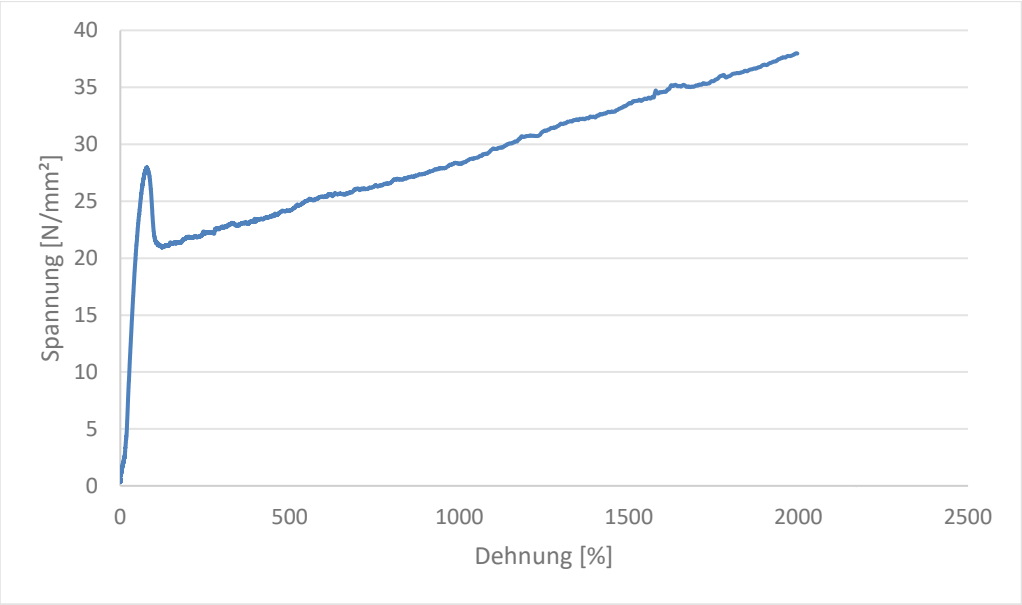


Abbildung 45: Spannungs-Dehnungsdiagramm - Probe 9

2.4. Version 2.1 – Kameratrigger

Durch die Version 2 der Schwenkautomatik wurde sichergestellt, dass Spektroskop und Steuerung synchronisierte Messdaten bereitstellen. Da für die Auswertung der Dehnungsrate eine auf den Probenkörper gerichtete Kamera, welche im Fünf-Sekundenintervall Fotos schießt, notwendig ist, wurde die Möglichkeit ins Auge gefasst, auch dies automatisiert und synchronisiert umzusetzen. Dies ergäbe eine deutliche Vereinfachung und Zeitersparnis für die Auswertung der Daten. Um die Dehnung für einen bestimmten Messzeitpunkt der Messungen des FTIR-Spektroskops zu bestimmen, mussten bisher die Werte der beiden Zeitstrahlen herangezogen werden. Der Dehnungswert wurde dann durch Interpolation bestimmt. Hierfür wurde der Messzeitpunkt der spektroskopischen Messungen herangezogen, um dann den dazu passenden Dehnungswert innerhalb der beiden zeitlich benachbarten Dehnungswerte zu bestimmen. Durch die Einführung einer triggergesteuerten Kamera, würde die Angleichung der Zeitstrahlen wegfallen, da die aufgenommen Bilddaten zeitlich zu den Messdaten des Spektroskops passen würden.

Die Nutzung einer kabelgebundenen Fernbedienung der Kamera stellte klar, dass externe Signale grundsätzlich zur Steuerung dienen können. Eine nähergehende Untersuchung dieser Fernbedienung mittels Multimeter diente der Entwicklung der eigenständigen Steuerung. Bei der verwendeten Canon Spiegelreflexkamera können

externe Signale über eine dreipolige 2,5mm-Klinkenbuchse verarbeitet werden. Diese Kabel sind kostengünstig im Elektrofachhandel zu erwerben. Durch die Untersuchung wurde klar, dass alle drei Pole des Kabels in Verwendung sind. Die drei Anschlüsse dienen als Masseanschluss, Fokus und Auslöser. Dabei wurde herausgefunden, dass die Kabel durch Kurzschluss, z.B. mittels eines Tasters, zu der gewünschten Funktion führen. Wird also das Massekabel mit dem Auslösekabel verbunden, so wird bei eingeschalteter Kamera ein Foto ausgelöst. Analoges gilt für den Fokus. Für die Umsetzung in der Schwenksteuerung kann dies technisch mit Hilfe eines Transistors realisiert werden. Der Stromlaufplan in Abbildung 22 zeigt bereits die Elemente zur Steuerung der Kamera. An Pin D12 kann mittels des Programmcodes der Fokus ausgelöst werden. Analog dient Pin D13 der Kontrolle des Auslösers. Im Stromlaufplan ist erkennbar, dass beide Pins zu den Basis-Pins zweier Transistoren führen. Die beiden Emitter-Pins der Transistoren werden mit dem Masseanschluss des Kamerakabels und der Steuerung verbunden. Die jeweiligen Collector-Pins führen zu dem Fokus- bzw. Auslösekabel der Kamera. Wird der Basis-Pin eines Transistors nun durch die Steuerung bestromt, so wird der Transistor leitfähig bzw. niederohmig. Dies entspricht näherungsweise dem zuvor beschriebenen Kurzschluss der Leitungen und löst damit kameraintern das jeweilig erwünschte Signal aus. Bereits ein kurzer Impuls reicht aus, um die Kamera zu steuern. Im Versuch stellte sich heraus, dass ein Triggersignal mit einer Dauer von 10 ms immer erkannt wird und damit den Zweck erfüllt.

Da der Fokus manuell vor dem Versuchsstart eingestellt wird, wäre grundsätzlich keine Steuerung des Fokus notwendig. Es stellte sich allerdings heraus, dass die Kamera trotz eingelangten Auslösesignals sporadisch nicht auslöste. Dies konnte behoben werden, in dem zwischen jedem Auslösesignal ein Fokussignal gesendet wurde. Durch dies dürfte die Kamera wieder in Bereitschaft gesetzt werden. Hardwareseitig mussten für die Umsetzung der Kamerasteuerung dem Filtermotor-Steuergerät also zwei Transistoren und die entsprechenden, bereits erwähnten Leitungen hinzugefügt werden. Außerdem muss ein ausreichend langes, dreipoliges Kabel aus der Steuerung zur Kamera geführt werden.

Hardwaretechnische Umbaumaßnahmen sowie die Implementierung der Kamerasteuerung innerhalb des Programmcodes wurden bereits in Kapitel 2.3 berücksichtigt. Wie bereits in Kapitel 2.3 erwähnt, wurde eine verbesserte Auswertemethode entwickelt. Deshalb werden die Ergebnisse aus der vorherigen Auswertemethode nicht

angeführt, da die verbesserte Methode eine errechnete, höhere Messauflösung bietet. Innerhalb dieser Versuchsreihe wurden gesamt zehn Proben an zwei Versuchstagen zu je fünf Versuchsdurchführungen geprüft. Am ersten Prüftag wurden die Proben 16 bis 20 geprüft. Die Versuche zu Probe 17 und 18 schlugen fehl. Der Grund liegt in der fehlerhaften Aufzeichnung der Absorbationsbanden des FTIR‘s. Der Ausfall kann entweder durch einen Bedienungsfehler oder durch technisches Versagen, welches evtl. mit der erhöhten Innenraumtemperatur an diesem Tag zusammenhängen könnte, erklärt werden.

Die später erfolgten Versuche dienen der Evaluierung der vorherigen Ergebnisse und neuerlichen Überprüfung des gesamten Ablaufs auf Grund der hohen Ausfallsrate der letzten Versuchsdurchführung. Die nachfolgenden Versuche konnten weitgehend problemlos durchgeführt werden.

Nachfolgend werden in Abbildung 46 und Abbildung 47 die Diagramme zu Proben 16 und 19 gezeigt. Während die Ergebnisse zur Probe 17 denen der Literatur entsprechen, fällt bei Probe 19 auf, dass die Werte des Orientierungsgrades deutlich geringer ausfallen. Selbiges wurde auch bei Probe 20 beobachtet. Auch aus diesem Grund wurden erneut Versuche durchgeführt. Da die Spektroskopdaten aus Probe 17 und 18 zweifelsfrei fehlerbehaftet sind, sind auch die weiteren Ergebnisse aus dieser Versuchsreihe in Frage zu stellen.

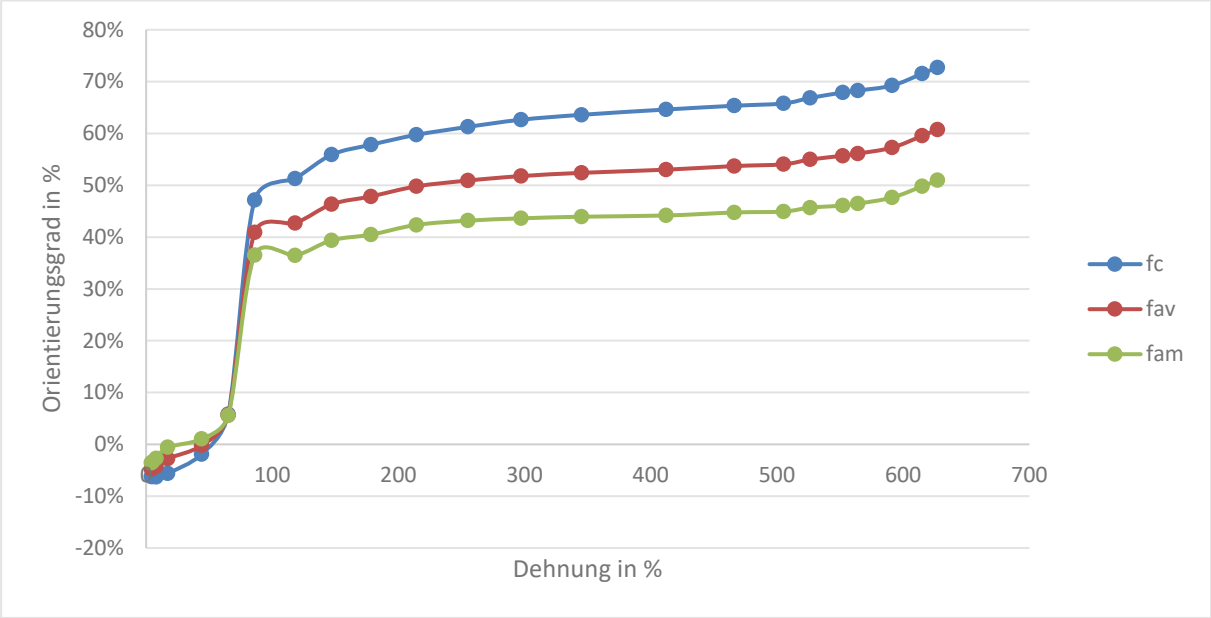


Abbildung 46: Orientierungsgrad über Dehnung - Probe 16 - Methode 2

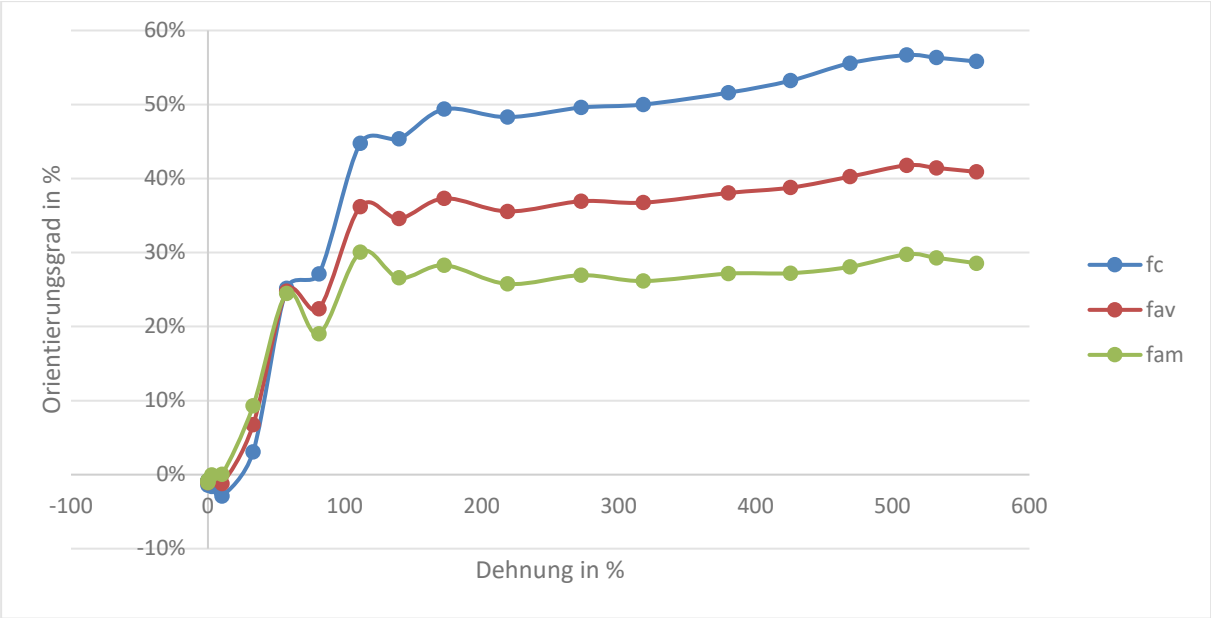


Abbildung 47: Orientierungsgrad über Dehnung - Probe 19 - Methode 2

Anschließend werden die zugehörigen Spannungs-Dehnungsdiagramme zu den Proben 16 und 19 in Abbildung 48 und Abbildung 49 dargestellt.

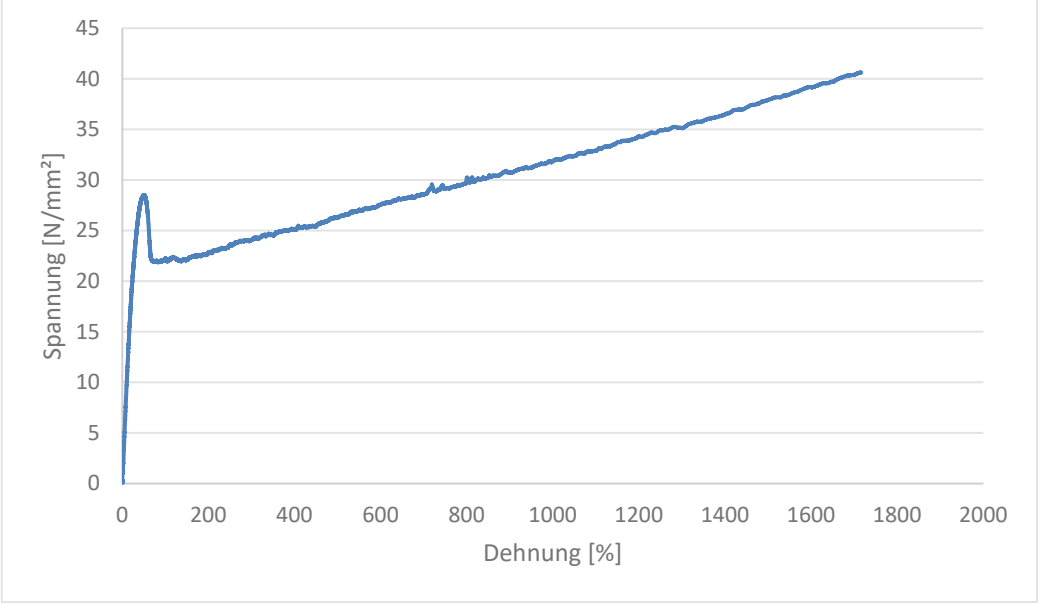


Abbildung 48: Spannungs-Dehnungsdiagramm - Probe 16

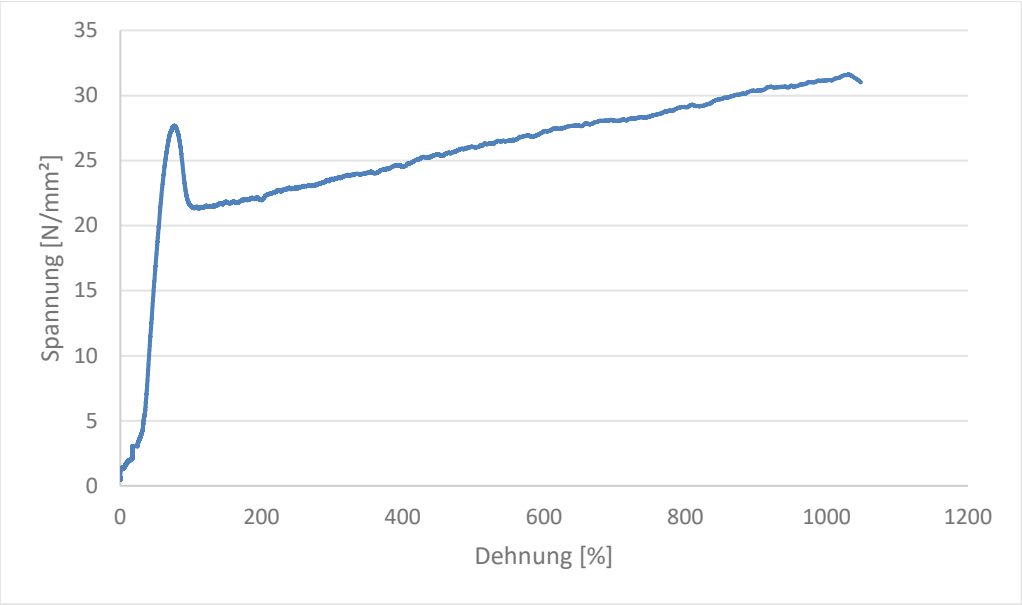


Abbildung 49: Spannungs-Dehnungsdiagramm - Probe 19

2.5. DPT-Auswerter (Python-Tool)

Wegen eines Totalausfalles der Festplatte des Spektroskop-Computers gingen die exakten Einstellungen der FTIR-Messungen verloren. Deshalb musste ein neues Einstellungsprofil angelegt werden, welches aus der Erinnerung heraus erstellt werden musste.

Da das bestehende, mittels LabVIEW programmierte, Auswertetool mit den neuen Messdaten zu offensichtlich falschen Ergebnissen führte und ein Eingriff in die objektorientierte Programmierung durch mangelnde Kenntnisse nicht möglich war, wurde ein eigenes Tool mit Python programmiert.

Im Laufe der Auswertungsarbeit, allerdings nachdem bereits die Messungen durchgeführt wurden, wurden die rohen, nicht in das dpt-Format übersetzten, Messdaten näher untersucht. Dabei fiel auf, dass innerhalb der Datei die Messdaten in verschlüsselter Form vorliegen. Am Ende der Datei werden jedoch die Messeinstellungen als Klartext gespeichert. Dadurch wurde es möglich, die vor dem Datenverlust verwendeten Einstellungen mit denen für die abschließenden Messungen verwendeten zu vergleichen. Beide Settings sind weitgehend ident, womit davon ausgegangen werden kann, dass diese vergleichbar sind. Eine vollständige Liste der ungefähr 170 Einstellungsmöglichkeiten wird auf Grund ihrer schlechten, durch Abkürzungen

kryptischen Lesbarkeit nicht angefügt. Die durch Vergleich beider Listen erkannten Unterschiede werden in Anhang 9.5 angehängt.

Der Grund für das Versagen des bereits vorhandenen Tools zur Auswertung könnte an der verschieden gewählten Auflösung liegen. Vergleicht man die Auflösungseinstellungen der alten und neuen Einstellungen so fällt auf, dass die neuen Auflösungswerte nun dem doppelten des alten Einstellungswertes entsprechen. Die Daten dazu sind im entsprechenden Anhang unter der Abkürzung RES, wie Resolution, angeführt. Bevor die Erstellung eines neuen Auswertungs-Tools in Betracht gezogen wurde, wurden die Einstellwerte des vorhandenen LabVIEW-Tools verändert, um eine eventuelle Auswirkung auf die errechneten Ergebnisse zu untersuchen. Dabei wurde herausgefunden, dass die oberen eingestellten Grenzwerte für die Auswertung nun ungerade sein mussten, um Ergebnisse generieren zu können. Dies könnte auf einen Fehler innerhalb des Codes hinweisen, welcher allerdings wegen der objektorientierten Programmierung nicht entdeckt werden konnte.

Um die Auswertung ohne Rahmenbedingungen, wie das Setzen einer ungeraden Obergrenze, auch für nachfolgende Auswertungen zu gewährleisten, wurde ein auf Python basierendes Tool programmiert.

Die Auswertung mit diesem Tool funktioniert für alle aufgenommenen Messdaten problemlos. Um das Auswertungstool zu validieren, wurden bereits ermittelte Ergebnisse mit den neu errechneten Werten abgeglichen. Der Vergleich ergab idente Ergebnisse. Die Ergebnisse des Python-Tools sind damit gleichwertig, bieten gleichzeitig jedoch einige Vorteile bei der Bedienung. Beispielsweise muss im Vorfeld nicht die genaue Anzahl der auszuwertenden DPT-Daten manuell eingegeben werden. Des Weiteren werden die Ergebnisse direkt in eine Excel-Datei abgelegt, welche automatisch im Ordner der auszuwertenden DPT-Dateien gespeichert wird. Außerdem wird die lizenzpflichtige und speicherplatzraubende Software LabVIEW für die Auswertung nicht benötigt. Mit dem Einsatz von Python werden also Kosten und Speicherplatz gespart. Ein späterer Eingriff durch Dritte in den Code ist durch die textbasierte Programmiersprache ebenfalls einfacher möglich.

Voraussetzung für die Verwendung des Tools ist Python 3.6 oder höher. Zusätzlich muss die Library „xlsxwriter“ installiert werden, um die Ausgabe der Excel-Dateien zu ermöglichen. Der Programmcode wird in Anhang 9.6 gezeigt.

Durch den Einsatz des Tools wird eine besonders zeitaufwändige, manuelle Auswertung umgangen. Das Programm öffnet die jeweiligen DPT-Dateien selbstständig und ermittelt die für die Orientierungsgradbestimmung notwendigen Amplituden. Weitere Theorie zum Thema der Orientierungsgradbestimmung wurde bereits in Kapitel 1 diskutiert. Abbildung 50 und Abbildung 51 zeigen Diagramme aus welchen sich die mathematischen Beziehungen für die Berechnung der Amplituden herleiten.

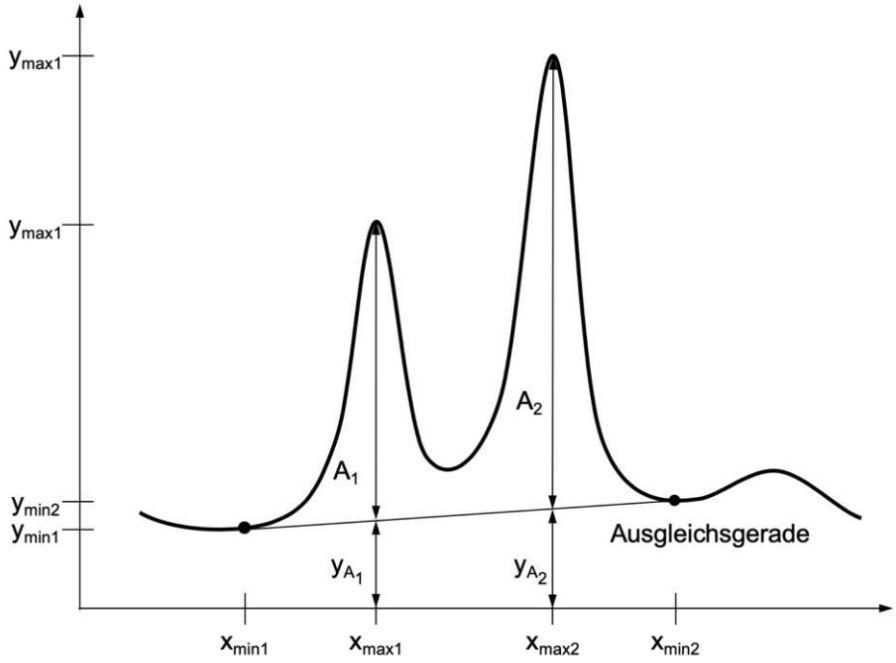


Abbildung 50: Bestimmung der Amplituden

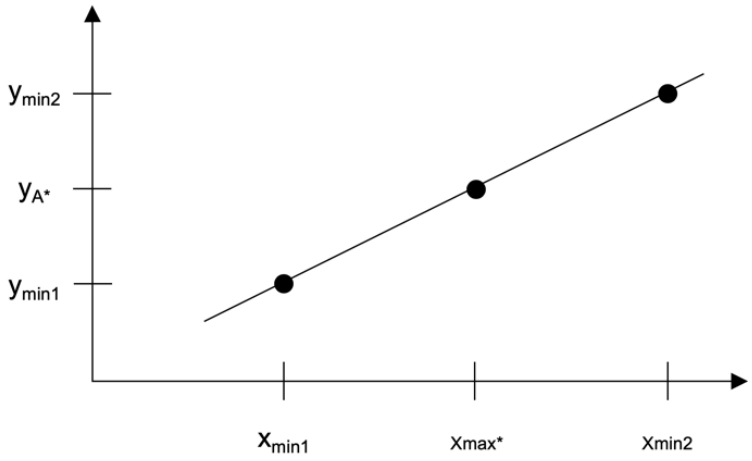


Abbildung 51: Skizze zur Herleitung der Amplituden

Durch die in Abbildung 51 dargestellte Skizze wird die in Abbildung 50 gezeigte Ausgleichsgerade ermittelt.

Formel 16 zeigt die sich darauf ergebende, formelle Beziehung.

$$\frac{y_{min2} - y_{min1}}{x_{min2} - x_{min1}} = \frac{y_{A_*} - y_{min1}}{x_{max*} - x_{min1}}$$

Formel 16: Ermittlung der Ausgleichsgerade

Mit dem Index * zur Bestimmung der beiden Amplituden 1 bzw. 2, welche in Abbildung 50 dargestellt sind. Umgestellt nach y_{A*} erhält man Formel 17.

$$y_{A_*} = \frac{y_{min2} - y_{min1}}{x_{min2} - x_{min1}}(x_{max*} - x_{min1}) + y_{min1}$$

Formel 17: Umstellen nach y_{A*}

Aus Abbildung 50 ergibt sich der Zusammenhang, welcher in Formel 18 gezeigt wird.

$$A_* = y_{max*} - y_{A_*}$$

Formel 18: Bestimmung der Amplitude

Der DPT-Auswerter ermittelt alle für die Berechnung der Amplituden nötigen Werte und gibt diese automatisiert aus. Eine entsprechende manuelle Auswertung wäre zeitlich sehr aufwändig.
Zur Auswertung der Daten mit Hilfe des Tools wird die Tooldatei „dpt_auswerter.py“ in den Ordner mit den DPT-Dateien kopiert. Wichtig hierbei ist es, vorab dafür zu sorgen, dass die DPT-Datei chronologisch durchnummeriert und pro Probe in einem Ordner gespeichert werden. Alle Messdaten eines Versuchs sind also zusammen in einem Ordner abzulegen. Nachdem diese Tooldatei im Dateiordner abgelegt wurde, wird diese mit der Python-IDLE geöffnet und ausgeführt. In weiterer Folge erscheint eine Excel-Datei mit den Ergebnissen im selbigen Ordner. Analog wird mit weiteren Messdaten umgegangen.

3. Resümee

In der vorliegenden Arbeit wurde erfolgreich eine automatisierte Polarisationsfilter-Schwenksteuerung für kontinuierliche Spektroskopiemessungen während des Zugversuchs an Kunststoffen entwickelt. Die Steuerung funktioniert autark, es ist also kein weiterer Computer nötig. Als besondere Herausforderung stellte sich die Synchronisierung der Schwenkvorgänge mit den FTIR-Messzeitpunkten dar, da die Messzeiten des Spektroskops um einige Millisekunden variieren. Der Grund für die unterschiedlichen Intervalle konnte nicht gänzlich geklärt werden. So ist es möglich, dass grundsätzlich hardwareseitig eine bestimmte Varianz auftritt. Des Weiteren könnte auch ein Softwareupdate der Software OPUS zu besser vorhersehbaren Messzeiten führen. Um die Schwenkvorrichtung zwischen zwei Messungen auszurichten, wurde eine Möglichkeit gefunden, die Schwenksteuerung mit dem Spektroskop kommunizieren zu lassen. Dabei teilt das Spektroskop mittels eines Triggers den Messbeginn bzw. des Messende mit. Dieses Signal wird von der Steuerung verarbeitet, woraufhin mit einer bestimmten gewünschten Verzögerung ein Trigger in Richtung FTIR-Spektroskop gesendet wird. Das Spektroskop erkennt diesen Auslöser und startet die nächste Messung.

Grundsätzlich traten mit dieser synchronisierten Steuerung kaum Probleme auf. Einzig und alleine die in der Arbeit beschriebene, zeitliche Differenz zwischen Triggerende des Spektroskops und der tatsächlichen Bereitschaft für eine erneute Messung des Geräts kann für Probleme sorgen, wenn diese über die eingestellte Verzögerung fällt. Geschieht dies, wird der Trigger der Steuerung wegen der fehlenden Bereitschaft nicht erkannt. Die Verzögerung wurde so eingestellt, dass dies nicht mehr passierte. Sollte aus irgendeinem Grund der Trigger in Zukunft nicht erkannt werden, sollte zuerst die Verzögerung hochgesetzt werden. Generell ist diese Verzögerung von etwa einer Sekunde ärgerlich, da diese wertvolle Messzeit raubt. Es kann mit ziemlicher Sicherheit davon ausgegangen werden, dass diese Verzögerung auf Grund von Speichervorgängen oder Ähnlichem stattfindet. Wie bereits erwähnt, könnte ein Softwareupdate, ein aktuelleres Betriebssystem oder neue Computerhardware zu besseren Ergebnissen führen.

Neben der Schwenksteuerung wurde außerdem eine automatisierte Kamerasteuerung realisiert, welche dann ausgelöst wird, wenn gerade geschwenkt wird, also zwischen

zwei FTIR-Messungen. Es wurde beobachtet, dass die verwendete Kamera sporadisch nicht auslöst und dass obwohl die Steuerung einen Trigger sendet. Geschieht dies, so müssen die fehlenden Daten innerhalb der Auswertung mittels Interpolation ermittelt werden. Generell ist die Auswertung der aufgenommenen Bilder sehr aufwendig, da diese manuell vermessen werden müssen. Zukünftige Arbeiten könnten sich damit beschäftigen, diese Auswertung mittels Software automatisiert auszuwerten.

Wegen Problemen mit der Vorgängerversion wurde in dieser Arbeit außerdem ein Tool zur Auswertung der Amplituden der aufgenommenen Spektren programmiert. Im Gegensatz zum vorherigen Berechnungstool bietet dieses einige Vorteile. Auf Grund des textbasierten Programmcodes ist dieses für spätere Anwender wesentlich besser adaptier- und lesbar. Aufgetretene Probleme wurden erledigt und zudem ist die Software frei zugängig.

Die abschließenden Auswertungen der finalen Version der Steuerung zum Orientierungsgrad entsprechen weitgehend den Daten aus der Literatur. Mit diesen Ergebnissen kann davon ausgegangen werden, dass die kontinuierliche Fahrt bei den Spektroskopiemessungen mit automatisierter Schwenksteuerung die vorherige, gestufte Fahrt ohne Bedenken ersetzen kann. Innerhalb künftiger Arbeiten könnte man sich damit beschäftigen, die Messzeiten zu reduzieren, um damit eine höhere Mess-auflösung und eine höhere Genauigkeit der Spektroskopie-Daten zu erhalten. Es bleibt das Problem bestehen, dass zwei zusammengehörige Messdaten durch die Verschiebung während des Zugversuchs nicht am exakt gleichen Ort aufgenommen werden. Der Fehler kann durch eine Reduzierung der dazwischenliegenden Zeit minimiert werden. Hier seien allerdings zum Vergleich nochmals die Missstände der Ergebnisse der gestuften Fahrt angemerkt, welche durch etwaige Kriechvorgänge innerhalb des Materials verfälscht werden. Schlussendlich müssen bei beiden Methoden Kompromisse eingegangen werden, wobei bei der neuen Methode der Fehler durch die Verringerung der Messzeit prinzipiell steuerbar ist.

4. Literaturverzeichnis

1. https://de.wikipedia.org/wiki/Duroplaste#Anwendungsgebiete. [Online]
2. https://www.modulor.de/werkstoffbibliothek/kunststoffe/. [Online]
3. Lechler, Katharina und Menner, Marietta. [Online] 2016. https://docplayer.org/68227467-Faserverbundwerkstoffe.html.
4. http://www.scinexx.de/wissen-aktuell-22397-2018-02-08.html. [Online]
5. https://computerwelt.at/news/3d-druck-gadgets-revolutionieren-chirurgie/. [Online]
6. https://www.nicolaginzler.com/2013/07/01/cortex-3d-printed-cast-for-fractures/. [Online]
7. http://noe.orf.at/news/stories/2848082/. [Online]
8. http://www.faz.net/aktuell/wirtschaft/wirtschaftspolitik/abschaffung-der-plastiktuete-bei-rewe-laut-oekologen-nutzlos-14263982.html. [Online]
9. http://www.baustoffwissen.de/wissen-baustoffe/baustoffknowhow/grundstoffe/kunststoffe/umstrittene-zusaetze-weichmacher/. [Online]
10. https://www.global2000.at/pvc-gesundheitsschaedliche-weichmacher. [Online]
13. Michler, G. H. Part B: Polymer physics. *Journal of polymer science.* 2004, Bd. 42, 24, S. 4478-4488.
14. Houska, M. *Polymer Engineering and Science.* 1987, 27. S. 917-924.
15. https://de.nanotec.com/fileadmin/files/Software/Treiber/SMCI_3x-1_USB.zip. [Online]
16. http://www.nordpraezision.de/proben-stanzmesser.html. [Online]
17. https://www.precifast.de/servomotor-funktion-ansteuern-anschliessen/. [Online]
18. https://store.arduino.cc/usa/arduino-uno-rev3. [Online]
19. http://www.spiegel.de/netzwelt/gadgets/arduino-erklaert-das-kann-der-microcontroller-a-1105328.html. [Online]
20. https://hilftdirweiter.de/treiber-fuer-arduino-nano-clones/. [Online]
21. https://de.wikipedia.org/wiki/Arduino_(Plattform). [Online]
22. https://www.mikrocontroller.net/articles/Entprellung. [Online]
23. https://www.mikrocontroller.net/articles/Platinenherstellung_mit_der_Photo-Positiv-Methode. [Online]

24. Datenblatt MCT6 Optokoppler. [Online]

25. https://www.arduino.cc/en/Reference/Wire. [Online]

26. https://www.arduino.cc/reference/en/language/functions/time/millis/. [Online]

5. Anhang

5.1. Programmcode der Version 1

```
// Displayanschluesse
// SDA auf A4
// SDL auf A5

#include <Wire.h>

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // LCD I2C address

#include <Servo.h>  // servo library
Servo servo1;  // servo control object

const int motorled = 7;
const int tasterpin = 4; // mit Spannungsteiler
const int motorpin = 3;
const int potpin = 1; // Analog A1
const int nullled = 8;
const int neunzigled = 9;

// flags
int flag1 = 1; // einmalige Einstellung der Zeit zu Beginn
int flag2 = 0; // wenn flag2 = 1, dann starte Motor
int flag3 = 0; // erster startknopf ohne pauseanfang, wichtig für Pause-Zeitmessung
int flag4 = 0; // Pause hochzählen
int flag5 = 1; // Erster Servodurchlauf ohne FTIRpause, --> Servostellung zu Beginn
der Messpause von FTIR

int interval; // millisekunden startwert, einstellbar über Poti zu Beginn
int shifttime;
int tasterwert; // Tasterstatusabfrage
int oldtasterwert;
int anfangspos = 30; // Anfangswinkel um mechanisches Anfahren des Servos zu
verhindern
int winkel = 0; // Winkel switcht jeden Durchgang zwischen 0 und 90
int count = 0; // zählt Durchgaenge
int nullledState = HIGH; // Winkelanzeige LED 0 Grad
int neunzigledState = LOW; // Winkelanzeige LED 90 Grad

unsigned long starttime; // Speichert Zeitpunkt beim erstmaligen Drücken des
Startknopfes
unsigned long time; // Laufzeit
unsigned long FTIRpause = 1000; // ms, 1 Sekunde Pause des FTIR-Spektrometers
zwischen Messungen
```

```
unsigned long verzoegerung; // verhindert prellen der Taste
unsigned long pause = 0; // Summe der Pausen in sec
unsigned long pausestart = 0; // wie starttime, Zeitpunkt des Starts der Pause wird
gespeichert
unsigned long oldpause = 0; // um mehrer Pausen zu summieren

double timedisplay;
double pausedisplay;

// folgender Block: um delay() zu umgehen, delay stoppt das Programm,
nachfolgendes lässt parallele Vorgänge zu, wichtig für Zeitmessung
unsigned long currentMillis; // für Motorinterval
unsigned long previousMillis = 0; // für Motorinterval
unsigned long previousMillis2 = 0; // für Laufzeit
unsigned long currentMillis2; // für Pause
unsigned long previousMillis3 = 0; // für Pause

// ------------------------------ Setup Display und Pins -------------------------
void setup()
{
  pinMode(motorled, OUTPUT); // LED ist Output
  pinMode(tasterpin, INPUT); // Taster is Input

  //Serial.begin(9600);
  servo1.attach(motorpin);
  servo1.write(anfangspos); // Anfahren der Anfangsposition 0 Grad

  digitalWrite(nullled, HIGH);  // 0 Grad LED leuchtet
  digitalWrite(neunzigled, LOW); // 90 Grad LED leuchtet nicht

  lcd.begin(20, 4);  // LCD hat 4 Zeilen à 20 Zeichen
  lcd.setCursor(0, 0); //Start in Zeile 0, Zeichen 0
  lcd.print("SetSteptime: ");
  lcd.setCursor(0, 1);
  lcd.print("Counter: 0");
  lcd.setCursor(0, 2);
  lcd.print("Runtime: 0 s");
  lcd.setCursor(0, 3);
  lcd.print("Pause: 0 s");
}
////////////////////////////////////////////////////////////////////////////////

void loop()   {
  tasterwert = digitalRead(tasterpin); // Tasterstatusabfrage, 0 heißt Taste gedrückt

// --------------------------- Intervaleinstellung, Start & Stopp ----------------------------

  if ((tasterwert == 1) && (flag1 == 1)) { // Einstellung der Intervalzeit über Poti
    int potval = analogRead(potpin);
```

```
    interval = map(potval, 0, 1023, 2000, 6000); // Lineare Interpolation
    interval = interval / 10;
    interval = interval * 10; // damit 10 ms Sprünge, Display "springt" weniger
    shifttime = interval - FTIRpause; // erster shift ohne ftirpause
    lcd.setCursor(13, 0);
    lcd.print(interval);
    lcd.print(" ms");
  }

  if ((tasterwert == 0) && (oldtasterwert == 1) && (flag1 == 1)) { // wenn Start/Stop-
Taste gedrückt wird, lege Interval fest
    lcd.setCursor(0, 0);
    lcd.print("Steptime: ");
    lcd.print(shifttime);
    lcd.print(" ms   ");
    starttime = millis();
    flag1 = 0; // unterbinden von voriger if, pot einstellung unterbrochen
  }

  if (count == 2) { // ab jetzt normales (mit poti eingestelltes) interval, erster durchlauf
ohne FTIRpause, weil Spektroskop direkt startet (ohne Pause zu Beginn), dies führt
dann zur Servostellung am Pausenbeginn
     shifttime = interval;
     lcd.setCursor(0, 0);
     lcd.print("Steptime: ");
     lcd.print(shifttime);
     lcd.print(" ms   ");
  }

  if ((tasterwert == 0) && (oldtasterwert == 1) && (flag2 == 0) && (flag3 == 0)) {    //
Knopf als Startknopf, zuerst ohne pause, nur 1x
    flag2 = 1; // für Start des Motors
    digitalWrite(motorled, HIGH);
    verzoegerung = millis();
    flag3 = 1;
  }

currentMillis = millis();
  if ((currentMillis - verzoegerung  >= 500) && (tasterwert == 0) && (oldtasterwert ==
1) && (flag2 == 0) && (flag3 == 1)) {    // Knopf als Startknopf; für Pausenmessung
bereit
    flag2 = 1; // für Start des Motors, siehe unten
    flag4 = 0; // deaktiviert Pausenzähler und Darstellung
    digitalWrite(motorled, HIGH);
    verzoegerung = millis();
    oldpause = pause; // Pausen summieren
    //pausedisplay(); // Anzeige am LCD
  }
```

```
  if ((currentMillis - verzoegerung  >= 500) && (tasterwert == 0) && (oldtasterwert ==
1) && (flag2 == 1) && (flag3 == 1)) { // Als Stopknopf, taste erst nach 500 ms
möglich, prellen verhindern
    flag2 = 0;
    flag4 = 1; // aktiviert Pausenzähler und Darstellung
    pausestart = millis(); // Pausestart wenn Stop gedrückt wurde
    digitalWrite(motorled, LOW);  // MotorLED aus
    verzoegerung = millis();
  }

////////////////////////////////////////////////////////////////////////////////

// ------------------------------ Servosteuerung und Zeitdarstellung ---------------

  if (flag2 == 1) {
    currentMillis = millis();
    if (currentMillis - previousMillis >= shifttime) {
      previousMillis = currentMillis;
      servo1.write(anfangspos + winkel); // Servostellung zu winkel
      counter();
      digitalWrite(nullled, nullledState);
      digitalWrite(neunzigled, neunzigledState);

      if (winkel == 0) // switch winkel
        winkel = 98; // entspricht 90° auf zahnrad
      else
        winkel = 0;

      if (nullledState == LOW) // switch LEDstate
        nullledState = HIGH;
      else
        nullledState = LOW;

      if (neunzigledState == LOW) // switch LEDstate
        neunzigledState = HIGH;
      else
        neunzigledState = LOW;
    }
  }

  currentMillis2 = millis();
  if ((currentMillis2 - previousMillis2 >= 100) && (flag2 == 1)) { // Laufzeit alle 100ms
aktualisieren
    previousMillis2 = currentMillis2;
    time = millis() - starttime - pause;
    timedisplay = (double)time / 1000;

    lcd.setCursor(0, 2);
    lcd.print("Runtime: ");
```

```
    lcd.print(timedisplay, 1); // 3 nachkommastellen
    lcd.print(" s  ");
     }
  if ((currentMillis2 - previousMillis3 >= 100) && (flag4 == 1)) { // Laufzeit alle 100 ms
akt.
    previousMillis3 = currentMillis2;
    pause = oldpause + millis() - pausestart;
    pausedisplay = (double)pause / 1000;

    lcd.setCursor(0, 3);
    lcd.print("Pause: ");
    lcd.print(pausedisplay, 1);
    lcd.print(" s  ");
    }

  oldtasterwert = tasterwert; // verhindert prellen
}

////////////////////////////////////////////////////////////////////////////////

// ---------------------------- Funktionen -------------------------------------


  void counter() { // Schrittzähler
  count++;
  lcd.setCursor(9, 1);
  lcd.print(count);
}

////////////////////////////////////////////////////////////////////////////////
```

5.2. Kurzanleitung Version 1

Vorbereitung

1. Kabel des Servomotors (Klinkenkabel) an der Steuereinheit anschließen.
2. Stromversorgung herstellen, dafür beide USB-Kabel der Steuereinheit anschließen (wahlweise mit beigelegtem Netzteil oder durch Anschlüsse am PC).
3. Reset-Knopf (Weißer Taster) drücken (die Steuerung ist nun einsatzbereit, der Servomotor steht auf der 0°-Position).
4. Polarisationsfilter in FTIR einschieben und Zahnrad des Filters auf 0° stellen.
5. Motorhalterung an Polarisationsfilter montieren (mit Flügelmuttern festziehen), Zahnräder müssen in einander greifen.
6. Überprüfen ob Zahnrad des Filters noch immer möglichst genau auf 0° steht, falls nicht zurück zu Punkt 4.

Betrieb

1. Reset-Taste drücken, dies setzt alle Einträge zurück und stellt den Motor auf 0°.
2. Das gewünschte Intervall (in ms) mit dem Drehknopf einstellen.
   Hinweis: Am Spektroskop muss zwischen den Messungen eine 1-sekündige Pause eingestellt werden, da in der Pause der Motor gestellt wird. Die Pause ist Bestandteil des obig erwähnten Intervalls, also: Intervall = Messzeit + Pause (1 Sekunde). Die Messzeit (welche von der Anzahl der Messungen pro Schwenkperiode abhängt) muss vorab per Stoppuhr mehrmals gemessen und gemittelt werden.
3. Bei Messbeginn Start/Stopp-Knopf drücken (schwarzer Taster).
   Hinweis: Die Messung startet direkt, da das Spektroskop nicht mit der (bereits erwähnten) Pause beginnt, wird beim ersten Durchgang die Pause automatisch vom Intervall abgezogen.
4. Bei Versuchsende Start/Stopp-Taste drücken. Am Display wird das Intervall, die Anzahl der Schwenkperioden, die Versuchszeit und (falls pausiert wurde) die

Dauer der Pause angezeigt. Die Einträge können nicht gespeichert werden und verfallen mit dem Reset.

Hinweis: Nach einem Stopp kann erneut gestartet werden, der Motor fährt dann sofort in die nächste Position (0° oder 90°).

Achtung! Die Reset-Taste löscht alle Einträge am Display und startet die Steuerung neu. Der Motor wird dabei wieder auf 0° gestellt. Vor jeder neuen Messung muss die Reset-Taste gedrückt werden.

Programmierung (nur bei Bedarf)

Voraussetzung: Arduino IDE mit Servo-Library und installiertem Arduino-Treiber (Ordner mit Software, Code und Treiber vorhanden). Der USB-Stecker mit der roten Markierung ist am Arduino angeschlossen, kann also nur über dieses Kabel erneut geflasht werden. Das zweite Kabel ist alleinig für die Stromversorgung des Displays verantwortlich und verhindert damit ein flackerndes Display während der Motorstellung.

5.3. Programmcode der Version 2

```
// Displayanschluesse
// SDA auf A4
// SDL auf A5
// ACK auf A6
// TRIG auf D2

#include <Wire.h>

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // LCD I2C address

#include <Servo.h>  // servo library
Servo servo1;  // servo control object

const int TRIGout = 2;
const int tasterpin = 4; // mit Spannungsteiler
const int motorpin = 3;
// const int potpin = 1; // Analog A1
const int TRIGled = 7;
const int nullled = 8;
const int neunzigled = 9;
const int kamera = 12;
```

```
const int kamerafokus = 13;

int ACKin;
int ACK;
int oldACK;
int tasterwert; // Tasterstatusabfrage
int oldtasterwert;
int anfangspos = 30; // Anfangswinkel um mechanisches Anfahren des Servos zu
verhindern
int winkel = 91; // Winkel switcht jeden Durchgang zwischen 0 und 91
int count = 0; // zählt Durchgaenge
int nullledState = HIGH; // Winkelanzeige LED 0 Grad
int neunzigledState = LOW; // Winkelanzeige LED 90 Grad

int startbutton = 0;
//int flag4 = 0;

unsigned long starttime; // Speichert Zeitpunkt beim erstmaligen Drücken des
Startknopfes
unsigned long time; // Laufzeit
unsigned long oldtime = 0;
double timedisplay;

//unsigned long pause = 0; // Summe der Pausen in sec
//unsigned long pausestart = 0; // wie starttime, Zeitpunkt des Starts der Pause wird
gespeichert
//unsigned long oldpause = 0; // um mehrer Pausen zu summieren
//double pausedisplay;

unsigned long verzoegerung; // verhindert prellen der Taste
unsigned long currentMillis;
unsigned long currentMillisS;
unsigned long currentMillisT;
unsigned long previousMillisP = 0; // für Pause
unsigned long previousMillisT = 0;
unsigned long previousMillisS = 0;

void setup() {
  pinMode(TRIGout, OUTPUT);
  pinMode(TRIGled, OUTPUT);
  pinMode(tasterpin, INPUT);

  servo1.attach(motorpin);
  servo1.write(anfangspos); // Anfahren der Anfangsposition 0 Grad

  digitalWrite(nullled, HIGH);  // 0 Grad LED leuchtet
  digitalWrite(neunzigled, LOW); // 90 Grad LED leuchtet nicht
  digitalWrite(TRIGout, HIGH);
  digitalWrite(TRIGled, LOW);
```

```
  digitalWrite(kamerafokus, HIGH);

  lcd.begin(20, 4);  // LCD hat 4 Zeilen à 20 Zeichen
  lcd.setCursor(0, 0); //Start in Zeile 0, Zeichen 0
  lcd.print("ACK: X");
  lcd.setCursor(0, 1);
  lcd.print("Counter: 0");
  lcd.setCursor(0, 2);
  lcd.print("Runtime: 0 s");
//  lcd.setCursor(0, 3); PAUSE
//  lcd.print("Pause: ");
}

void loop() {
  tasterwert = digitalRead(tasterpin); // Tasterstatusabfrage, 0 heißt Taste gedrückt

  int ACKin = analogRead(6);
  if (ACKin >= 1000) {
    ACK = 1; // Messung aktiv
    }
  else {
    ACK = 0; // keine Messung
    }

  lcd.setCursor(0, 0); //Start in Zeile 0, Zeichen 0
  lcd.print("ACK: ");
  lcd.print(ACK);
  lcd.print("   ");

////////////////////////////////////////////////////////////////////////////////////////////// Startknopf und Stop
  currentMillis = millis();
  if ((currentMillis - verzoegerung  >= 500) && (tasterwert == 0) && (oldtasterwert ==
1) && (startbutton == 0)) { // wenn Start/Stop-Taste gedrückt wird, lege Interval fest
    verzoegerung = millis();
    startbutton = 1;
    starttime = millis();
    TRIGCAMsend();
    counter();
//    flag4 = 0; // deaktiviert Pausenzähler und Darstellung    PAUSE
//    oldpause = pause; // Pausen summieren
  }

   if ((currentMillis - verzoegerung  >= 500) && (tasterwert == 0) && (oldtasterwert ==
1) && (startbutton == 1)) { // Als Stopknopf, taste erst nach 500 ms möglich, prellen
verhindern
    verzoegerung = millis();
    startbutton = 2; // 0 um startknopf zu reaktivieren
    oldtime = time;
```

```
//    flag4 = 1; // aktiviert Pausenzähler und Darstellung              PAUSE
//    pausestart = millis(); // Pausestart wenn Stop gedrückt wurde
  } /////////////////////////////////////////////////////////////////////////////////////////////////////////////////


//vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

  // Steuerung
  if ((ACK == 0) && (oldACK == 1) && (startbutton == 1)) { // keine Messung, Start
gedrückt
      delay(1300); // verzögerung, da datei auf spektroskop noch nicht erscheint
      servo1.write(anfangspos + winkel); // Servostellung 0°
      counter();
      switcher();
      digitalWrite(nullled, nullledState);
      digitalWrite(neunzigled, neunzigledState);

      delay(1200); // Zeit um Motor zu stellen // ack=0 ABER Messung SYNC Fenster
noch nicht bereit; 2000ms funktioniert nicht

      TRIGCAMsend();
  }

//^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  // Zeit
  currentMillisT = millis();
  if ((currentMillisT - previousMillisT >= 100) && (startbutton == 1)) { // Laufzeit alle
100ms aktualisieren
    previousMillisT = currentMillisT;
    time = oldtime + millis() - starttime;
    timedisplay = (double)time / 1000;

    lcd.setCursor(0, 2);
    lcd.print("Runtime: ");
    lcd.print(timedisplay, 1); // 3 nachkommastellen
    lcd.print(" s  ");
     }

//  if ((currentMillisT - previousMillisP >= 100) && (flag4 == 1)) { // Laufzeit alle 100 ms
akt.      PAUSE
//    previousMillisP = currentMillisT;
//    pause = oldpause + millis() - pausestart;
//    pausedisplay = (double)pause / 1000;
//
//    lcd.setCursor(0, 3);
//    lcd.print("Pause: ");
//    lcd.print(pausedisplay, 1);
```

```
//    lcd.print(" s  ");
//    }

 oldtasterwert = tasterwert; // verhindert prellen
 oldACK = ACK;
   }
   // ---------------------------- Funktionen -----------------------------------


  void counter() { // Schrittzähler
    count++;
    lcd.setCursor(9, 1);
    lcd.print(count);
  }

  void switcher() {
      if (winkel == 0) // switch winkel
        winkel = 91; // entspricht 90° auf zahnrad
      else
        winkel = 0;

      if (nullledState == LOW) // switch LEDstate
        nullledState = HIGH;
      else
        nullledState = LOW;

      if (neunzigledState == LOW) // switch LEDstate
        neunzigledState = HIGH;
      else
        neunzigledState = LOW;
    }

  void TRIGCAMsend() { // Trigger und Kamera
    digitalWrite(kamerafokus, LOW);
    delay(100);
    digitalWrite(kamera, HIGH);
    delay(10);
    digitalWrite(kamera, LOW);
    delay(10);
    digitalWrite(kamerafokus, HIGH);

    digitalWrite(TRIGout, LOW); // Trigger gesendet
    digitalWrite(TRIGled, HIGH);
    delay(10); // Triggersendedauer
    digitalWrite(TRIGout, HIGH);
    digitalWrite(TRIGled, LOW);
    }
////////////////////////////////////////////////////////////////////////////////
```

5.4. Kurzanleitung Version 2

Vorbereitung

1. Kabel des Servomotors (Klinkenkabel) an der Steuereinheit anschließen.
2. Den FTIR-Adapter zwischen Buchse des TENSOR 27 und des DSUB-25 Steckers einsetzen.
3. Stromversorgung herstellen, dafür beide USB-Kabel der Steuereinheit anschließen (wahlweise mit beigelegtem Netzteil oder durch Anschlüsse am PC).
4. Reset-Knopf (Weißer Taster) drücken (die Steuerung ist nun einsatzbereit, der Servomotor steht auf der 0°-Position).
5. Polarisationsfilter in FTIR einschieben und Zahnrad des Filters auf 0° stellen.
6. Motorhalterung an Polarisationsfilter montieren (mit Flügelmuttern festziehen), Zahnräder müssen in einander greifen.
7. Überprüfen ob Zahnrad des Filters noch immer möglichst genau auf 0° steht, falls nicht zurück zu Punkt 4.

Betrieb

1. Reset-Taste drücken, dies setzt alle Einträge zurück und stellt den Motor auf 0°.
2. Im Programm OPUS 6.0 die entsprechenden Einstellungen für den Mehrfach-Trigger vornehmen. Dazu im Menü „Measure / Enhanced Measurement“ den Ordner „Optics parameters“ auswählen und „ON“ für „External synchronization“ auswählen.

Hinweis: Die entsprechenden Parameter können auch mit der XPM-Datei im folgenden Verzeichnis geladen werden:
C:\Programme\OPUS\XPM\Mikroskop_Trigger_Weiss.XPM

3. Die Messung innerhalb von OPUS 6.0 starten. Danach erscheint ein Fenster, welches anzeigt, dass die Software auf einen Trigger wartet.
4. Mit Hilfe des Start/Stopp-Knopfes der Filtersteuerung wird der erste Trigger gesendet und damit die Messung gestartet.

Hinweis: Während des Versuchs erscheint das Fenster, welches anzeigt, dass die Software auf einen Trigger wartet, nach jeder Messung erneut. Durch das ACK-Signal des Spektroskops löst die Steuerung den Trigger für die nächste Messung automatisch aus.

5. Während des gesamten Versuchs wird am Display das Intervall, die Anzahl der Schwenkperioden und die Versuchszeit angezeigt. Die Einträge können nicht gespeichert werden und verfallen mit dem Reset.

Hinweis: Achtung! Die Reset-Taste löscht alle Einträge am Display und startet die Steuerung neu. Der Motor wird dabei wieder auf 0° gestellt. Vor jeder neuen Messung muss die Reset-Taste gedrückt werden.

Programmierung (nur bei Bedarf)

Voraussetzung: Arduino IDE mit Servo-Library und installiertem Arduino-Treiber (Ordner mit Software, Code und Treiber vorhanden).
Der USB-Stecker mit der roten Markierung ist am Arduino angeschlossen, kann also nur über dieses Kabel erneut geflasht werden. Das zweite Kabel ist alleinig für die Stromversorgung des Displays verantwortlich und verhindert damit ein flackerndes Display während der Motorstellung.

5.5. FTIR-Einstellungen Vergleich

Verlorengegangene Einstellungen	Neue Einstellungen
{EXP='mikroskop_weiss_trig.XPM'	{EXP='Mikroskop_Mehrfach_Trigger_Weiss.XPM'
CNM='gast'	CNM='Gast'
SNM='Probe16'	SNM='Probe22'
APT='4 mm'	APT='6 mm'
MXE=600.000000	MXE=100.000000
MYE=2.500000	MYE=1.100000
NSS=4	NSS=3
NSR=4	NSR=3
REP=35	REP=50
RES=2.000000	RES=4.000000
SFM='Mikroskop'	SFM=''
UNI='ITC=1'	UNI='?'
VEL='20.0'	VEL='10.0'

5.6. DPT-Auswerter - Python Tool

```
import xlsxwriter
import os

list = []
files = []

def error(msg):
    print()
    for i in range(4):
        print("#"*70)
    print()
    print(msg)
    print()
    for i in range(4):
        print("#"*70)
    print()

def load_file_to_list(filename):
    global list
    try:
        file = open(filename, "r")
        for line in file:
            values = line.split()
            list.append(values)
        file.close()
        return True
    except:
        error("Datei '"+filename+"' nicht gefunden. Dateiname überprüfen.")
        return False

def getSection(start, end):
    global list
    section = []
    for item in list:
        x = float(item[0])
        y = float(item[1])
        if start >= x >= end:
            section.append([x,y])
    return section

def getEnd(filename):
    end = filename.replace(".DPT","")
    end = end[end.index("_")+1:]
    return end
```

```
def main():
    global files
    global list
    path = os.getcwd()
    print("Du bist hier: "+path)
    for file in os.listdir(path):
        if file.endswith(".DPT"):
            print("Datei gefunden: "+file)
            files.append(file)
    print(str(len(files))+" Dateien gefunden:")
    #print(files)

    filename = files[0]
    probenumber = filename.replace("Probe", "")
    probenumber = probenumber.replace(".DPT", "")
    numbers = probenumber.split("_")
    probenumber = numbers[0]

    resultfilename = "dpt_auswerter_Amplituden_"+str(probenumber)+".xlsx"

    f = {}
    for file in files:
        f[file] = int(getEnd(file))

    sortedFiles = sorted(f.keys(), key = lambda x: f[x])

    workbook = xlsxwriter.Workbook(resultfilename) # Create an new Excel file and
add a worksheet.
    worksheet = workbook.add_worksheet()
    worksheet.set_column('A:A', 12) # Widen the first column to make the text clearer.

    worksheet.write(0, 0, 'Datei')
    worksheet.write(0, 1, 'A998')
    worksheet.write(0, 2, 'A974')
    for i in range(len(sortedFiles)):
        file_flag = load_file_to_list(sortedFiles[i])
        if file_flag:

            s1 = 1039.0 #1040.0
            e1 = 990.0
            s2 = e1
            e2 = 951.0 #950.0

            section1 = getSection(s1,e1)
            section2 = getSection(s2,e2)

            max1 = max(section1, key=lambda item: item[1])
            min1 = min(section1, key=lambda item: item[1])
```

```
        max2 = max(section2, key=lambda item: item[1])
        min2 = min(section2, key=lambda item: item[1])

        xmax1 = max1[0]
        ymax1 = max1[1]

        xmin1 = min1[0]
        ymin1 = min1[1]

        xmax2 = max2[0]
        ymax2 = max2[1]

        xmin2 = min2[0]
        ymin2 = min2[1]

        yA1 = ((ymin2-ymin1)/(xmin2-xmin1))*(xmax1-xmin1)+ymin1
        yA2 = ((ymin2-ymin1)/(xmin2-xmin1))*(xmax2-xmin1)+ymin1

        A1 = ymax1 - yA1
        A2 = ymax2 - yA2

        list = []

####################### Excel-Ausgabe ######################

        #print(str(A1)+" "+str(A2))
        line = i+1
        worksheet.write(line, 0, sortedFiles[i])
        worksheet.write(line, 1, A1)
        worksheet.write(line, 2, A2)


    print("Excel-Datei erstellt für Probe"+str(probenumber))
    workbook.close()

##########################################################################

if __name__ == "__main__":
    main()
```

5.7. Datenblatt Optokoppler

FAIRCHILD
SEMICONDUCTOR®

February 2010

MCT6, MCT61, MCT62
Dual Phototransistor Optocouplers

Features

- Two isolated channels per package
- Two packages fit into a 16 lead DIP socket
- Choice of three current transfer ratios
- Underwriters Laboratory (U.L.) recognized File E90700
- VDE approved for IEC60747-5-2

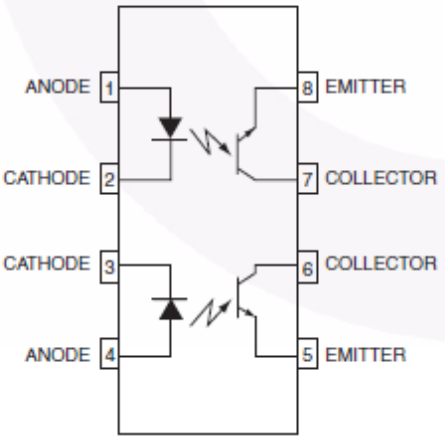
Description

The MCT6X Optocouplers have two channels for density applications. For four channel applications, two-packages fit into a standard 16-pin DIP socket. Each channel is an NPN silicon planar phototransistor optically coupled to a gallium arsenide infrared emitting diode.

Applications

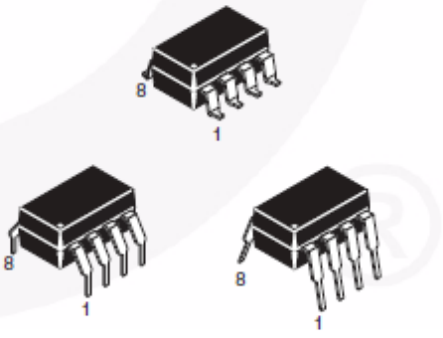
- AC line/digital logic – isolate high voltage transients
- Digital logic/digital logic – eliminate spurious grounds
- Digital logic/AC triac control – isolate high voltage transients
- Twisted pair line receiver – eliminate ground loop feedthrough
- Telephone/telegraph line receiver – isolate high voltage transients
- High frequency power supply feedback control – maintain floating grounds and transients
- Relay contact monitor – isolate floating grounds and transients
- Power supply monitor – isolate transients

Schematic



Equivalent Circuit

Package Outlines



MCT6, MCT61, MCT62 — Dual Phototransistor Optocouplers

MCT6, MCT61, MCT62 Rev. 1.0.5

www.fairchildsemi.com

Absolute Maximum Ratings

Stresses exceeding the absolute maximum ratings may damage the device. The device may not function or be operable above the recommended operating conditions and stressing the parts to these levels is not recommended. In addition, extended exposure to stresses above the recommended operating conditions may affect device reliability. The absolute maximum ratings are stress ratings only.

Symbol	Rating	Value	Unit
TOTAL DEVICE			
T_{STG}	Storage Temperature	-55 to +150	°C
T_{OPR}	Operating Temperature	-55 to +100	°C
T_{SOL}	Lead Solder Temperature (Refer to Reflow Temperature Profile)	260 for 10 sec	°C
P_D	Total Device Power Dissipation @ T_A = 25°C	400	mW
	Derate above 25°C	5.33	mW/°C
EMITTER (Each channel)			
I_F	Forward Current – Continuous	60	mA
I_F(pk)	Forward Current – Peak (PW = 1µs, 300pps)	3	A
V_R	Reverse Voltage	3.0	V
P_D	LED Power Dissipation @ T_A = 25°C	100	mW
	Derate above 25°C (Total Input)	1.3	mW/°C
DETECTOR (Each channel)			
I_C	Collector Current – Continuous	30	mA
P_D	Detector Power Dissipation @ T_A = 25°C	150	mW
	Derate above 25°C	2.0	mW/°C

Electrical Characteristics (T_A = 25°C unless otherwise specified)

Individual Component Characteristics

Symbol	Parameter	Test Conditions	Min.	Typ.*	Max.	Units
EMITTER						
V_F	Input Forward Voltage	I_F = 20mA		1.2	1.5	V
V_R	Reverse Voltage	I_R = 10µA	3.0	25		V
I_R	Reverse Current	V_R = 5V		0.001	10	µA
C_J	Junction Capacitance	V_F = 0V, f = 1MHz		50		pF
DETECTOR						
BV_{CEO}	Collector-Emitter Breakdown Voltage	I_C = 1.0mA, I_F = 0	30	85		V
BV_{ECO}	Emitter-Collector Breakdown Voltage	I_E = 100µA, I_F = 0	6	13		V
I_{CEO}	Collector-Emitter Dark Current	V_{CE} = 10V, I_F = 0		5	100	nA
C_{CE}	Capacitance	V_{CE} = 0V, f = 1MHz		8		pF

Transfer Characteristics

Symbol	Characteristic	Test Conditions	Min.	Typ.*	Max.	Units
SWITCHING CHARACTERISTICS (AC)						
t_{on}	Non-Saturated Turn-on Time	R_L = 100Ω, I_C = 2mA, V_{CC} = 10V		2.4		µs
t_{off}	Non-Saturated Turn-off Time			2.4		µs
CURRENT TRANSFER RATIO, COLLECTOR-EMITTER (DC)						
CTR	MCT6	I_F = 10mA, V_{CE} = 10V	20			%
	MCT61	I_F = 5mA, V_{CE} = 5V	50			
	MCT62		100			
$V_{CE(sat)}$	Saturation Voltage	I_F = 16mA, I_C = 2mA		0.15	0.40	V

Isolation Characteristics

Symbol	Characteristic	Test Conditions	Min.	Typ.*	Max.	Units
V_{ISO}	Input-Output Isolation Voltage	$I_{I-O} \leq$ 10µA, t = 1min.	5000			Vac(rms)
R_{ISO}	Isolation Resistance	V_{I-O} = 500VDC	10^{11}			Ω
C_{ISO}	Isolation Capacitance	f = 1MHz		0.5		pF

*All typicals at T_A = 25°C

Typical Performance Curves

Fig. 1 Normalized CTR vs. Forward Current

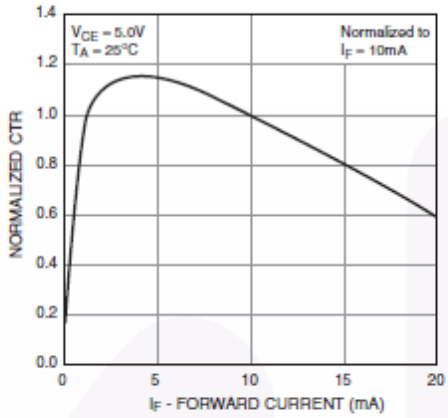


Fig. 2 Normalized CTR vs. Ambient Temperature

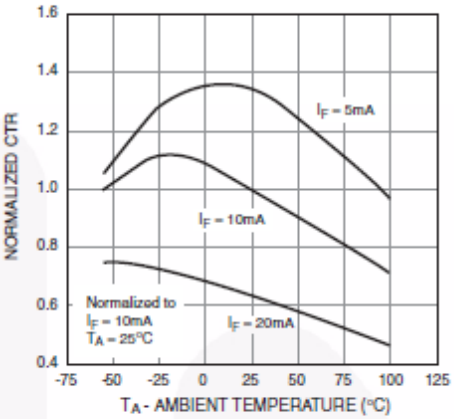


Fig. 3 Dark Current vs. Ambient Temperature

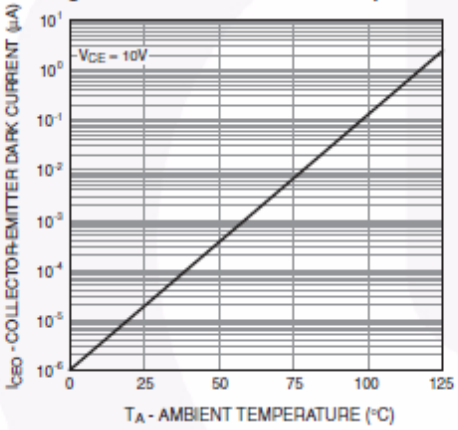


Fig. 4 Switching Speed vs. Load Resistor

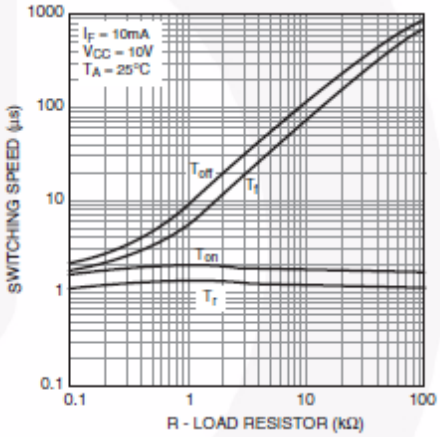


Fig. 5 LED Forward Voltage vs. Forward Current

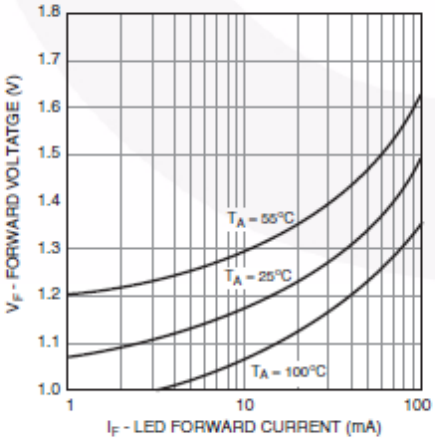
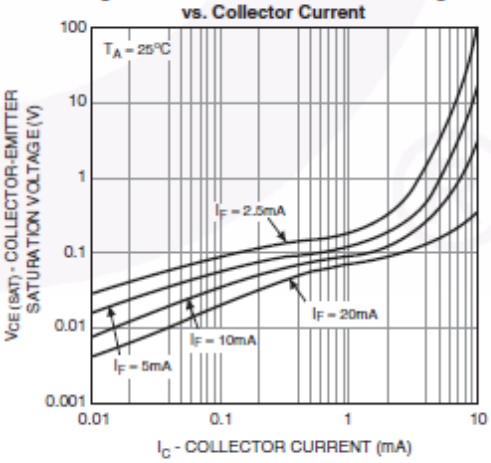
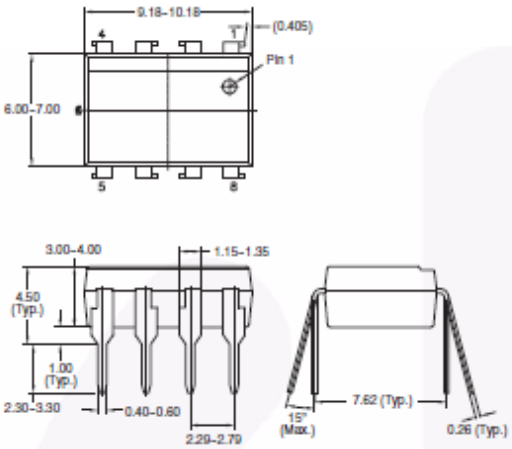


Fig. 6 Collector-Emitter Saturation Voltage vs. Collector Current

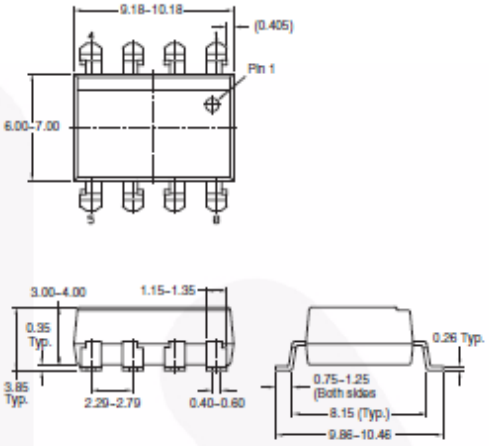


Package Dimensions

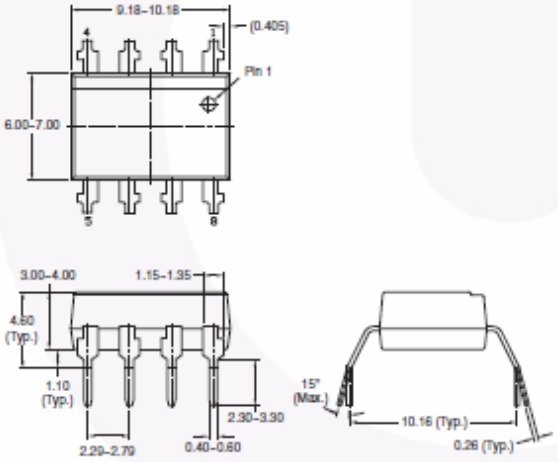
Through Hole



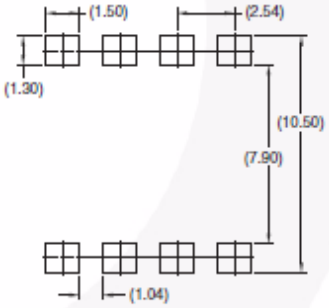
Surface Mount



0.4" Lead Spacing



Recommend Pad Layout for Surface Mount Leadform

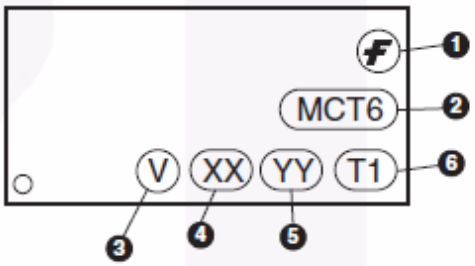


Note:
All dimensions are in millimeters.

Ordering Information

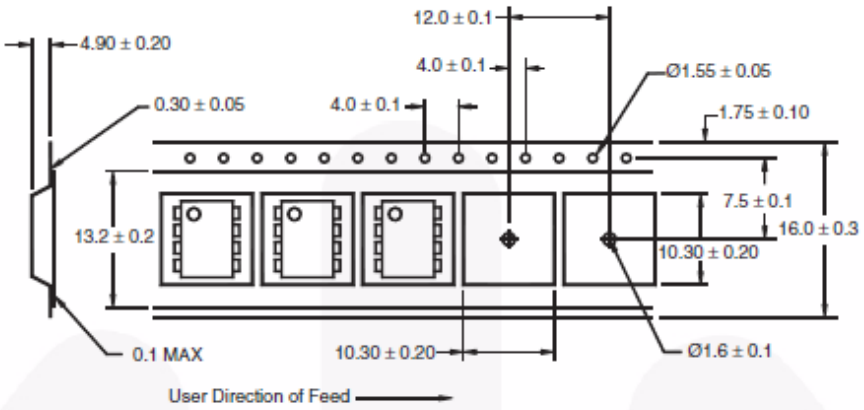
Option	Example Part Number	Description
No Option	MTC6	Standard Through Hole
S	MTC6S	Surface Mount Lead Bend
SD	MTC6SD	Surface Mount; Tape and Reel
300	MCT6300	VDE Approved
3S	MCT63S	Surface Mount Lead Bend; VDE Approved
3SD	MCT63SD	Surface Mount; Tape and Reel; VDE Approved
300W	MTC6300W	0.4" Lead Spacing; VDE Approved

Marking Information



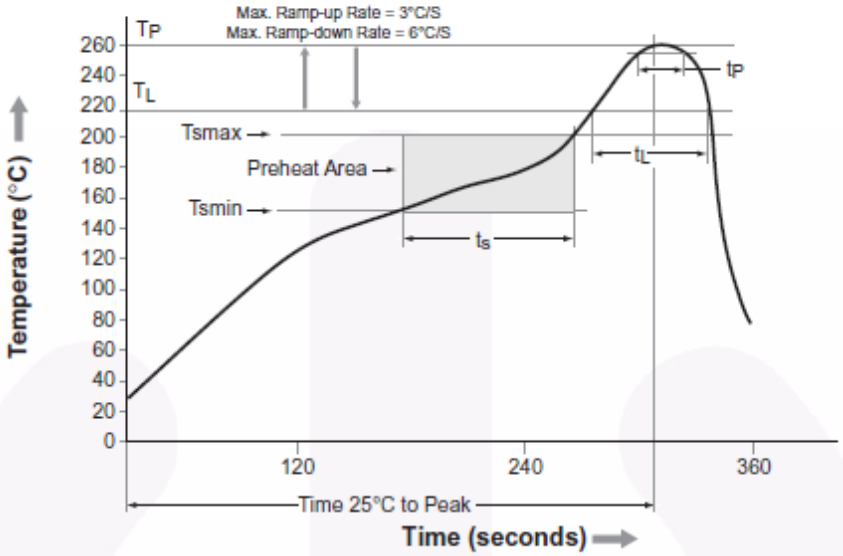
Definitions	
1	Fairchild logo
2	Device number
3	VDE mark (Note: Only appears on parts ordered with VDE option – See order entry table)
4	Two digit year code, e.g., '03'
5	Two digit work week ranging from '01' to '53'
6	Assembly package code

Carrier Tape Specifications



Note:
All dimensions are in inches (millimeters)

Reflow Profile



Profile Feature	Pb-Free Assembly Profile
Temperature Min. (Tsmin)	150°C
Temperature Max. (Tsmax)	200°C
Time (t_S) from (Tsmin to Tsmax)	60–120 seconds
Ramp-up Rate (t_L to t_P)	3°C/second max.
Liquidous Temperature (T_L)	217°C
Time (t_L) Maintained Above (T_L)	60–150 seconds
Peak Body Package Temperature	260°C +0°C / –5°C
Time (t_P) within 5°C of 260°C	30 seconds
Ramp-down Rate (T_P to T_L)	6°C/second max.
Time 25°C to Peak Temperature	8 minutes max.

FAIRCHILD
SEMICONDUCTOR®

TRADEMARKS

The following includes registered and unregistered trademarks and service marks, owned by Fairchild Semiconductor and/or its global subsidiaries, and is not intended to be an exhaustive list of all such trademarks.

AccuPower™
Auto-SPM™
Build it Now™
CorePLUS™
CorePOWER™
CROSSVOLT™
CTL™
Current Transfer Logic™
DEUXPEED®
Dual Cool™
EcoSPARK®
EfficientMax™
F®
Fairchild®
Fairchild Semiconductor®
FACT Quiet Series™
FACT®
FAST®
FastvCore™
FETBench™
FlashWriter®*
FPS™
F-PFS™
FRFET®
Global Power Resource℠
Green FPS™
Green FPS™ e-Series™
Gmax™
GTO™
IntelliMAX™
ISOPLANAR™
MegaBuck™
MICROCOUPLER™
MicroFET™
MicroPak™
MicroPak2™
MillerDrive™
MotionMax™
Motion-SPM™
OptoHiT™
OPTOLOGIC®
OPTOPLANAR®
®
PDP SPM™
Power-SPM™
PowerTrench®
PowerXS™
Programmable Active Droop™
QFET®
QS™
Quiet Series™
RapidConfigure™
™
Saving our world, 1mW/W/kW at a time™
SignalWise™
SmartMax™
SMART START™
SPM®
STEALTH™
SuperFET™
SuperSOT™-3
SuperSOT™-6
SuperSOT™-8
SupreMOS™
SyncFET™
Sync-Lock™
SYSTEM ®* GENERAL
The Power Franchise®
the power franchise
TinyBoost™
TinyBuck™
TinyCalc™
TinyLogic®
TINYOPTO™
TinyPower™
TinyPWM™
TinyWire™
TriFault Detect™
TRUECURRENT™*
μSerDes™
SerDes
UHC®
Ultra FRFET™
UniFET™
VCX™
VisualMax™
XS™

* Trademarks of System General Corporation, used under license by Fairchild Semiconductor.

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION, OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS. THESE SPECIFICATIONS DO NOT EXPAND THE TERMS OF FAIRCHILD'S WORLDWIDE TERMS AND CONDITIONS, SPECIFICALLY THE WARRANTY THEREIN, WHICH COVERS THESE PRODUCTS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF FAIRCHILD SEMICONDUCTOR CORPORATION.

As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury of the user.
2. A critical component in any component of a life support, device, or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

ANTI-COUNTERFEITING POLICY

Fairchild Semiconductor Corporation's Anti-Counterfeiting Policy. Fairchild's Anti-Counterfeiting Policy is also stated on our external website, www.fairchildsemi.com, under Sales Support.

Counterfeiting of semiconductor parts is a growing problem in the industry. All manufacturers of semiconductor products are experiencing counterfeiting of their parts. Customers who inadvertently purchase counterfeit parts experience many problems such as loss of brand reputation, substandard performance, failed applications, and increased cost of production and manufacturing delays. Fairchild is taking strong measures to protect ourselves and our customers from the proliferation of counterfeit parts. Fairchild strongly encourages customers to purchase Fairchild parts either directly from Fairchild or from Authorized Fairchild Distributors who are listed by country on our web page cited above. Products customers buy either from Fairchild directly or from Authorized Fairchild Distributors are genuine parts, have full traceability, meet Fairchild's quality standards for handling and storage and provide access to Fairchild's full range of up-to-date technical and product information. Fairchild and our Authorized Distributors will stand behind all warranties and will appropriately address any warranty issues that may arise. Fairchild will not provide any warranty coverage or other assistance for parts bought from Unauthorized Sources. Fairchild is committed to combat this global problem and encourage our customers to do their part in stopping this practice by buying direct or from authorized distributors.

PRODUCT STATUS DEFINITIONS

Definition of Terms

Datasheet Identification	Product Status	Definition
Advance Information	Formative / In Design	Datasheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	Datasheet contains preliminary data; supplementary data will be published at a later date. Fairchild Semiconductor reserves the right to make changes at any time without notice to improve design.
No Identification Needed	Full Production	Datasheet contains final specifications. Fairchild Semiconductor reserves the right to make changes at any time without notice to improve the design.
Obsolete	Not In Production	Datasheet contains specifications on a product that is discontinued by Fairchild Semiconductor. The datasheet is for reference information only.

Rev. I47