Lehrstuhl für Thermoprozesstechnik

Masterarbeit

Steuer- und regelungstechnische Umsetzung einer Großversuchsanlage zur Untersuchung der Flammenfortpflanzung in Staub/Luft- Gemischen

Marco Ernst Stockinger, BSc

Mai 2020

LEHRSTUHL FÜR THERMOPROZESSTECHNIK
Lehrstuhlleiter:
Univ.Prof. Dipl.-Ing. Dr.techn. Harald Raupenstrauch
Montanuniversität Leoben
Franz-Josef-Strasse 18
A-8700 Leoben
www.unileoben.ac.at

Masterarbeit

Steuer- und regelungstechnische Umsetzung einer Großversuchsanlage zur Untersuchung der Flammenfortpflanzung in Staub/Luft-Gemischen

erstellt am

Lehrstuhl für Thermoprozesstechnik

Vorgelegt von:
Marco Ernst Stockinger, BSc
01335192

Betreuer:
Univ.Prof. Dipl.-Ing. Dr.techn. Harald Raupenstrauch
Dipl.-Ing. Katja Hüttenbrenner
Dipl.-Ing. Michael Hohenberger

Leoben, 2.5.2020

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt, und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Ich erkläre, dass ich die Richtlinien des Senats der Montanuniversität Leoben zu "Gute wissenschaftliche Praxis" gelesen, verstanden und befolgt habe.

Weiters erkläre ich, dass die elektronische und gedruckte Version der eingereichten wissenschaftlichen Abschlussarbeit formal und inhaltlich identisch sind.

Datum 29.04.2020

Marco Stockinger

Unterschrift Verfasser/in
Marco Ernst, Stockinger

Kurzfassung:

Staubexplosionen führen schon seit Jahrhunderten zu großen Sachschäden und leider auch zu tödlichen Zwischenfällen. Der Lehrstuhl für Thermoprozesstechnik an der Montanuniversität Leoben forscht schon seit Jahren an dieser Thematik. Das Wissen wie sich die Flammenfront in einem Staub-Luft-Gemisch fortbewegt ist für die Auslegung der Sicherheitseinrichtungen notwendig. Um unter geringen turbulenten Bedingungen die Flammengeschwindigkeit zu messen, wurde am Lehrstuhl eine Apparatur nach der TUBE Methode konstruiert. Der Eintrag des Staubes in die Anlage ist ein für den Versuch wesentlicher Bestandteil. Der Eintrag durch Einrieselung von oben hat sich aufgrund der Einfachheit und der hohen Genauigkeit bewährt. Die Steuer- und Regelgrößen dieses Prozesses sind für die Reproduzierbarkeit der Explosionsversuche unabdingbar. Im Zuge dieser Arbeit sind die staubfördernden Rührer mit Wägezellen und Sensorik ausgestattet worden. Zusätzlich wurde die gesamte Anlagensteuerung und Messelektronik für die Flammengeschwindigkeit in LabVIEW programmiert. Ausgeführt wird die Steuerung über einen Arduino Mega 2560, der zusammen mit weiteren elektronischen Bauteilen in einem Schaltkasten installiert und verkabelt wurde. Die Anlage wurde erfolgreich in Betrieb genommen.

Abstract:

For centuries dust explosions have been causing great damage to facilities and even lead to fatal incidents. The Chair of Thermo Process Technology at the Montan University Leoben is doing researches on this topic. The knowledge of how the flame front spreads in a dust-air mixture is necessary to develop safety devices. In order to measure the flame speed under low turbulent conditions, an apparatus was constructed at the department using the TUBE-Method. Inserting dust into the system is an essential part of the experiment. Trickling dust from above has proven itself due to its simplicity and high accuracy. The control variables of this process are an important feature for the reproducibility of the explosion tests. In the scope of this work, the dust-promoting stirrers were equipped with load cells and sensors. In addition, the entire system control and measuring electronics for the flame speed were programmed via LabVIEW. The control is carried out with the help of an Arduino Mega 2560, which was installed and wired together with other electronic components in a control box. The test facility was successfully commissioned.

Inhaltsverzeichnis

Akronyme

ADC	Analog-Digital Wandler
ASCII	American Standard Code for Information Interchange
M	Medianwert
LWL	Lichtwellenleiter
MZE	Mindestzündenergie
OEG	Obere Explosionsgrenze
PLA	Polylactide
PWM	Pulsweitenmodulation
TPT	Lehrstuhl für Thermoprozesstechnik
UEG	Untere Explosionsgrenze
VI	Virtuelles Instrument
µC	Mikrokontroller

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Ausgangslage

Schon seit Jahrhunderten ist die Gefahr, die von Staubexplosionen ausgeht, bekannt. Die Vergangenheit zeigt, dass es immer wieder zu großen Sachschäden und leider auch zu tödlichen Zwischenfällen kommt. Für die entsprechende Auslegung von Sicherheitseinrichtungen werden sicherheitstechnische Parameter herangezogen. Diese Kenngrößen sind grundsätzlich von der Messmethode abhängig, weshalb die Übertragung auf den realen Anwendungsfall eine Schwierigkeit darstellt. Eine Größe, die vom turbulenten Stoff-, Energie- und Impulsaustausch nicht beeinflusst wird und somit weitgehend unabhängig ist, ist die laminare Flammengeschwindigkeit. Am Lehrstuhl für Thermoprozesstechnik (TPT) wird schon seit einiger Zeit an der Messmethodik dieser Kenngröße geforscht. Dafür wurde eine Versuchsapparatur nach der TUBE-Methode im Labormaßstab entwickelt.[1]

Zur Überprüfung der Anwendbarkeit der Flammengeschwindigkeit als unabhängige Kenngröße muss sie in größerem Maßstab gemessen werden. Zu Beginn dieser Arbeit war bereits das zylindrische Explosionsrohr inklusive der benötigten Mechanik vorhanden. Zudem war bereits die Hardware für den Staubeintrag und des Zündmechanismus konstruiert und hergestellt. Die Messtechnik sollte soweit wie möglich von einer kleineren Apparatur übernommen werden und verbessert werden.

1.1.1 Aufbau der Versuchsanlage

Mit dem Bau des vier Meter langen vertikalen Explosionsrohres wurde ein externes Unternehmen beauftragt. Das Rohr besteht aus einem Stahlmantel, der mittels eines Stahlgerüstes fixiert ist. Die Apparatur ist so ausgelegt, dass sie in Freiluft aufgebaut wird. Sie besteht zudem aus einem Kopf- und Bodenteil, welche jeweils vom Hauptteil getrennt werden können. Am Kopfteil befinden sich zudem zwei Klappen, die im Falle einer Explosion öffnen und damit den Druck in der Anlage abbauen. Des Weiteren verfügt die Konstruktion über einen ausfahrbaren Bodenteil. In Abbildung 1 ist die Versuchsanlage bildlich dargestellt.

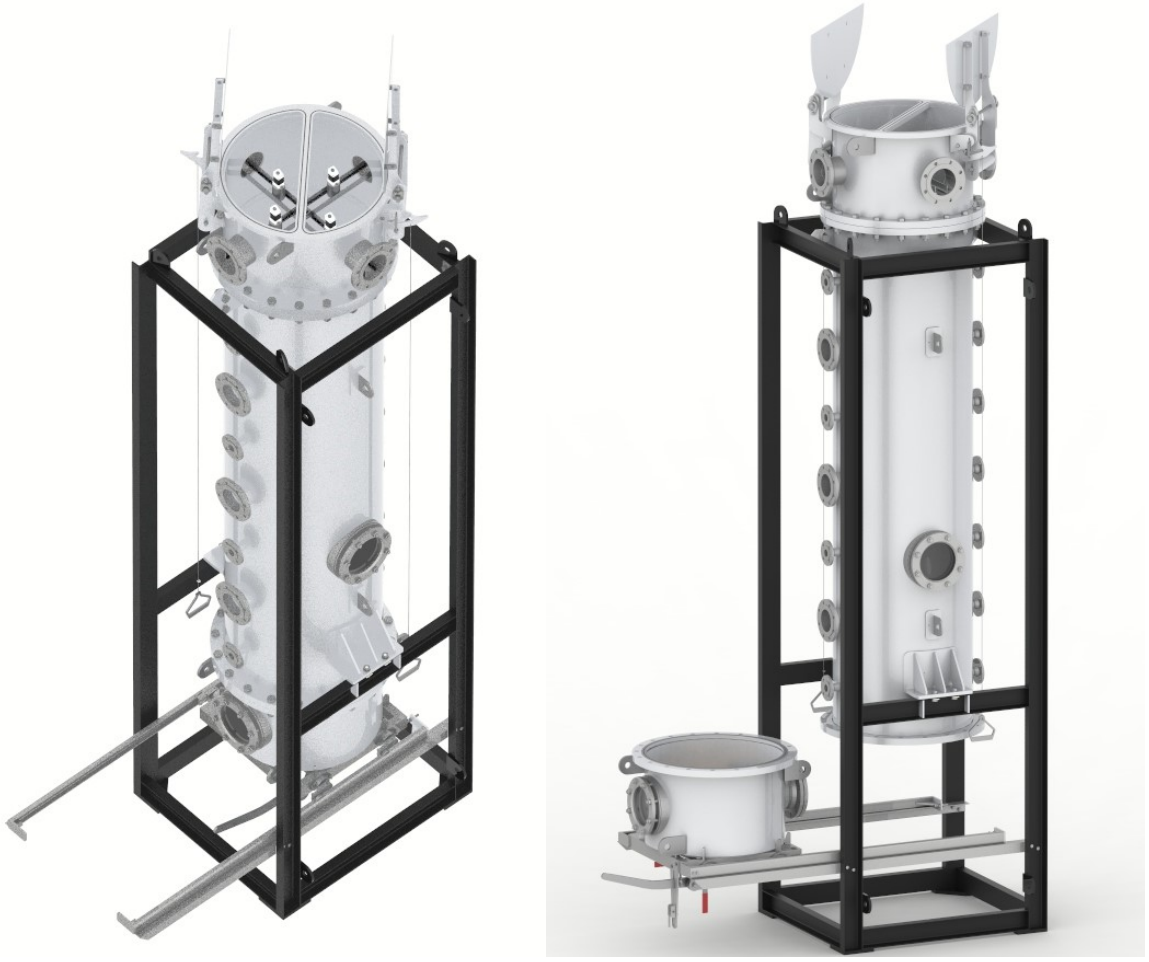


Abbildung 1: Modell der Versuchsanlage, links mit eingefahrenem Bodenteil, rechts mit offenem Bodenteil [2]

1.1.2 Kurzbeschreibung des geplanten Versuches

Der Staubeintrag im Kopfteil besteht aus vier Rührer, die den brennbaren Staub möglichst gleichmäßig in das Explosionsrohr einrieseln. Dadurch bildet sich ein explosionsfähiges

Staub/Luft- Gemisch. Diese explosionsfähige Atmosphäre soll mittels einer Zündeinrichtung, welche im Fußteil des Rohrer angebracht ist, zur Explosion gebracht werden. Die Flamme breitet sich von der Zündstelle weg nach oben aus. Auf dem Weg wird die Flammenfront von Fotodioden und Thermoelementen diktiert. Aus den detektierten Werten wird dann die Flammengeschwindigkeit ermittelt.

1.2 Problemstellung

Zu Beginn dieser Arbeit war bereits ein Staubeintrag vorhanden, welcher aus vier Rührern besteht. Die Förderrate variiert von Rührer zu Rührer bei gleicher Drehzahl und ist zudem zeitabhängig. Daher kann die geförderte Staubmenge nicht geregelt werden

Um einen Versuch mit der Anlage durchzuführen zu können, musste noch sämtliche Messelektronik erstellt werden, sie sollte so weit wie möglich von der bereits vorhanden Laboranlage übernommen werden. Des Weiteren gab es noch keine Software die den Versuchsaufbau steuern konnte.

Einige Komponenten waren zwar bereits entwickelt, mussten aber zum Teil noch verfeinert und richtig in die Anlage integriert werden.

1.3 Zielsetzung

Ziel dieser Arbeit ist es, sämtliche Schlüsselkomponenten der Anlage soweit fertig zu stellen, dass Versuche mit der Anlage durchgeführt werden können.

Dafür sind folgende Schritte notwendig:

- Der Staubeintrag muss so abgeändert werden, dass sämtliche Rührer einzeln regelbar sind. Dies soll gewährleisten, dass der Staub möglichst homogen und präzise eingebracht werden kann.
- Sämtliche Messelektronik für die Flammenerfassung muss entwickelt und in die Anlage integriert werden.
- Eine Software zum Betreiben der Hardware muss entwickelt werden.
- Sämtliche Komponenten müssen schließlich zusammengefügt und erprobt werden.

2 Theoretische Grundlagen

2.1 Grundlegendes zu Staubexplosionen

2.1.1 Einleitung

Die Gefahren und die Folgen von Staubexplosionen in der Industrie sind seit Jahrzenten bekannt. Jeder Staub der aus brennbarem Material besteht ist im aufgewirbelten Zustand explosionsfähig, wenn dieser eine ausreichend kleine Korngrößenverteilung besitzt. Bei einer Korngröße von über 500 µm kommt es nur sehr unwahrscheinlich zu einer Staubexplosion. Die Korngröße eines Staubes ist einer der wichtigsten Parameter. Stäube kommen in technischen Prozessen als Produkt gewollter Herstellung sowie als Abfall- oder Nebenprodukt vor. Die Größe der einzelnen Staubteilchen variieren in der Regel sehr stark. Bei Korngrößen von 100-500 µm spricht man in der Praxis von Stäuben, bei Korngrößen von 100-300 µm spricht man von sogenannten Feinstäuben. Als Feinststäube werden Gemische mit einer Korngröße von 30-100 µm bezeichnet. [3]

Um eine Staubexplosion handelt es sich immer dann, wenn ein Staub Luft Gemisch nach einer Entzündung eine selbständige Flammenausbreitung sowie eine Drucksteigerung auftritt. [3] Nähere Informationen können den Quellen [4], [5] und [3] entnommen werden.

Der zeitliche Verlauf einer Staubexplosion kann unter anderem mit der Flammengeschwindigkeit beschrieben werden. Grundlagen hierzu können [1] und [6] entnommen werden.

2.2 Verfahren zur Bestimmung von Flammengeschwindigkeiten

Es gibt mehrere Verfahren, um die Flammengeschwindigkeit zu bestimmen. Zu den diesen Verfahren zählen unter anderem:

- Bestimmung der Flammengeschwindigkeit durch die Auswertung einer Slow-Motion Aufnahme der Flamme
- Detektion der Flammenfront durch die entstehende Temperaturänderung
- Bestimmung Flammengeschwindigkeit durch optische Messverfahren
- Bestimmung der Flammengeschwindigkeit durch die Änderung des elektrischen Widerstandes

2.2.1 Verfahren mittels Temperaturmessung

Die Flammenfront kann durch einen Anstieg der Temperatur erfasst werden. Wird die Temperatur an mehreren Stellen detektiert, kann man daraus die Geschwindigkeit der Flammenfront bestimmen, indem man die Strecke zwischen den beiden Messpunkten durch die Zeitdifferenz der Temperaturerhöhungen dividiert.

Grundsätzlich wird zwischen der berührenden und der nicht berührenden Temperaturmessung unterschieden.

Berührende Temperaturmessung:

Bei dieser Art von Temperaturmessung ist eine gute Wärmeanbindung an das Messobjekt nötig, um zu gewährleisten, dass das Thermometer und das zu messende Objekt im thermischen Gleichgewicht sind. Die Messung kann aufgrund unterschiedlicher Effekte geschehen. Zu diesen Methoden zählen unter anderem: [7]

- Platin- Widerstandsthermometer
- Halbmetall- Widerstandsthermometer
- Thermoelemente

Thermolemente nützen den sogenannten Seebeck-Effekt aus. Dieser Effekt besagt, dass sich ein Stromfluss bildet, wenn der Stromkreis aus zwei verschiedenen metallischen Leitern besteht und beide Verbindungstellen der Leiter sich auf einem unterschiedlichen Temperaturniveau befinden. Thermoelemente messen eine Temperaturdifferenz und keine absolute Temperatur. [8]

Nicht berührende Temperaturmessung:

Die Strahlungspyrometrie und die Thermografie funktionieren auf dem Prinzip, dass Körper mit einer Temperatur über den absoluten Nullpunkt Infrarotstrahlung emittieren. Die Temperatur kann über das Plancksche Strahlungsgesetz ermittelt werden. Ist der Körper nicht transparent kann nur die Oberflächentemperatur bestimmt werden. [7]

2.2.2 Verfahren mittels Photodioden

Bei diesem Verfahren werden die Lichtimpulse der Flammenfront mittels eines Lichtwellenleiters (LWL) erfasst und anschließend zu der Messelektronik weitergeleitet.

Photodioden

Bei der Photodiode handelt es sich um eine Diode, die einfallendes Licht an einem p-n Übergang in elektrischen Strom wandelt. Die Photodiode kann aus verschiedenen Halbleitern gefertigt werden, je nach dem welcher Wellenbereich das einfallende Licht aufweist. Der entstehende Strom der Photodiode kann mittels einer Messschaltung ausgewertet werden. [9]

2.3 LabVIEW

2.3.1 Allgemein

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) ist eine grafische Programmiersprache. Bei grafischen Programmiersprachen werden im Gegensatz zu textbasierten Programmiersprachen anstatt von Textzeilen Symbole verwendet, um ein Programm zu erstellen. Der Datenfluss bestimmt in LabVIEW die Ausführungsreihenfolge. [10]

LabVIEW wird vor allem für mess- steuer- und regelungstechnischen Anwendungen verwendet. Die Entwicklungsumgebung basiert auf der grafischen Programmiersprache G, die speziell zur Abbildung technischer Prozesse entwickelt wurde. LabVIEW hat die Möglichkeit physikalische Signale in ein Programm einzulesen, zu verarbeiten und darzustellen. Es ermöglicht dem Benutzer sein eigenes Messgerät zu designen. Ein LabVIEW Programm bezeichnet man als VI (Virtuelles Instrument). [10]

Abbildung 2: Logo von LabVIEW

LabVIEW wurde von National Instruments entwickelt und erschien zum ersten Mal im Jahr 1986. Abbildung 2 zeigt das aktuelle Logo, Stand 2019, der Programmiersprache LabVIEW.

2.3.2 LabVIEW Umgebung

Jede LabVIEW VI enthält die folgenden drei Komponenten:

- Frontpanel: stellt die Benutzeroberfläche dar
- Blockdiagramm: enthält den grafischen Quellcode und definiert daher die Funktion der VI
- Symbol und Anschlussfeld: dient zur Identifikation der VI, so dass diese von einer anderen VI verwendet werden kann

Symbol und Anschlussfeld: Jede VI hat ein Symbol, welche in der rechten oberen Ecke, wie in Abbildung 3 (roter Kreis), des Frontpanels und Blockdiagramms dargestellt wird. Bei dem Symbol handelt es sich um eine grafische Darstellung der VI. Das Symbol kann durch einen Doppelklick editiert werden. [11]

Um eine VI in einer anderen VI verwenden zu können, benötigt man in der Regel Anschlüsse. Das Anschlussfeld stellt die Gesamtheit der einzelnen Anschlüsse dar. Man unterscheidet zwischen Ein- und Ausgängen. Eine VI kann bis zu 28 Anschlüsse besitzen, je nach Datentyp wird der Anschluss in einer unterschiedlichen Farbe dargestellt. [11]

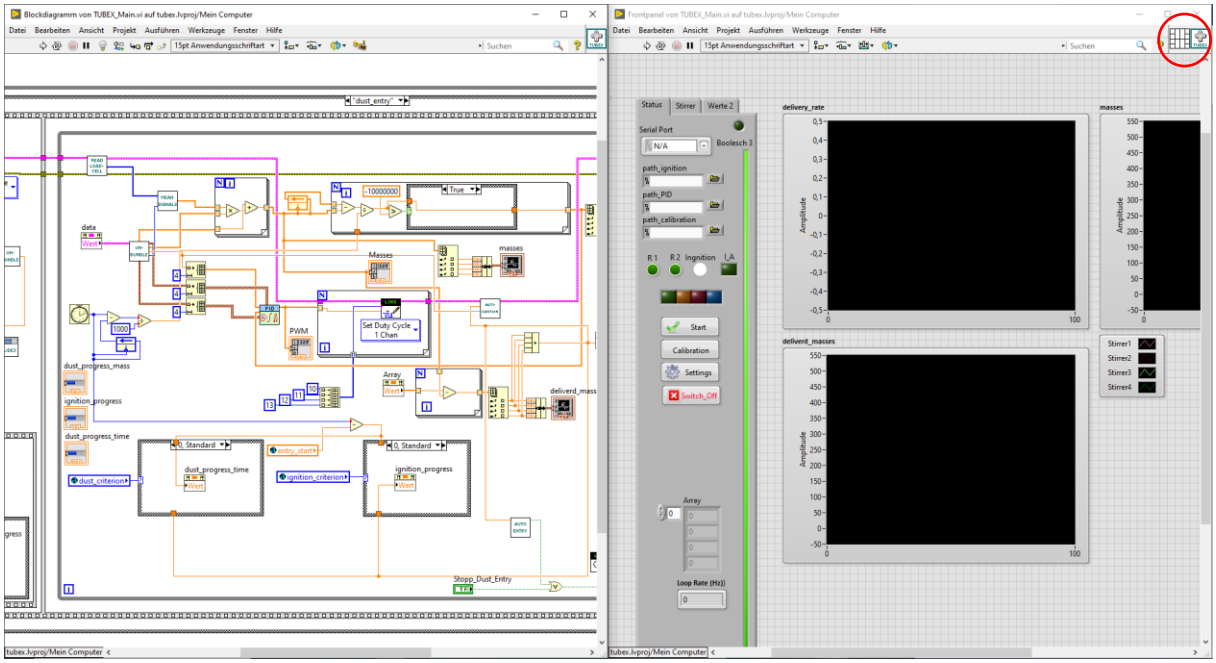


Abbildung 3: Das sogenannte Blockpanel und Frontpanel einer VI

Frontpanel: Im sogenannten Frontpanel werden sämtliche Bedien- und Anzeigeelemente dargestellt. Als Bedienelemente können unter anderem Knöpfe, Drehregler und Drucktasten dienen. Die Daten der Bedienelemente werden an das Blockdiagramm der VI übergeben. [10]

Blockdiagramm: Das Blockdiagramm enthält sämtlichen Quellcode, welcher nach der Erstellung des Frontpanels der VI mittels grafischer Programmierung hinzugefügt wird. Das Blockdiagramm steuert zudem die Elemente des Frontpanels. [10]

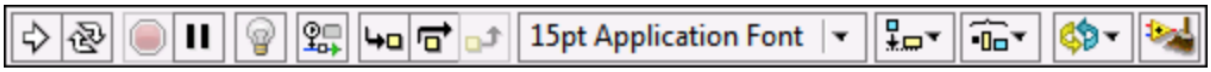


Abbildung 4: Symbolleiste des Blockdiagramms

Jedes Fenster besitzt eine Symbolleiste, in Abbildung 4 wird die Symbolleiste des Blockdiagramms angezeigt. Mittels dieser Symbolleiste kann eine VI ausgeführt, gestoppt oder bearbeitet werden. Im Folgenden wird auf die ersten vier Icons eingegangen. [11]

Abbildung 5: Verschiedene Anzeigen des Icons Ausführen

Beim ersten Icon handelt es sich um das sogenannte **Ausführen** Icon, welches vier verschieden Zustände anzeigen kann, die in der Abbildung 5 dargestellt sind. Klickt man auf

die erste dargestellte Form wird das Programm kompiliert und anschließend ausgeführt. Der weiße Pfeil zeigt zudem an, dass die VI als SubVI verwendet werden kann. [10]

Der zweite schwarze Pfeil zeigt, dass die VI ausgeführt wird und dabei nicht von einer anderen VI aufgerufen wird, sprich es handelt sich um keine SubVI. Der dritte Pfeil wird angezeigt falls die VI von einer anderen VI aufgerufen wurden, dabei dient die VI als SubVI. Das vierte Symbol zeigt an, dass die VI nicht ausgeführt werden kann, da ein Fehler beim Bearbeiten der VI aufgetreten ist. Klick man diesen Pfeil an wird eine Liste mit allen Fehlern und Warnungen angezeigt. [10]

Bei dem zweiten Icon welches in Abbildung 4 dargestellt ist handelt es sich um das sogenannt **Wiederholt ausführen** Icon. Klickt man dieses wird die VI solange ausgeführt bis man sie anhält oder abbricht. Ein wiederholtes Klicken deaktiviert die Funktion wieder. [10]

Das nächste Icon stellt die Schaltfläche **Ausführen abbrechen** dar. Wird dieses Icon gedrückt wird die VI sofort angehalten. Wird die VI von mehreren anderen laufenden VIs verwendet, wird dieses Icon ausgegraut. [10]

Das letzte Icon stellt die sogenannte Schaltfläche **Pause** dar, beim Klicken dieses Icons geklickt, wird das Programm solange angehalten bis das Icon erneut geklickt wird. [10]

2.3.3 Erstellen des Frontpanels

Das Frontpanel ist die Benutzeroberfläche, meistens wird zuerst das Frontpanel erstellt und anschließend das Blockdiagramm. Durch Hilfe von Bedien- und Anzeigeelementen, welche interaktive Aus- und Eingänge darstellen, wird das Frontpanel erstellt. Über das Frontpanel kann man auch Daten zwischen verschiedenen VIs austauschen. [11]

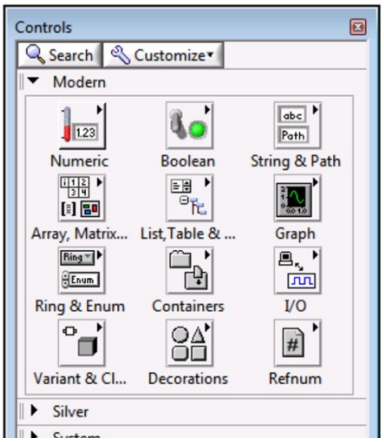


Abbildung 6: Elementpalette

Alle Bedien- und Anzeigeelemente haben einen bestimmten Datentyp. Zu den gängigsten Datentypen zählen „numerisch“, „boolesch“ und „strings“.

Über die sogenannte Elementpalette kann auf alle Bedien- und Anzeigeelemente zugegriffen werden und in das Panel hinzugefügt werden. Die Palette ist in verschiedene Kategorien unterteilt. Abbildung 6 zeigt die Elementpalette, man kann die verschiedenen Kategorien gut erkennen. [12]

Boolesche Bedien- und Anzeigeelemente

Mit Hilfe von booleschen Bedien- und Anzeigeelementen können Daten vom Dateityp „boolesch“ Werte ein- und ausgegeben werden. Der Datentyp boolesch hat nur zwei Zustände, TRUE und FALSE. Mit booleschen Elementen werden meistens Schalter und LEDs dargestellt. In Abbildung 7 sind ein paar der häufigsten booleschen Elemente dargestellt. [12]

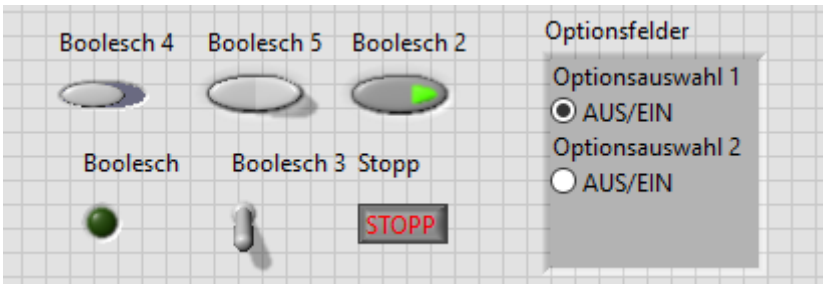


Abbildung 7: Boolesche Bedien- und Anzeigeelemente

Numerische Bedien- und Anzeigeelemente

Mit numerischen Bedien- und Anzeigeelementen können Zahlen von verschiedensten Datentypen ein- und ausgegeben werden. Es gibt eine große Palette von numerischen Elementen. Die Palette reicht von Drehknöpfen über Eingabefeldern bis hin zu Skalen. In Abbildung 8 sind ein paar der häufigsten numerischen Elemente dargestellt. [12]

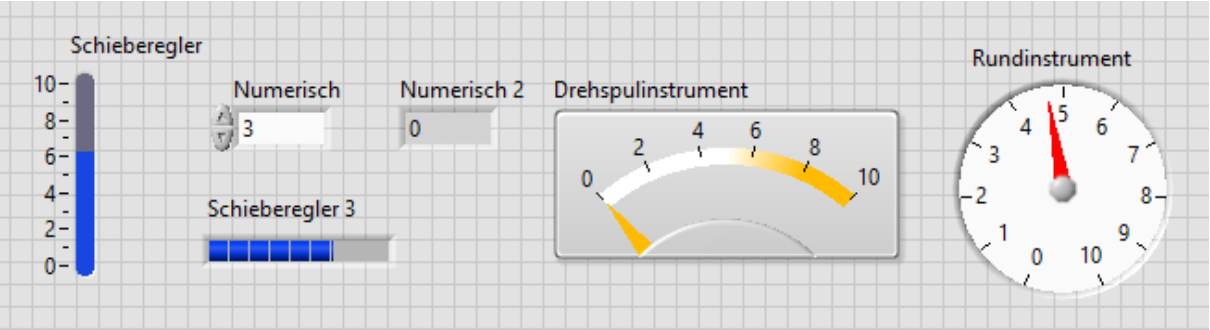


Abbildung 8: Numerische Bedien- und Anzeigeelemente

String Bedien- und Anzeigeelemente

Mit diesen Objekten werden ASCII- Zeichen dargestellt. Diese Felder dienen unter anderem zur Eingabe von Passwörtern, Benutzernamen und Texten. In Abbildung 9 sind ein paar der häufigsten string Elemente dargestellt. [12]

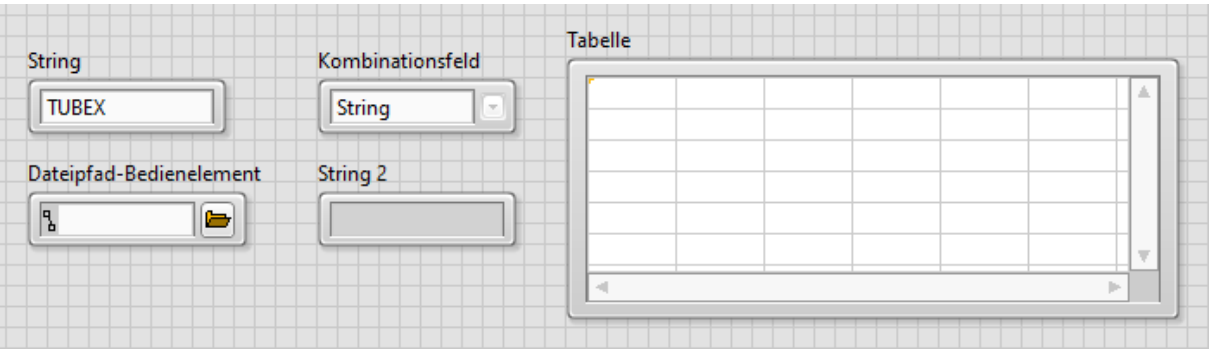


Abbildung 9: Eine kleine Auswahl von String Bedien- und Anzeigeelementen

2.3.4 Erstellen des Blockdiagramms

Die im Frontpanel erstellten Elemente werden im Blockdiagramm als Symbol dargestellt. Diese Symbole können Ein- und Ausgänge besitzen. Über diese Ein- und Ausgänge werden Daten zwischen den Frontpanel und Blockdiagramm ausgetauscht. Diese Anschlüsse sind das Gegenstück zu Konstanten in textorientierten Programmiersprachen. Es gibt Anschlüsse für Bedien-, Anzeigeelemente und Knoten. In der Regel werden Eingänge im Symbol links angezeigt und Ausgänge rechts am Symbol. [12]

Mithilfe von Knoten können Daten zwischen verschiedenen VIs und anderen Stellen ausgetauscht werden. Knoten sind Funktionen, SubVIs, Schleifen und Strukturen. Einfache Knoten sind zum Beispiel die Addition, Multiplikation oder eine While-Schleife, siehe Kapitel 2.3.5. [13]

Über die sogenannte Funktionspalette kann auf VIs, Funktionen und Konstanten zugegriffen werden und in das Panel hinzugefügt werden. Die Palette ist in verschiedene Kategorien unterteilt. Abbildung 10 zeigt die Funktionspalette und man kann die verschiedenen Kategorien gut erkennen. [12]

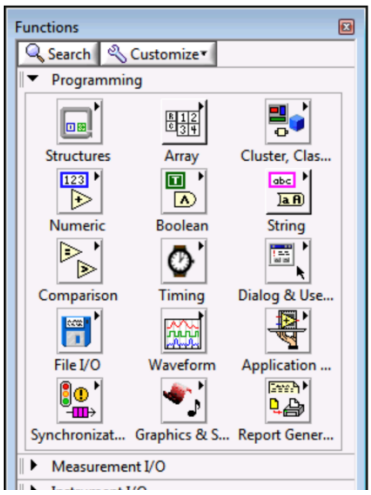


Abbildung 10: Funktionspalette

Verbindungen

Im Blockdiagramm werden Daten zwischen Objekten mittels sogenannter Verbindungen übertragen. Jede Verbindung hat eine bestimmte Datenquelle, die mit einer Vielzahl von Objekten verbunden werden kann. Je nach Datentyp wird die Verbindung in unterschiedlicher Farbe, Darstellung und Linienstärke dargestellt. In Tabelle 1 sind die Verbindungen der wichtigsten Datentypen angeführt. Fehlerhafte Verbindungen werden schwarz gestrichelt mit einem roten X in der Mitte dargestellt. Eine fehlerhafte Verbindung ist zum Beispiel, wenn sie zwei Verbindungen unterschiedlichen Datentyps miteinander verbinden. Eine fehlerhafte Verbindung ist in Abbildung 11 dargestellt. [12]

Abbildung 11: Fehlerhafte Verbindung in LabVIEW

Tabelle 1: Darstellung gängiger Verbindungen

Datentyp	Skalar	1D- Array	2D- Array	Farbe
Numerisch				Orange
Boolesch				Grün
String				Rosa

2.3.5 Schleifen und Strukturen

While- Schleife

Will man einen bestimmten Programmabschnitt mehrmals hintereinander ausführen, wird dies mittels sogenannter Schleifen verwirklicht. LabVIEW beinhaltet zwei Schleifenarten, die While- und For-Schleife. Die While-Schleife wird solange ausgeführt bis eine bestimmte Bedinung erfüllt ist. Am Ende des Schleifendurchlaufes wird überprüft ob die Bedingung erfüllt ist, weshalb der Code in der Schleife mindestens einmal ausgeführt wird. Die While-Schleife kommt immer dann zum Einsatz, wenn im Vorhinein nicht bekannt ist wie oft die Schleife ausgeführt werden soll. In Abbildung 12 ist der Code für eine einfache While-Schleife dargestellt. [10]

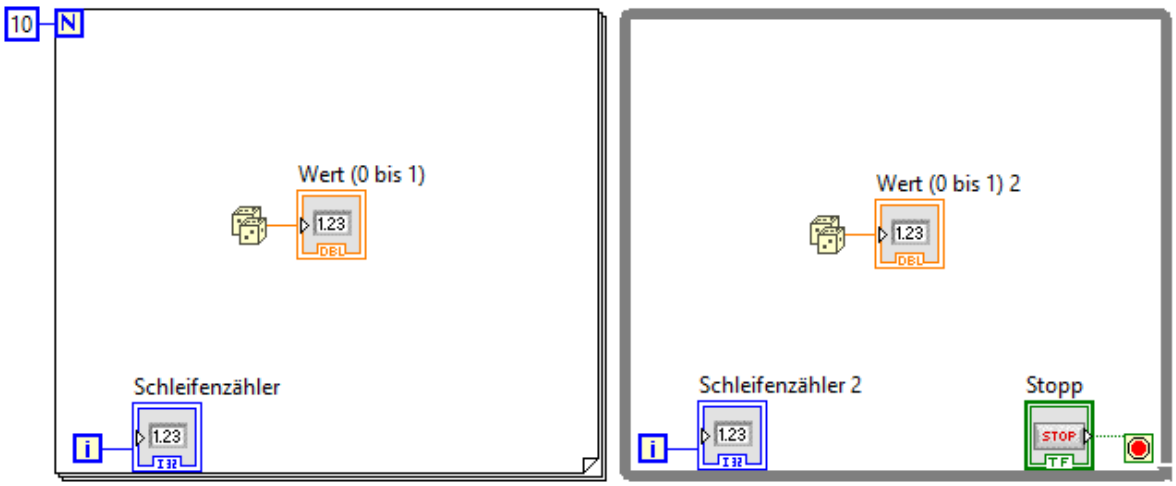


Abbildung 12: links: for-Schleife, rechts: while-Schleife

Eine While-Schleife besteht aus drei Elementen:

Schleifenzähler: Hierbei handelt es sich um einen numerischen Ausgang, welcher die aktuelle Anzahl der Schleifendurchläufe ausgibt. Der Schleifenzähler wird durch ein blaues „i“ dargestellt, das „i“ steht für Indexvariable. [10]

Bedienungsanschluss: Hierbei handelt es sich um einen booleschen Eingang. Es kann über einen rechten Mausklick eingestellt werden, ob die Schleife bei FALSE oder TRUE abgebrochen wird. Standardmäßig wird die Schleife abgebrochen falls an dem Bedienungsanschluss TRUE anliegt. Der Bedienungsanschluss wird durch einen schwarz umrandeten roten Punkt dargestellt. Dieser Anschluss muss immer beschalten sein, ansonsten kann die VI nicht gestartet werden. [10]

Ausführbarer Code: In diesem Teil der Schleife wird sämtlicher Code eingefügt welcher mehrfach ausgeführt werden soll. [10]

For-Schleife

Bei der sogenannten for-Schleife handelt es sich um eine kopfgesteuerte Schleife. Das heißt, dass vor dem Durchlaufen überprüft wird ob die Schleife ausgeführt werden soll. Diese Art der Schleife wird verwendet, wenn im vorhinein bekannt ist wie oft die Schleife durchlaufen wird, hierfür wird der Schleife die Anzahl der Ausführungen mitgeteilt. [10]

Die For-Schleife besteht standardmäßig aus drei Elementen und kann um bis zu drei Elemente erweitert werden:

Schleifenzähler: Gleiche Funktion wie bei der While-Schleife

Ausführbarer Code: Gleiche Funktion wie bei der While-Schleife

Zählanschluss: Hierbei handelt es sich um einen numerischen Eingang, welcher angibt wie oft die Schleife durchlaufen werden soll. Wichtig ist, dass dieser Eingang immer beschalten ist, ansonsten kann die VI nicht gestartet werden. [10]

Bedienungsanschluss(optional): Optional kann einer For-Schleife ein Bedienungsanschluss hinzugefügt werden, welcher die gleiche Funktion wie bei der While-Schleife besitzt. [10]

Case Strukturen

Mittels sogenannter Case-Strukturen sind Fallunterscheidungen möglich. Je nach dem welcher Datentyp an dem sogenannten Selektoranschluss angeschlossen wird, unterscheidet man zwischen booleschen, numerischen, String und Enum Case Strukturen. [10]

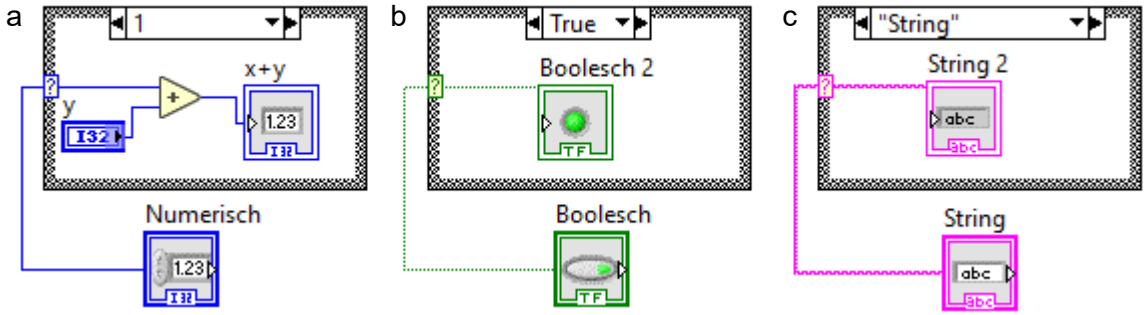


Abbildung 13: Verschiedene Case-Strukturen, (a) nummerisch, (b) boolsch, (c) string

Jede Case Struktur hat zumindest zwei Unterdiagramme, die sogenannten Cases. In Abbildung 13 sind verschiedene Case Strukturen dargestellt. Eine Case Struktur besteht ausfolgenden Elementen:

Selektoranschluss: Hierbei handelt es sich um den Anschluss, der entscheidet welcher Case ausgeführt wird. Bei booleschen Cases gibt es nur zwei Cases, TRUE und FALSE. Bei anderen Datentypen gibt es mindestens zwei Cases, wobei es auch mehr sein können. Je nach angeschlossenem Datentyp ändert sich die Farbe des Anschlusses. [10]

Selektorfeld: Über dieses Feld kann man zwischen den einzelnen Cases wechseln. Wichtig ist, dass hier ein Case als Standard verwendet wird, der ausgeführt werden soll falls kein anderer Case zutrifft. [10]

Case: Das Unterdiagramm, welches auch als Case bezeichnet wird, enthält den Code, der ausgeführt werden soll, falls der Wert am Selektoranschluss mit jenem im Selektorfeld übereinstimmt. [10]

2.3.6 Datenstrukturen

String Datentyp

Dieser Datentyp ist ein plattformunabhängiger Datentyp. Es handelt sich hierbei um eine Zeichenkette aus darstellbaren und nicht darstellbaren ASCCI Zeichen. Nichtdarstellbare ASCCI- Zeichen sind unter anderem ein Leerzeichen, Umlaute oder etwa ein Zeilenumbruch. [14]

Es gibt verschiedene Darstellungsmöglichkeiten für Strings. Die hier am häufigsten verwendeten Strings sind die sogenannte „Normale Anzeige“ und die sogenannte „Passwortanzeige“. Bei der Passwortanzeige wird für jedes Zeichen ein Stern „*“ dargestellt. [14]

Numerischer Datentyp

Numerische Daten können in LabVIEW in komplexen Zahlen, Fließkomawerten, Integer, vorzeichenlosen Integer und Festkomawerte dargestellt werden. Es gibt eine Menge von unterschiedlichen numerischen Datentypen, die sich vor allem durch die Anzahl der Bits unterscheiden. In Tabelle 2 sind exemplarisch einige numerische Datentypen aufgelistet. [14]

Tabelle 2: Auswahl von numerischen Datentypen [14]

Anschluss	Datentyp	Bits	Dezimalstellen
	Fließkommazahl, doppelte Genauigkeit	64	15
	Vorzeichenbehafteter Word-Integer	16	2
	Vorzeichenbehafteter Long-Integer	64	18
	Vorzeichenloser Word-Integer	16	4

Numerischen Werten können physikalische Einheiten, wie Kilogramm, Sekunde oder Meter zugewiesen werden. Die Einheit wird in der sogenannten Einheitenbeschriftung angezeigt,

welche standardmäßig nicht sichtbar ist. Wenn sie einem Objekt eine Einheit hinzufügen, können sie dieses nur mit Objekten verbinden, die eine kompatible Einheit besitzen. [14]

Boolescher Datentyp

In LabVIEW werden boolesche Daten durch acht Bit dargestellt. Ist der achte Bit 0 so lautet der Wert FALSE. Jede andere Kombination bedeutet, dass der Wert TRUE ist. Mit booleschen Daten werden digitale Daten dargestellt. Oft werden damit Elemente für die Ausführung von digitalen Strukturen simuliert. Wird ein boolesches Element als Schalter im Frontpanel verwendet ist es wichtig das gewünschte Schaltverhalten auszuwählen. Standardmäßig wird „beim Loslassen schalten“ verwendet. Hier wird der Schalter nur geschalten nachdem die Maustaste innerhalb der grafischen Grenze des Schalters losgelassen wird. [14]

Arrays

Manchmal ist es sinnvoll Daten zusammenzufassen. Dafür gibt es zwei Möglichkeiten: die Verwendung eines Arrays oder eines Clusters. In sogenannten Arrays werden Daten eines gleichen Datentyps gespeichert. Ein Array besteht aus einzelnen Elementen und kann aus mehreren Dimensionen bestehen. Ein Element ist ein Wert im Array. Eine Dimension definiert zum Beispiel die Höhe, Länge oder Tiefe eines Arrays. Ein Array kann aus booleschen Werten, Zahlen, Strings, Cluster, Pfade und Signalverläufe bestehen.[14]

Cluster

Bei einem Cluster handelt es sich um einen Datentyp welcher es ermöglicht thematisch zusammengehörige Daten unterschiedlichen Typs zusammenzufassen. Wichtig ist, dass eine Verbindung von Clustern nur möglich ist, wenn die Anzahl die Typen und die Reihenfolge der Elemente in den Clustern gleich sind. Die Verwendung von Clustern kann auch dazu benutzt werden, um Verbindungsleitungen im Blockdiagramm einzusparen. [14]

Enums

Bei diesem Datentyp handelt es sich um einen Datentyp, welcher aus einem String und einer vorzeichenlosen Ganzzahl besteht. Dieser wird verwendet fall es begrenzte und bekannte Auswahlmöglichkeiten gibt, wie zum Beispiel Farben, Wochentage oder bestimmte Messstellen. [14]

3 Design der Hardware und Software

3.1 Staubeintrag

Wie schon in Kapitel 1.1 erwähnt, war am Beginn dieser Arbeit der Staubeintrag vorhanden. Dieser musste allerdings umgebaut werden, da es mit der vorhandenen Hardware nicht möglich war, die einzelnen Rührer mit unterschiedlichen Drehzahlen anzusteuern. Im Folgenden wird kurz auf den vor der Arbeit vorhandenen Staubeintrag eingegangen und anschließend die Erweiterungen erläutert. Es soll nur ein kurzer Überblick über die verwendeten Komponenten gegeben werden. Für eine detailliertere Dokumentation möchte ich auf die Arbeit von Dipl.-Ing. Glechner verweisen [2].

Infolge einer Brainstorming-Sitzung am TPT Lehrstuhl wurden diverse Methoden für den Staubeintrag besprochen. Die wichtigsten Anforderungen an solch einen Staubeintrag sind:

- Kontinuierliche Förderrate
- Homogene Verteilung
- Reproduzierbare Förderrate

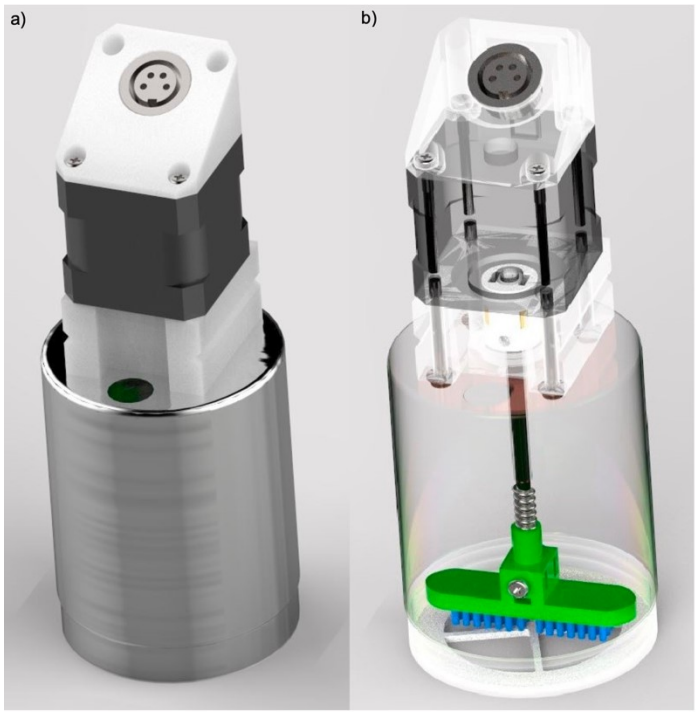


Abbildung 14: Darstellung eines Rührers [2]

Schließlich wurde in der Sitzung beschlossen ein System zu verwenden, welches ähnlich zu handelsüblichen Staubzucker- und Kakaostreuern ist.

Das Grundprinzip ist, Bürsten werden mithilfe von Federn gegen ein Sieb gedrückt. Die Bürsten werden mittels eines Schrittmotors zur Rotation gebracht. Abhängig von der Drehzahl kann die Förderrate variiert werden. Zudem wurden mehrere Siebeinsätze mit unterschiedlichen Maschenweiten angefertigt. Abbildung 14 soll das Prinzip des Staubeintrages visuell veranschaulichen.

Glechner konstruierte und fertigte sämtliche Hardware inklusive der Elektronik für die Ansteuerung der Schrittmotoren. Versuche zeigten allerdings, dass die einzelnen Rührer unterschiedliche Förderraten bei identer Drehzahl aufweisen.

In Abbildung 15 ist dieser Grund ersichtlich, zudem sind die Förderraten der einzelnen Rührer über die Zeit nicht konstant, da eine Abnahme mit kleiner werdender Füllhöhe beobachtet wurde. Der Versuch wurde bei einer Drehzahl von 60 Umdrehungen pro Minute und einem Siebeinsatzder Maschenweite 200 µm durchgeführt. Die Rührer waren voll angefüllt.

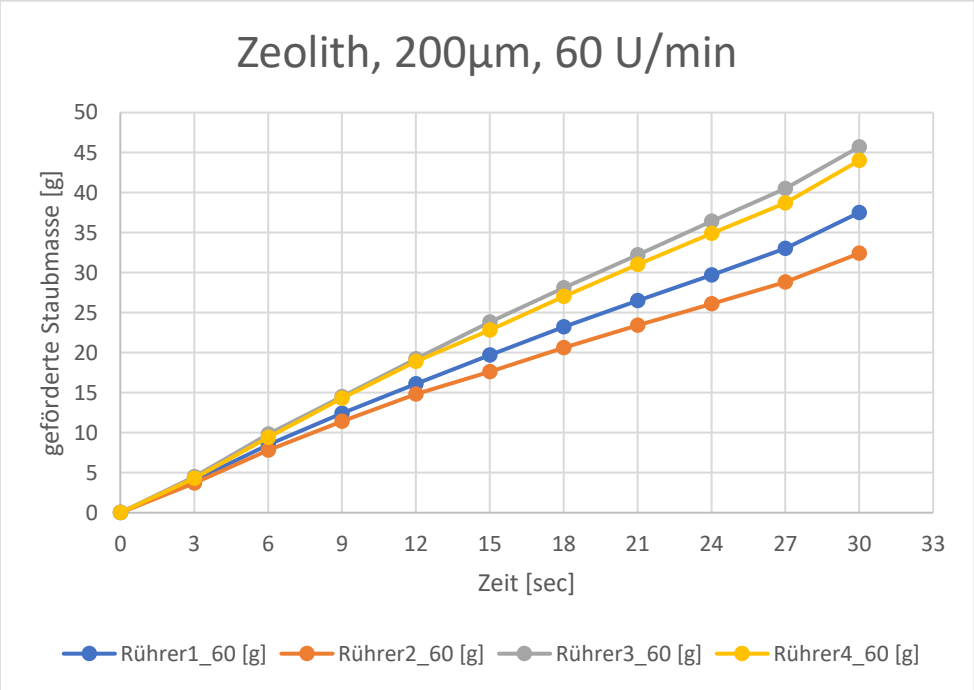


Abbildung 15: Versuch zur Ermittlung der Förderrate [2]

Da eine gleichmäßige Verteilung des Staubes über den Querschnitt eines der wichtigsten Anforderungen an den Staubeintrag ist, musste der Staubeintrag so abgeändert werden, dass dies gewährleistet werden kann. Die von Glechner erstellte Hardware war nicht dafür

ausgelegt, die Rührer mit unterschiedlicher Drehzahl anzusteuern. Infolge dieser Arbeit wurde die Elektronik zur Ansteuerung der einzelnen Rührerantriebe so erweitert, dass jeder individuell angesteuert werde kann.

3.1.1 Motorisierung

Ein Schrittmotor erfüllt alle Anforderungen für das Antreiben des Rührers, die wichtigsten sind:

- Die maximale Drehzahl soll bei über 200 Umdrehungen pro Minuten liegen
- Der Antrieb soll über das gesamte Drehzahlspektrum gut regelbar sein
- Speziell im niedrigen Drehzahlbereich soll genug Leistung aufgebracht werden

Fast alle handelsüblichen Schrittmotoren erfüllen die oben genannten Kriterien. Bei dem eingesetzten Modell handelt es sich um den Typ 17 Nema. Die wichtigsten Daten des Schrittmotors sind in Tabelle 3 dargestellt.

Tabelle 3: Wichtige Kenndaten des Schrittmotors Typ 17 Nema laut Datenblatt

Schrittwinkel	1,8°
Schritte pro Umdrehung	200
Nennstrom	1,7 A/Phase
Nennspannung	3,7 V
Haltemoment	4000 g*cm

Die Schrittmotoren werden von je 2 Spulen angetrieben, die über 4 Anschlüsse mithilfe eines Schrittmotorentreibers angesteuert werden.

3.1.2 Ansteuerung

Die Steuerung des Staubeintrages beruht auf einen Arduino Mega 2560, hierbei handelt es sich um einen leistungsstarken µC. Die Grundidee ist, dass man über eine Benutzeroberfläche die Förderrate des Staubeintrages einstellen kann. Der Arduino errechnet aus den eingegebenen Daten die erforderliche Drehzahl und gibt ein Rechtecksignal mit der dazugehörigen Frequenz aus, welches an einen Schrittmotortreiber angeschlossen wird.

Dieser schaltet die zwei Spulen des Schrittmotors so, dass sich der Schrittmotor mit der gewünschten Drehzahl dreht.

Wie bereits erwähnt lag eine Software und dazugehörige Hardware zu Beginn der Arbeit vor. Die Hardware musste aufgrund der Optimierungsmaßnahmen ausgetauscht werden. Folgende Mankos der vorliegenden Hardware wurden verbessert:

- Regelung der Förderrate
- Rührer mit unterschiedlichen Drehzahlen zu verwenden
- Echtzeitmessung der eingerieselten Staubmenge

3.1.3 Wägezelle

Um die Regelung der einzelnen Rührer zu realisieren, ist es notwendig die Massen der einzelnen Rührer zu kennen, beziehungsweise zumindest deren Änderung. Dabei sollten folgende Anforderungen erfüllt werden:

- Der Messwert der Wägezelle muss mit einem Arduino Mega 2560 auslesbar sein, da die Software für die Regelung von dem Arduino ausgeführt wird.
- Die Mindestmessgenauigkeit soll mindestens ein Gramm betragen.
- Das System soll keine Trägheit aufweisen, bei einer Massenänderung soll ohne Verzögerung der neue Wert auslesbar sein.
- Preiswert, da das System im Flammenrohr verbaut wird kommt es in Kontakt mit der Flammenfront und muss daher von Zeit zu Zeit gewechselt werden.

Eine Dehnmessstreifen-Microwägezelle erfüllt alle der oben genannten Anforderungen. Die Entscheidung fiel auf die Wägezelle CZL635-2kg von TinkerForge. In Tabelle 4 ist dargestellt wie diese Wägezelle zu verschalten ist.

Tabelle 4: Anschlüsse der Wägezelle

Farbe des Kabels	Bestimmung
Rot	Hierbei handelt es sich um den positiven Anschluss der Versorgungsspannung. Die Versorgungspannung muss fünf Volt betragen.
Schwarz	Hierbei handelt es sich um den negativen Anschluss der Versorgungsspannung.
Grün	Der grüne Anschluss ist der positivere Pol des Messwertsignals.
Weiß	Der weiße Anschluss ist der negativere Pol des Messwertsignals.

Eine Seite der Wägezelle wurde mithilfe zweier Schraubverbindungen an das Gerüst für den Staubeintrag fixiert, auf der anderen Seite wurde eine Halterung für die Rührer montiert. Die Halterung für die Rührer wurden aus PLA im 3D- Druckverfahren hergestellt. In Abbildung 16 ist der Staubeintrag in einer Skizzenform ersichtlich.

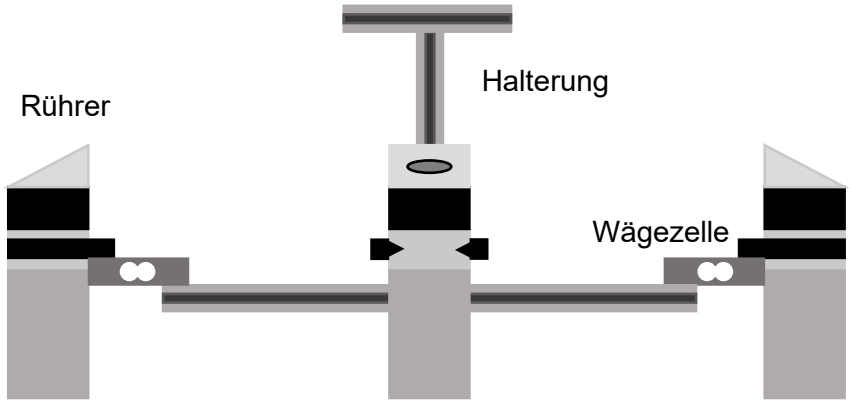


Abbildung 16: Skizze der Halterung der Rührer inklusive der Wägezellen

Um die Daten später digital in den Arduino einzulesen, werden die Messsignale der Wägezellen mittels Analog-Digital Wandler in ein digitales Signal gewandelt. Um selbst kleine Änderung des Signales richtig zu erfassen, wurde ein 16 Bit Analog Digital Wandler

verwendet. Formel 1 zeigt wieviel Volt einem Bit bei der gewählten Auflösung entspricht. Es wurde das Breakout-Board der Sparkfun verwendet. Dieses Board ist so konstruiert, dass die Versorgungsspannungen des ADC und der Wägezelle getrennt sind. Die Versorgung für den ADC verfügt zudem über einen Filter, welcher aus einem 0.1 uF Kondensator und einer 3 uH Spule zusammengesetzt ist.

$$1\ Bit = \frac{5\ Volt}{2^{16}}$$

Formel 1: Ermittlung wieviel Volt ein Bit entspricht

In Abbildung 17 ist dargestellt wie die Wägezellen jeweils an den Analog-Digital-Wandler angeschlossen sind und wie die Analog-Digital-Wandler an den Arduino Mega angeschlossen sind. Der Arduino liest nicht nur die einzelnen Messwerte ein, sondern versorgt auch die ADC und die Wägezellen mit 5 Volt.

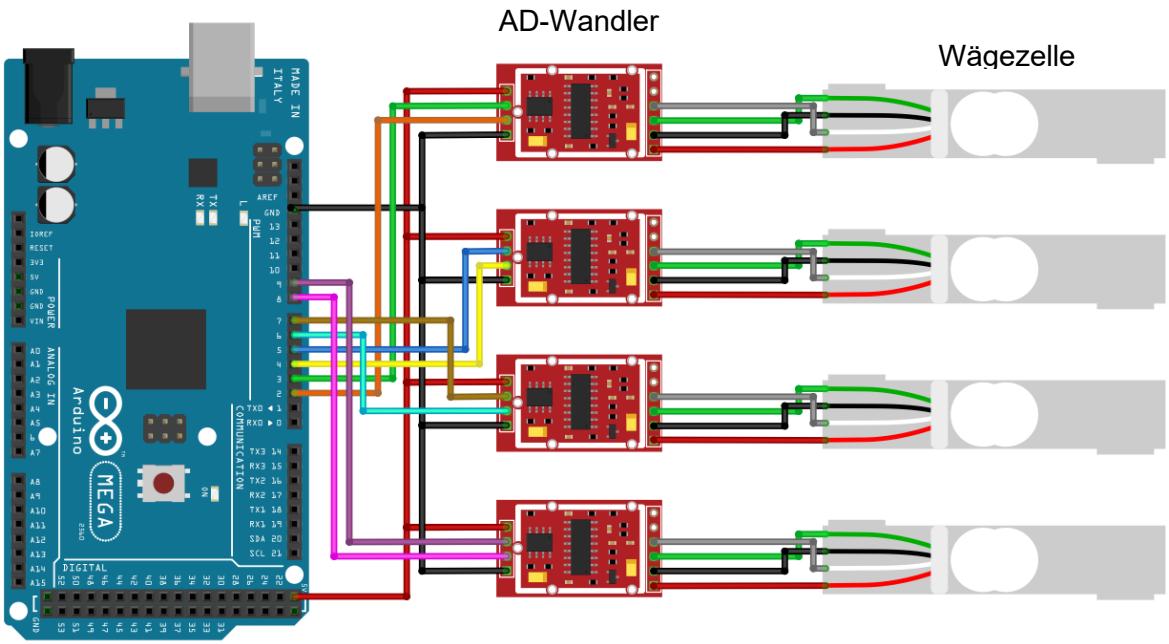


Abbildung 17:Verkabelung der Wägzellen und der Analog- Digital Wandler

In Tabelle 5 ist wiedergegeben wie die einzelnen Wägezellen mit den dazugehörigen ADC-Modulen verbunden sind. Um die ADC vor der Flammenfront zu schützen, wurden zwei Gehäuse aus PLA mittels des 3D- Druckverfahrens erstellt. In jedem der Gehäuse befinden sich jeweils zwei ADCs.

Tabelle 5: Verkabelung der DMS- Wägezellen mit dem ADC-Board

Kabel Wägezelle	Pin HX 711 Board
Rot	E+
Schwarz	E-
Weiß	A-
Grün	A+

3.1.4 Firmware

Um den Arduino in LabVIEW leicht anzusteuern zu können, wurde das Toolkit LINX im LabVIEW Makerhub verwendet. Das Toolkit umfasst eine Vielzahl von Operationen, allerdings ist für das Einlesen der Werte ein sogenannter Custom Command notwendig. LINX stellt eine einfache Möglichkeit zur Verfügung, um Custom Commands zu verwenden.

Um die Werte des ADC in LabVIEW verwenden zu können muss ein Custom Command in der Firmware des Arduino hinzugefügt werden. Wichtig ist, dass diese Firmware nach jedem Wechsel des Arduinos oder nach jeden Reset neu auf den Arduino hochgeladen werden muss. Die Firmware wurde in dem Programm „Arduino IDE“ erstellt. Im Folgenden wird auf die wichtigsten Abschnitte des Programms eingegangen.

```
// Library inclusion, pin declaration and initialisation:
#include <Q2HX711.h>
const byte hx711_data_pin = 3; // HX711 data line connected to arduino D3
const byte hx711_clock_pin = 2; // HX711 clock line connected to arduino D2
const byte hx711_data_pin2 = 5; // HX711 #2 data line connected to arduino D5
const byte hx711_clock_pin2 = 4; // HX711 #1 clock line connected to arduino D4
const byte hx711_data_pin3 = 7; // HX711 #2 data line connected to arduino D5
const byte hx711_clock_pin3 = 6; // HX711 #1 clock line connected to arduino D4
const byte hx711_data_pin4 = 9; // HX711 #2 data line connected to arduino D5
const byte hx711_clock_pin4 = 8; // HX711 #1 clock line connected to arduino D4

Q2HX711 hx711(hx711_data_pin, hx711_clock_pin);
Q2HX711 hx7112(hx711_data_pin2, hx711_clock_pin2);
Q2HX711 hx7113(hx711_data_pin3, hx711_clock_pin3);
Q2HX711 hx7114(hx711_data_pin4, hx711_clock_pin4);
```

Abbildung 18: Einbindung der Libary und Pin Deklaration

In Abbildung 18 ist die Einbindung der Bibliothek, Q2HX711.h, des ADC sowie die Deklaration der einzelnen Pins ersichtlich. Jeder ADC verfügt über eine Clock und Data

Leitung, welche an den Arduino angeschlossen werden müssen. In Tabelle 6 ist aufgelistet wie die einzelnen Data und Clock Leitungen der einzelnen ADCs mit den Arduino Mega angeschlossen sind.

Tabelle 6: Übersicht der Verbindungen zwischen der ADCs und den Arduino

Nr. ADC Wandler	Leitung	Pin Arduino
1	Clock	D 2
1	Data	D 3
2	Clock	D 4
2	Data	D 5
3	Clock	D 6
3	Data	D 7
4	Clock	D 8
4	Data	D 9

Aus den jeweiligem zwei Pins des dazugehörigen ADC werden im Programm vier Objekte des Typs Q2HX711 erzeugt.

```
void loop()
{
  //Listen For New Packets From LabVIEW
  LinxSerialConnection.CheckForCommands();

  //Your Code Here, But It will Slow Down The Connection With LabVIEW

  LinxSerialConnection.AttachCustomCommand(0x9, readout_HX11);
  // Custom commands should have number 0-15

  // here, custom command nr 9 was chosen ==> = hex '9'

}
```

Abbildung 19: Einfügen des erstellten Unterprogrammes

Abbildung 19 zeigt wie der selbst erstellte Code zum Auslesen der ADC Werte in die Firmware hinzugefügt wird. Dies geschieht in der Loop Funktion, je mehr Funktionen bzw. umso aufwendiger diese werden, desto mehr die Verbindung zwischen LabVIEW und des Arduinos beeinflusst. Zudem muss dem Custom Command eine hexadezimale Zahl hinzugefügt über welche später in LabVIEW auf die Funktion zugegriffen werden kann.

```
// -------  Custom command definition    -------
int readout_HX11(unsigned char numInputBytes, unsigned char* input,
unsigned char* numResponseBytes, unsigned char* response){

signed int rawsignal = int (float(hx711.read()/100.0));
signed int rawsignal2 = int (float(hx7112.read()/100.0));
signed int rawsignal3 = int (float(hx7113.read()/100.0));
signed int rawsignal4 = int (float(hx7114.read()/100.0));

unsigned int value =rawsignal;      // Make the value an unsigned integer, to shift 0 in
                                    // from the left, instead of ones.
unsigned int value2 =rawsignal2;    .
unsigned int value3 =rawsignal3;
unsigned int value4 =rawsignal4;

*numResponseBytes = 8;
response[0] = (value & 0xFF00) >> 8; //mask top bits and bottom bits, shift 8
response[1] = (value & 0x00FF) ; //mask top bits, no need to shift.
response[2] = (value2 & 0xFF00) >> 8; //mask top bits and bottom bits, shift 8
response[3] = (value2 & 0x00FF) ; //mask top bits, no need to shift.
response[4] = (value3 & 0xFF00) >> 8; //mask top bits and bottom bits, shift 8
response[5] = (value3 & 0x00FF) ; //mask top bits, no need to shift.
response[6] = (value4 & 0xFF00) >> 8; //mask top bits and bottom bits, shift 8
response[7] = (value4 & 0x00FF) ; //mask top bits, no need to shift.
return 0;
};
```

Abbildung 20: Code der erstellten Custom Command

Wie in Abbildung 20 ersichtlich ist muss jede Custom Command einen Integer-Wert zurückgeben. Eine Null bedeutet dabei, dass kein Fehler aufgetreten ist. Jede andere Zahl bedeutet, dass ein Fehler aufgetreten ist. In Tabelle 7 sind die vier zwingend notwendigen Parameter genauer beschrieben.

Tabelle 7: Grundparameter einer Custom Command

Parameter	Beschreibung
numInputBytes	Anzahl der Bytes des Eingangarrays(von LabVIEW).
input	Hierbei handelt es sich um ein Array des Types U8, welches die Daten enthält, die von LabVIEW an die Funktion übermittelt werden. Daten die übermittelt werden können sind z.B. Integer-Werte.
numResponseBytes	Muss auf die Anzahl von Bytes gesetzt werden, die man an LabVIEW zurückschicken möchte.
response	Hierbei handelt es sich um die Daten, die man an LabVIEW zurückschicken möchte, es handelt sich um ein Array des Typs U8.

Bei der erstellten Custom Command werden keine Daten von LabVIEW an die Funktion übermittelt, sprich es werden nur Daten an LabVIEW übermittelt. Da es sich um einen 16 Bit Analog-Digital-Wandler handelt, müssen pro Wandler zwei Bytes geschickt werden. Zuerst werden die Float-Werte der Wägezellen in Integer Werte konvertiert. Anschließend werden die Daten auf je zwei Bytes aufgesplittet.

3.1.5 Zusätzliche Motortreiberplatine

Wie schon erwähnt, war die bereits vorhandene Hardware nicht dafür geeignet, um die Rührer jeweils mit individueller Drehzahl zu betreiben. Die entwickelte Treiberplatine für die Schrittmotoren besitzt zwar für jeden Rührer einen individuellen Eingang mit welchem die Drehzahl der einzelnen Rührer vorgeben werden kann. Allerdings ist die Platine so ausgelegt, dass man ein Rechtecksignal mit entsprechender Frequenz anlegen muss. Möchte man die Rührer mit verschiedener Drehzahl betreiben, muss man an den Eingängen der Treiberplatine Rechtecksignale unterschiedlicher Frequenz anlegen. Leider kann der gewählte Arduino zwar an mehreren Ausgängen ein Rechtecksignal ausgeben, allerdings nur mit gleicher Frequenz.

Eine variable Ansteuerung der einzelnen Rührer ist zwingend notwendig, da die Rührer bei gleicher Drehzahl eine unterschiedliche Förderrate aufweisen und des Weiteren die Förderrate zusätzlich von der Füllhöhe der Rührer abhängt. Sprich je niedriger die Füllhöhe ist, desto schneller muss sich der Rührer drehen, um eine zeitlich konstante Förderrate zu gewährleisten.

Um die variable Ansteuerung erfolgreich umzusetzen wurde die bestehende Hardware erweitert. Der Arduino kann zwar nur Rechtecksignale mit gleicher Frequenz generieren, jedoch kann er mehrere PWM Signale mit unterschiedlichen Werten generieren. Es wurde eine zusätzliche Platine angefertigt, auf der vier Arduino Nano verbaut sind. Jeder dieser Arduinos wandelt ein PWM Signal in ein Rechtecksignal mit entsprechender Frequenz um. Der Haupt-Arduino generiert auf vier verschieden Channel jeweils verschiedene PWM-Signale. An allen vier der Arduino Nanos wird eines dieser Signale an einen Analogeneingang angeschlossen. Der Nano liest nun den PWM-Wert ein und multipliziert diesen mit einer Konstante. Das Ergebnis entspricht der Frequenz des Rechtecksignals welches anschließend von dem Arduino Nano generiert wird und an einen digitalen Ausgang angelegt wird. Der entsprechende Ausgang ist mit dem entsprechenden Eingang der Treiberplatine verbunden.

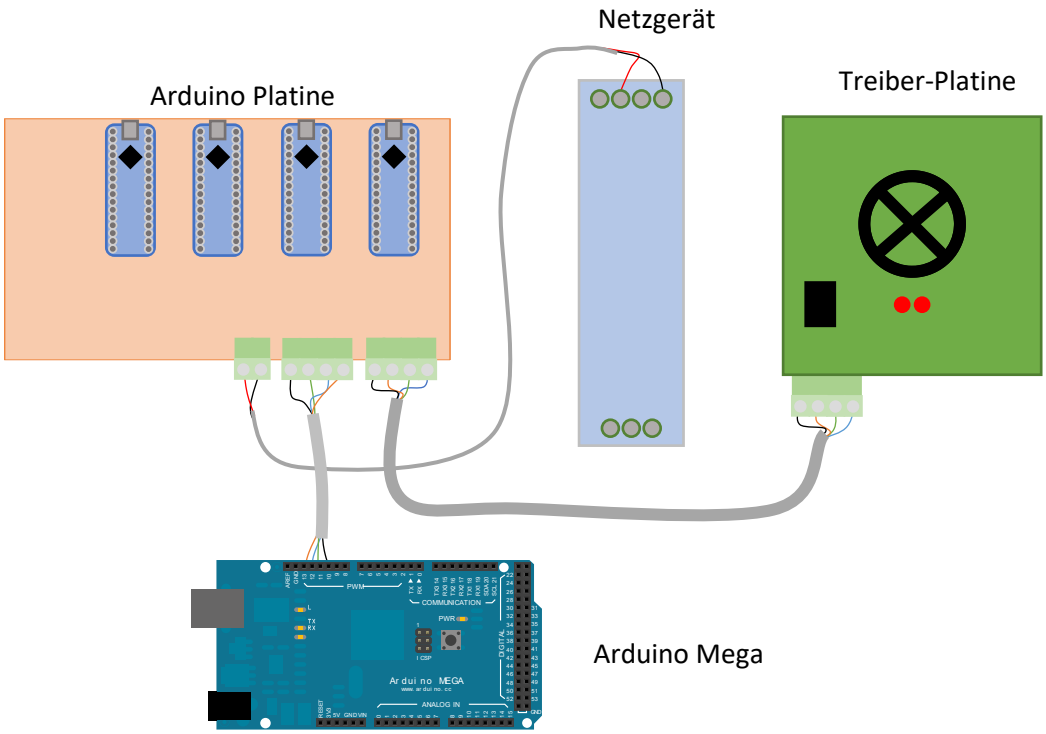


Abbildung 21:Verschaltung der zusätzlichen Motorentreiberplatine mit der restlichen Hardware

Wie die einzelnen Komponenten genau miteinander verschalten sind, kann in Tabelle 8 nachgelesen werden. Zusätzlich werden alle Eingänge über 10 kΩ Widerstände und alle Ausgänge über 47 kΩ Widerstände auf Masse gezogen. Die Arduinos werden über eines Pins mittels eines Netzgerätes, welches in dem Schaltkasten verbaut ist, mit Spannung versorgt. Wichtig ist, dass die Massen des Arduino Megas und der Arduinos Nanos miteinander verbunden sind, um sicherzustellen, dass die Arduino Nanos das eingelesene PWM-Signal auf das richtige Spannungsniveau bezieht. Abbildung 21 zeigt wie die neue Platine mit der restlichen Hardware verbunden ist. In der Skizze wird nur dargestellt wie die Platine mit der restlichen Hardware verbaut ist, es wird nicht dargestellt wie die restliche Hardware miteinander verbunden ist.

Tabelle 8: Verschaltung des Hauptarduinos mit den Nebenarduinos und der Treiberplatine

Pin Arduino Mega	Nummer Arduino Nano	Pin Arduino Nano	Pin Arduino Nano	Pin Treiberplatine
10	1	A0	A1	S1
11	2	A0	A1	S2
12	3	A0	A1	S3
13	4	A0	A1	S4

3.1.6 Software für die zusätzliche Motortreiberplatine

Bei der Software für die verwendeten Arduino Nanos handelt es sich um ein kurzes Programm, welches mit der Software Arduino IDE erstellt wurde. Die Software hat nur die Aufgabe das einkommende PWM-Signal einzulesen und daraus ein entsprechendes Rechtecksignal zu erzeugen.

In Abbildung 22 ist das kurze Programm dargestellt. Es werden nur zwei Pins verwendet, einer zum Einlesen des PWM-Signals mit derFunktion „pulseIn“. Diese Funktion gibt an wieviel Millisekunden zwischen zwei positiven Flanken vergangen sind. Da bei einem PWM Signal mit dem Wert keine Flanken vorhanden sind kann die Funktion keine Zeit angeben und gibt nach einer gewissen Zeit Null aus. Daher darf das PWM Signal maximal 0,99 betragen. Zwar treten bei einem PWM-Signal von Null ebenfalls keine Flanken auf, jedoch gibt die Funktion nach einer gewissen Timeoutzeit Null aus, daher ist ein PWM Signal von Null zulässig. Die gemessenen Millisekunden werden anschließend mit einem Faktor multipliziert. Das Ergebnis ist die Frequenz des Rechtecksignals welches mittels der Funktion „tone“ generiert wird.

```
int analogPin = A0;   // Input PWM
int analogPin2 = A1;  // Output Square Wave
float val = 0;
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  Serial.begin(9600);
  pinMode(analogPin2, OUTPUT);  // set pin as Output
  pinMode(analogPin, INPUT);    // set pin as Intput
}

// the loop function runs over and over again forever
void loop() {
  val = pulseIn(analogPin,HIGH);  // read millisecond between to HIGHS
  Serial.println(val);            // debug value
  tone(analogPin2, val*10);       // Generate Square wave
}
```

Abbildung 22: Code der das PWM Signal in Square Wave Signal konvertiert.

Das Programm wurde auf den vier Nanos hochgeladen und dort automatisch ausgeführt, wenn die Arduinos mit Spannung versorgt sind. Der Faktor, mit dem die Millisekunden zwischen den Flanken multipliziert wird ist von der minimalen Drehzahl, mit der die gewünschte Förderrate erreicht wird, nach unten begrenzt. Nach oben ist der Faktor dadurch begrenzt mit welcher maximalen Frequenz der Schrittmotor die Spulen schalten kann.

3.2 Software

Das Ziel dieser Arbeit war die Erstellung der Software für den Versuchsaufbau. Wie bereits erwähnt, wurde als Entwicklungsumgebung LabVIEW verwendet.

Das Herzstück der Steuerung stellt ein Arduino Mega2560 dar, welcher den ganzen Versuchsablauf steuert. Der Arduino kommuniziert über LabVIEW mit einem angeschlossen PC, dafür ist es notwendig das NI-Zusatzpaket „Digilent LINX“ zu installieren. Dieses Toolkit ermöglicht das Erstellen einer Schnittstelle zu gängigen Embedded- Plattformen wie Arduino und chipKit. Diligent Linx kann über den sogenannten Package Manager von National Instruments installiert werden. Nach der erfolgreichen Installation des Zusatzpaketes kann die notwendige Firmware über LabVIEW auf den µC wie folgt installiert werden. In LabVIEW öffnet man den Pfad Tools→MakerHub→LINX→Linx Firmware Wizard, es öffnet sich ein Fenster in welchem man den gegebenen Arduino auswählt. Falls der Arduino gewechselt wird, muss dieser Schritt erneut durchgeführt werden.

Das Programm kann in drei Phasen unterteilt werden, es gibt eine Startphase, eine wiederholende Phase und eine Endphase. In Abbildung 23 ist die Benutzeroberfläche kurz nach dem Start des Programmes dargestellt.

Abbildung 23: Benutzeroberfläche des Programmes kurz nach dem Start

3.2.1 Anforderungen an die Software

Wie bereits in Kapitel 1.1 erwähnt wurde, gab es bereits vor dieser Arbeit eine Labor-Versuchsanlage am TPT. Im Folgenden wird kurz darauf eingegangen wie dort ein Versuch durchgeführt wird, da die dort gesammelten Erfahrungen zu Anforderungen für die zu entwickelten Software führten.

Nachdem das Programm gestartet ist, kann mittels eines Buttons der Staubeintrag gestartet werden. Ein Drehpotentiometer ermöglicht die Drehzahlregelung einer Schnecke zur Staubförderung, welcher mittels eines Gleichstrommotor angetrieben wird. Das manuelle Betätigen eines Buttons löst die Zündung aus. Es wird nicht erfasst mit welcher Drehzahl, Förderrate und wie lange Staub eingetragen wird.

Daraus ergeben sich folgende Anforderungen für die Software:

Automatische Zündung: Die Zündung muss wahlweise automatisch erfolgen, wenn das jeweilige Zündkriterium erfüllt ist. Das Zündkriterium kann die Dauer des Staubeintrages oder die Menge des eingetragenen Staubes sein.

Automatisierter Staubeintrag: Der Staubeintrag sollte per Button gestartet werden. Ist das jeweilige Abbruchkriterium erfüllt, stoppt der Staubeintrag automatisch. Wichtig ist, dass noch eine definierte Zeit vor und nach der Zündung Staub gefördert werden kann und das Ende des Staubeintrages nicht an den Zündzeitpunkt geknüpft ist. Wählbare Abbruchskriterien sind z.B.: Zeit die Staubeintrag fördern soll oder Masse an Staub die eingerieselt werden soll.

Regelung der Förderrate: Die gewünschte Förderrate sollte mittels des User-Interfaces eingegeben werden. Die tatsächliche Förderrate pro Rührer soll ermittelt werden und die Förderrate mithilfe eines PID-Reglers autonom angepasst werden.

Einfache Bedienbarkeit: Das Programm muss nach einer kurzen Einschulung für jeden bedienbar sein. Werte wie z.B: die gewünschte Förderrate sollten automatisch von dem letzten Versuch übernommen werden, zudem sollten die Daten von Versuchen gespeichert werden können, um diese später einfach in das Programm laden zu können. Buttons, die nur zu einem bestimmten Zeitpunkt gedrückt werden können, sollen auch nur dann angezeigt werden.

Erweiterbarkeit: Die Software sollte in einer Struktur erstellt werden, dass Sie später relativ leicht von einer dritten Person erweitert, beziehungsweise abgeändert werden kann. Dafür muss das Programm auch übersichtlich programmiert werden.

Übersichtliches User Interface: Das Interface der Software sollte sehr übersichtlich gestaltet sein und alle relevanten Daten sollten gut aufbereitet dargestellt werden. So sollte in etwa die eingebrachte Staubmasse und die aktuelle Förderrate in einem Diagramm dargestellt werden.

Speicherung der Messdaten: Alle Messdaten sollten automatisch in einer Form gespeichert werden, welche die Auswertung einfach gestaltet. Die Daten sollen ab dem Zündzeitpunkt automatisch aufgezeichnet werden. Zudem sollten Daten zum Staubeintrag für jeden Versuch gespeichert werden.

Die oben genannten Punkte stellen die wichtigsten Anforderungen an die Software dar. Weitere Anforderungen an das Programm sind unter anderem:

- Integrierte Möglichkeit, um jede Wägezelle kalibrieren zu können.
- Checkliste am Beginn des Programmes, ob alle Kabel richtig angeschlossen sind
- Zündfunkendauer beliebig wählbar gestalten

- Alarmsignale per Signalleuchte und Lautsprecher zu generieren

3.2.2 Struktur des Programmes

Das Programm ist zu umfangreich, um den kompletten Code wiederzugeben. In diesem und in folgenden Kapiteln wird der Code übersichtsmäßig beschrieben und gewählte Abschnitte des Codes werden detaillierter beschrieben.

Das Programm besteht aus mehreren Unterprogrammen, die entweder vom Hauptprogramm oder von anderen Unterprogrammen aufgerufen werden. Im Folgenden werden sämtliche SubVis aufgelistet und kurz ihre Aufgabe beschrieben bevor später auf jedes einzelne Unterprogramm eingegangen wird. Dies dient dazu, einen Gesamteindruck des Codes zu vermitteln bevor dieser im Detail beschrieben wird.

Hauptprogramm – main.vi

Wie der Name schon erwarten lässt, ist dieses Programm das Fundament der Software. Es ist auch jenes Programm, welches als einziges vom User selbst gestartet wird. Es ist zudem jener Teil des Programmes, welches den Großteil des User- Interfaces darstellt. Im Wesentlichen besteht es aus einer Start-, End- und einer Schleifenphase. Das Hauptprogramm ruft je nach Bedarf die nötigen Unterprogramme auf.

Unterprogramme

- **auto_entry.vi:** Diese VI ist nur von Relevanz, falls der Staubeintrag automatisch stoppen soll. Diese VI überprüft welches Kriterium für den Staubeintragstopp ausgewählt ist. Ist dieses erfüllt dann gibt das Programm den Wert TRUE zurück.
- **auto_ignition.vi** Ist nur von Bedeutung, falls eine automatische Zündung erfolgen soll, je nach Zündkriterium wird die Zündung aktiviert. Das Programm kann die Zündung starten und beenden
- **calib.vi** Dieses Programm dient dazu eine Wägezelle neu zu kalibrieren. Dazu können eine unbestimmte Anzahl von Gewichten verwendet werden. Das Programm errechnet daraus die Daten der Wägezelle und speichert diese in ein Textdokument.
- **data_Read.vi** Dieses Unterprogramm kommt an verschiedensten Stellen zum Einsatz. Das Programm liest die Daten aus den entsprechenden Textdateien und setzt die einzelnen Werte des Programms auf die Werte aus den

Dateien. Das Programm wird in etwa nach der Kalibration oder in der Initialisierungsphase aufgerufen.

- **finish_auto_ignition.vi** Dieses Programm kommt zum Einsatz, wenn der Staubeintrag bereits abgeschlossen ist, die Zündung jedoch noch nicht gestartet oder beendet wurde. Je nach Fall startet die Zündung und/oder wartet eine bestimmte Zeit und deaktiviert die Zündung.
- **finish_measurment.vi** Nachdem der Staubeintrag und der Zündvorgang abgeschlossen sind, wartet dieses Programm eine gewisse Zeit und pausiert das Aufzeichnen der Daten. Das Programm dient dazu nur den zeitlich relevanten Zeitraum aufzuzeichnen.
- **init_progressbar.vi** Die Benutzeroberfläche besitzt zwei Fortschrittsbalken für den Staubeintrag und die Zündung, falls diese automatisch durchgeführt werden. Dieses Programm dient zu deren Initialisierung wie in etwa die richtige Skalierung.
- **measurment.vi** Diese VI besitzt für den User ein sichtbares Frontpanel. Das Programm stellt die Daten der Thermoelemente und der LWL in einem Diagramm dar und speichert die Daten in eine Messdatendatei. Die VI wird nicht von dem Hauptprogramm aufgerufen, sondern läuft parallel dazu. Der einzige Zusammenhang der beiden VIs ist, dass sie auf dieselbe globale Variable zugreifen.
- **prestart_check.vi** Bevor die Verbindung mit dem Arduino aufgebaut wird, wird eine kurze Checkliste angezeigt. Dies dient dazu, sicherzustellen, dass alle notwendigen Verbindungen für die erfolgreiche Durchführung eines Versuches vorhanden sind. Nur falls alle Punkte bestätigt sind, erscheint der Button zum Fortfahren.
- **read_4_loadcells.vi** Auf diese VI wird über die erstellte Custom Command zugegriffen, welche in die Firmware des Arduino geschrieben wurde. Das Programm erstellt aus den jeweiligen zwei Bytes die Werte der einzelnen ADCs.
- **set_Global.vi** Diese VI wird unmittelbar vor Beginn der Staubförderung aufgerufen und setzt alle Zündungs- und Staubeintragsvariablen auf den entsprechenden Wert. Dazu zählen unter anderem der Zeitpunkt des Startes der Förderung und welches Zündkriterium ausgewählt ist.
- **settings.vi** Hierbei handelt es sich um ein sehr umfangreiches Unterprogramm. Mit Hilfe des Programmes wird festgelegt, ob automatisch gezündet oder Staub gefördert wird. Mit der Funktion lässt sich die Förderdauer oder auch Fördermenge des Einrieselungsvorganges einstellen. Zudem können Zündfunkendauer und sämtliche Parameter für die PID Regelung eingestellt werden.

Alle eingestellten Daten können gespeichert bzw. gespeicherte Daten geladen werden.

- **smooth_singals.vi** Die Signale der einzelnen Wägezellen haben ein leichtes Rauschen übergelagert. Diese VI dient dazu die Signale zu glätten, um dafür zu sorgen, dass der PID-Regler nicht auf das Rauschen reagiert.
- **unbundle.vi** Die meisten Felder der Benutzeroberfläche sind aus Übersichtlichkeitsgründen zu Clustern zusammengefügt worden. Leider ist das Zugreifen auf Daten eines Clusters mit relativ viel Code verbunden. Da öfters auf die Daten zugegriffen wird, wurde diese VI erstellt welche die Cluster aufschlüsselt und die einzelnen Daten in einer leicht verwertbaren Form ausgibt.

Globale Variablen – Global.vi

Globale Variablen sind eine eigene Komponente in LabVIEW, dabei handelt es sich um spezielle VIs die kein Blockdiagramm haben. Dank der globalen Variablen können Daten zwischen zwei oder mehreren gleichzeitig ausgeführten VIs ausgetauscht werden. Da das Hauptprogramm und die Measurment VI gleichzeitig ausgeführt werden und das Hauptprogramm vorgibt wann Daten aufgezeichnet werden, ist die Verwendung von globalen Variablen notwendig. Zudem wurden auch mehrere Variablen als global gestaltet, obwohl dies nicht nötig wäre, aber die Übersichtlichkeit des Codes fördert.

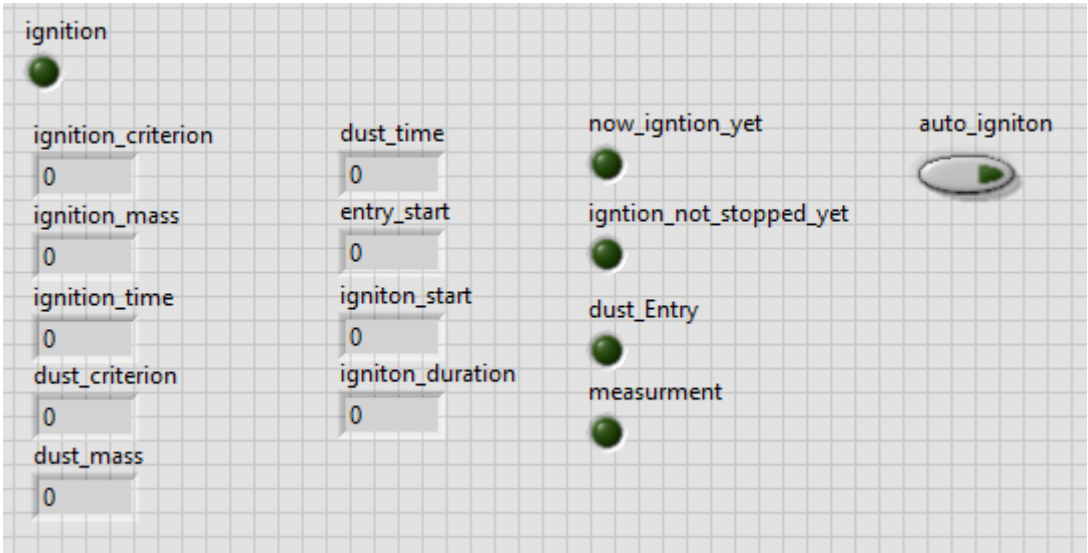


Abbildung 24: Frontpanel der VI der globalen Variablen

In Abbildung 24 ist das Frontpanel der VI dargestellt in welcher sich sämtliche globalen Variablen befinden. In Tabelle 9 werden sämtliche Variablen kurz beschrieben.

Tabelle 9: Auflistung der verwendeten globalen Variablen

Name	Typ	Beschreibung
auto_ignition	Boolean	TRUE: automatische Zündung aktiviert. FALSE automatische Zündung deaktiviert.
dust_criterion	Integer	Kann drei Werte annehmen: Null bedeutet der Staubeintrag endet nach einer gewissen Zeit, Eins der Staubeintrag endet nach einer gewissen Fördermenge und Zwei bedeutet der Staubeintrag wird manuell gestoppt.
dust_Entry	Boolean	TRUE: Staubeintrag ist aktiviert. FALSE: Staubeintrag ist deaktiviert.
dust_mass	Double	Gibt an wieviel Gramm Staub in Summe eingerieselt werden soll.
dust_time	Double	Gibt an wie viele Sekunden Staub eingetragen werden soll.
entry_start	Double	Speichert den Wert, welcher der Timer am Beginn des Eintrages besitzt.
ignition	Boolean	TRUE: Zündung ist aktiv. FALSE: Zündung ist nicht aktiv.
ignition_criterion	Integer	Null bedeutet, dass die Zündung starten soll nachdem eine gewisse Masse an Staub eingetragen wurde und Eins bedeutet, dass die Zündung nach einer gewissen Zeit starten soll.
ignition_duration	Double	Stellt die Dauer des Zündfunkens in Millisekunden dar.
ignition_mass	Double	Masse, die eingerieselt werden soll, bevor die Zündung startet.
ignition_not_stopped_yet	Boolean	TRUE: Die Zündung wurde noch nicht gestoppt. FALSE: Die Zündung wurde bereits gestoppt.

ignition_start	Double	Speichert den Wert, welcher der Timer besitzt, wenn die Zündung gestartet wird.
ignition_time	Double	Gibt an wie viele Sekunden eingerieselt werden soll, bevor die Zündung startet.
measurment	Boolean	TRUE: Das Programm speichert Messdaten. FALSE: Das Programm speichert keine Messdaten.
no_ignition_yet	Boolean	TRUE: Die Zündung wurde noch nicht gestartet FALSE: Die Zündung wurde bereits gestartet

3.2.3 Unterprogramme

3.2.3.1 auto_entry

Diese VI kontrolliert ständig, ob weiter Staub eingetragen werden soll oder nicht. Diese VI verfügt über einen Ein- und Ausgang. Der Eingang emtspricht einem Doublewert, der die geförderte Staubmenge in Gramm angibt. Bei dem Ausgang handelt es sich um einen boolschen Wert. Solange weiter Staub gefördert werden soll, gibt das Programm den Wert FALSE zurück, ist das Abbruchskriterium erfüllt gibt das Programm den Wert TRUE zurück.

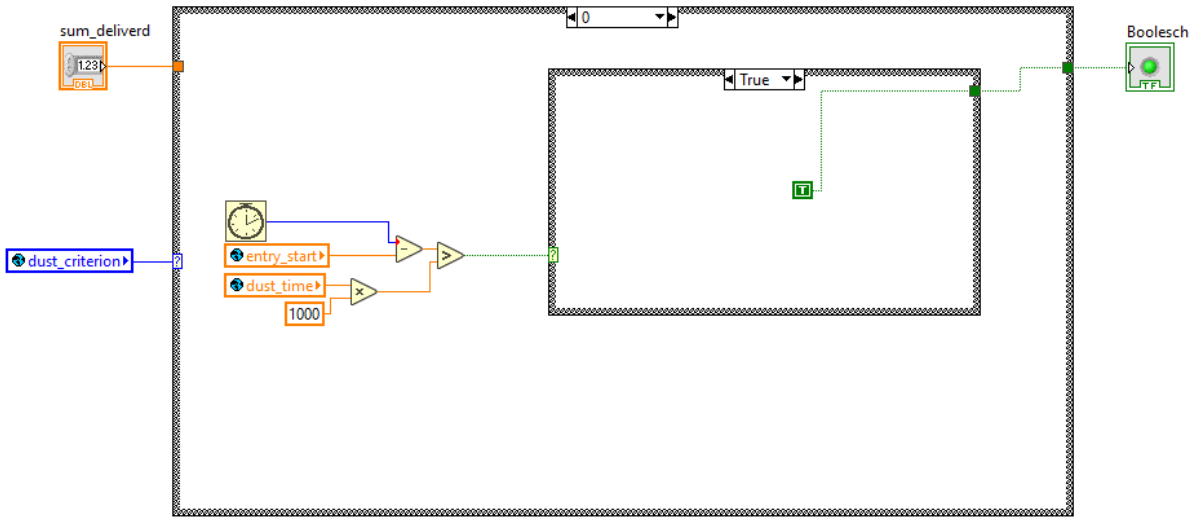


Abbildung 25: Teil des Blockdiagramm der auto_Entry VI

Die VI führt je nach gewählten Abbruchskriterium drei verschiedene Codes aus.

- Der Staub wird laut Abbruchskriterium eine gewisse Zeit eingerieselt und sollte nach dieser Zeit automatisch gestoppt werden. Für diesen Fall ist der Code in Abbildung 25 dargestellt. Wie zu erkennen ist, wird über die globale Variabel „dust_criterion“ entschieden welcher Teil des Codes ausgeführt werden soll. In diesem Fall wird der aktuelle Wert des Timers mit dem jetzigen Wert verglichen und kontrolliert ob die Differenz davon größer ist wie die gewünschte Einrieseldauer.
- Es soll eine gewisse Menge an Staub gefördert werden, wenn diese erreicht ist, stoppt die Förderung automatisch.
- Der Staub wird solange gefördert bis der Anwender den Staubeintrag manuell beendet, in diesem Fall gibt die VI immer FALSE zurück

3.2.3.2 Auto_ignition

Diese VI dient dazu stets zu kontrollieren, ob die Zündung aktiviert oder auch deaktiviert werden soll, während der Zeit, in der der Staubeintrag aktiviert ist. In dieser VI wird zwischen einer Vielzahl von Fällen unterschieden. Zudem berechnet die VI stets die Zeit bis zur Zündung aus. Beträgt diese Zeit unter vier Sekunden werden die Arduino Ausgänge so geschalten, dass der Alarmton und das rote Licht des Alarmsignals aktiviert werden. Ist der Zeitpunkt der Zündung erreicht wird der Ardunio Ausgang für die Zündung auf High geschalten. Der Zeitpunkt für die Zündung ist, wenn das Zündkriterium erreicht ist, welches wieder eine gewisse Zeit oder gewisse geförderte Staubmenge sein kann. Ist die Zündung bereits aktiviert worden, kontrolliert die VI, ob die Zündung wieder auf ausgeschalten werden soll, falls ja wird der Pin der Zündung auf Low gesetzt.

Die VI verfügt über die folgenden drei Eingänge:

- LINX Resource über diese kann auf den Arduino zugegriffen werden.
- setpoint hierbei handelt es sich um einen Doublewert, welcher die gewünschte Förderrate darstellt.
- Dum_deliverd ist ebenfalls eine Variable des Typs Double und ist die Summe der geförderten Massen.

Des Weiteren hat die VI einen Ausgang. Dabei handelt es sich um die LINX Resource.

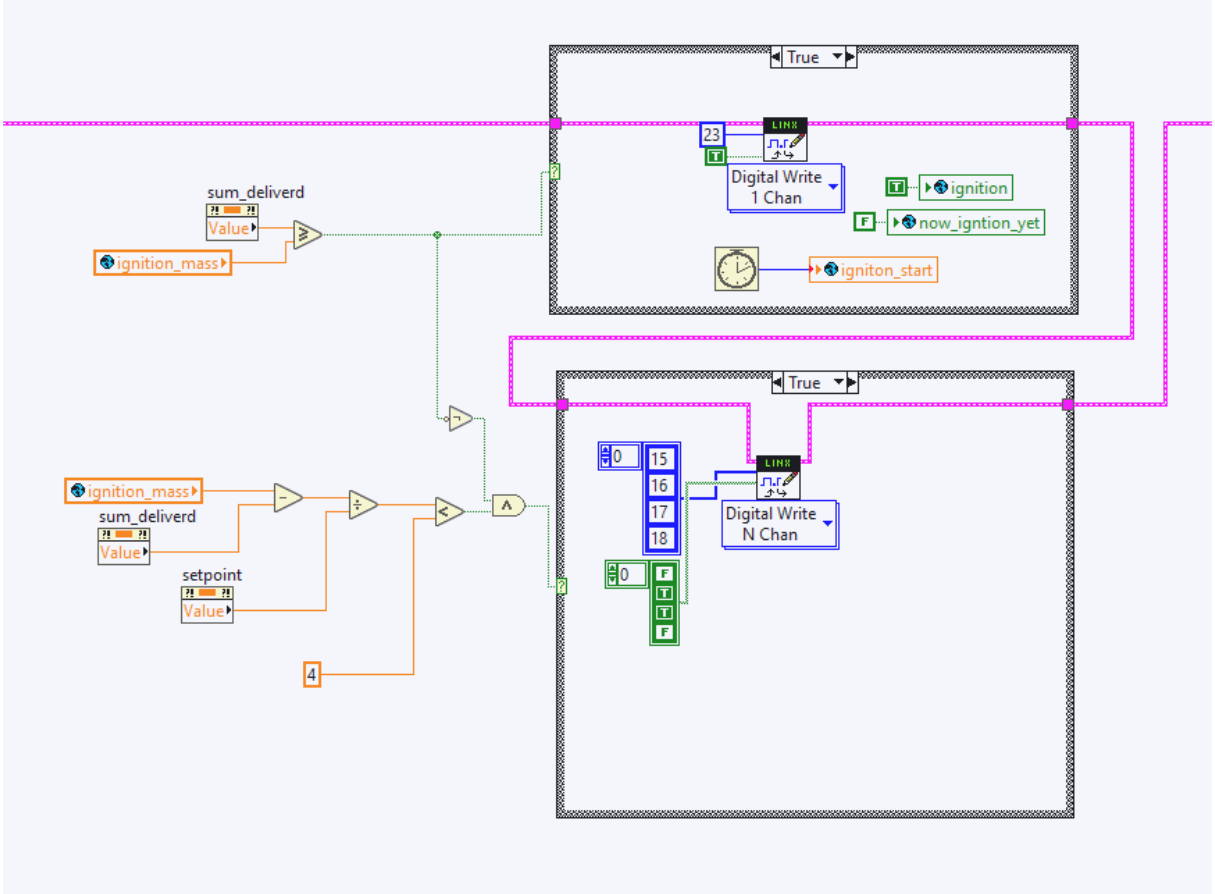


Abbildung 26: Teil des Blockdiagrammes der auto_Ignition VI

Abbildung 26 zeigt jenen Teil des Codes, welcher ausgeführt wird, falls die automatische Zündung aktiviert ist, keine Zündung stattgefunden hat und die Zündung nach einer gewissen eingebrachten Masse erfolgen soll. Im oberen Teil der Darstellung sieht man wie die geförderten Masse mit der Sollmasse verglichen wird, falls genug geliefert wurde wird der Pin 23 auf High gesetzt. Im unteren Teil der Darstellung sieht man wie die Zeit bis zum Zündungsstart errechnet wird, falls die Zeit unter vier Sekunden beträgt werden die Pins der Alarmleuchte entsprechend gesetzt.

3.2.3.3 Calib

Diese VI dient dazu eine Wägezelle zu kalibrieren. Für die Kalibration können sind bis zu zehn Referenzmessungen möglich. Jedoch müssen zumindest zwei Messungen mit unterschiedlichen Massen durchgeführt werden, da ansonsten keine lineare Regression möglich ist. Die VI wird aufgerufen, falls der User den Button „Calibration“ drückt.

Die VI besitzt über folgende vier Eingänge: Einen Pfad zu der Textdatei in welche die neuen Kalibrationsdaten gespeichert werden sollen. Einen Error-Cluster welcher Informationen zu

Fehlern enthält, die vor Start der VI aufgetreten sind, dieser sollte in der Regel leer sein. Eine LINX Resource über die auf den Arduino zugegriffen werden kann. Einen nummerischen Anschluss, der angibt an welche Wägezelle kalibriert werden soll.

Zudem besitzt die VI über folgende vier Ausgänge: Eine LINX Resource mit der auf den Arduino zugegriffen werden kann. Den Error-Cluster welcher Informationen zu Fehlern enthält, die bei der Durchführung dieser VI oder zuvor aufgetreten sind. Slope stellt die Steigung der kalibrierten Wägezelle dar. Intercept stellt den Ordinatenabschnitt der kalibrierten Wägezelle dar.

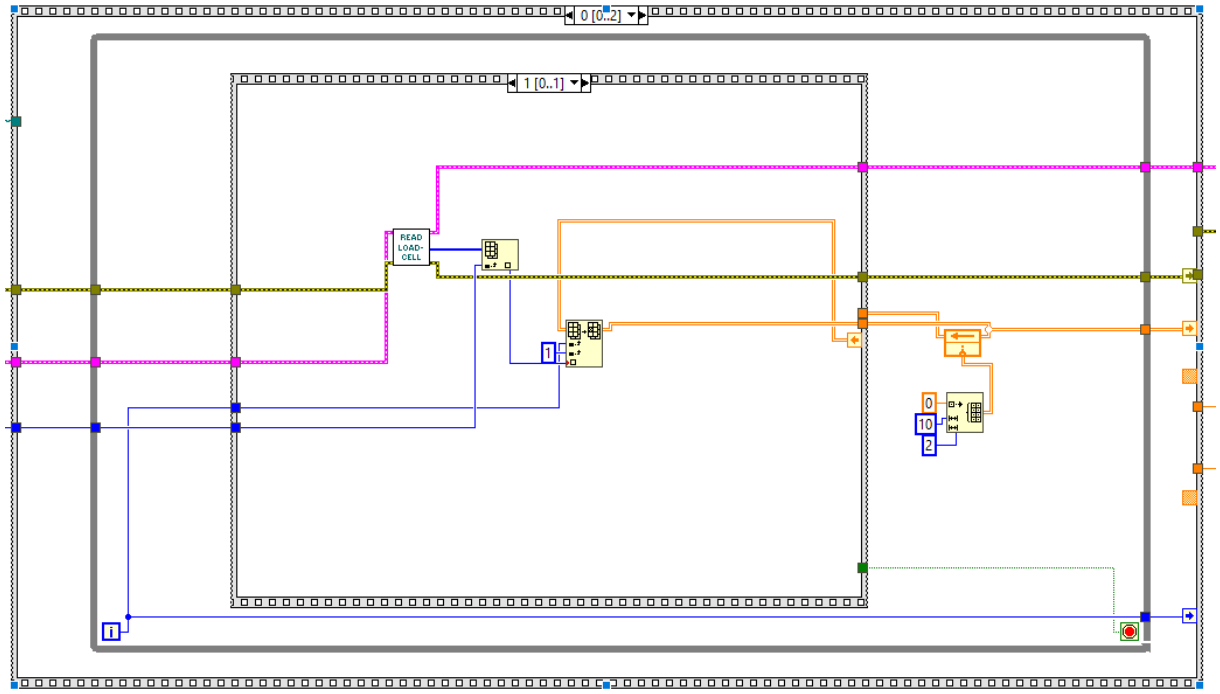


Abbildung 27: Code der die Sensordaten in das Array speichert.

Die VI besteht aus einer dreiteiligen gestaffelten Sequenz. In der ersten Sequenz wird über ein Pop Up Fenster nach der jeweiligen Masse gefragt, welche sich gerade auf der Wägezelle befindet. Klick man auf „Next“ wird der Wert des Sensors und die dazugehörige Masse in ein Array angehängt, und das Pop Up Fenster öffnet sich erneut. Dies geschieht solange bis der Nutzer auf „Last“ klickt. In diesem Fall wird das jeweilige Datenpaar in das Array geschrieben. In der nächsten Sequenz wird aus den Daten des Arrays ausgewertet und daraus die Steigung und der Ordinatenabschnitt berechnet. Dies geschieht über eine

lineare Regression. In der letzten Sequenz werden schließlich die neuen Kalibrationsdaten in die Textdatei geschrieben.

Abbildung 27 zeigt den Code welcher den Sensorwert mithilfe einer anderen VI einliest und dann den Wert im Array ändert. Es ist ersichtlich, dass beim ersten Durchlauf ein 2x10 Array erstellt wird, welches mit Nullen gefüllt ist.

3.2.3.4 data_Read

Mit Hilfe dieser VI werden eine Vielzahl von Parameter aus verschiedenen Textdateien ausgelesen und entsprechend aufgearbeitet, sodass sie leicht in das Frontpanel des Hauptprogrammes geschrieben werden können. Zu den Daten, die ausgelesen werden, zählen unter anderem die Kalibrationswerte sämtlicher Wägezellen, die PID Wert und diverse Daten für den Staubeintrag.

Die VI besitzt über fünf Eingänge: Der Eingang „Data“, dient dazu, die Struktur des Clusters des Hauptprogrammes zu übernehmen. Zudem werden an drei Eingängen die Pfade zu den relevanten Textdateien übermittelt. Beim letzten Eingang handelt es sich um den Fehler Cluster.

Zudem besitzt die VI über folgende vier Ausgänge: Error enthält Informationen zu Fehlern, die bei der Durchführung dieser VI oder zuvor aufgetreten sind. Der zweite Ausgang, Data, das fertig erstellt Cluster welches direkt in das Front Panel des Hauptprogrammes geschrieben werden kann

Die VI besteht aus mehreren Sequenzen, die nacheinander abgearbeitet werden. In den einzelnen Sequenzen werden jeweils sämtliche Daten einer Datei ausgelesen.

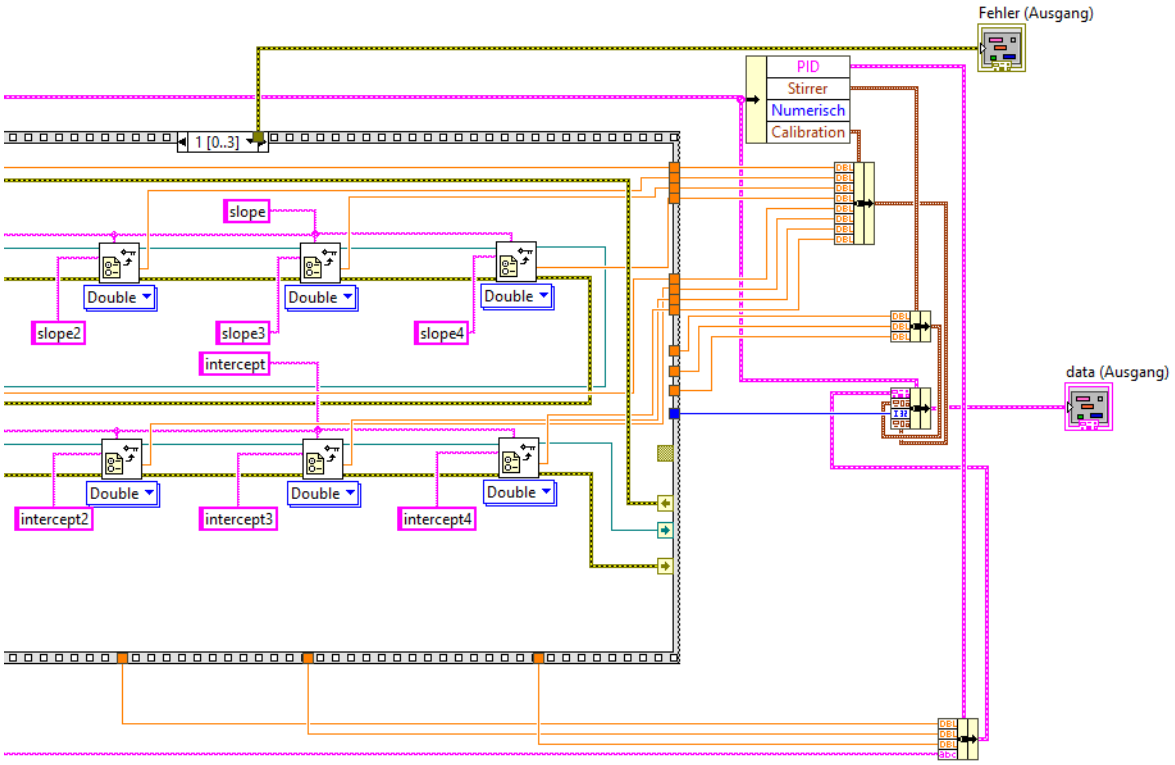


Abbildung 28: Darstellung der Zusammenführung sämtlicher Daten zu einem Cluster

Abbildung 28 deutet an wie alle Daten aus den Textdateien ausgelesen werden. Am rechten Ende der Abbildung ist dargestellt wie alle Dateien zu Clustern zusammengefügt werden. Das finale Cluster besteht unter anderem aus anderen Clustern.

3.2.3.5 finish_auto_ignition

Die VI ist für den Abschluss der automatischen Zündung zuständig. Sie kommt zum Einsatz, falls die Zündung noch nicht abgeschlossen oder gestartet ist, der Staubeintrag jedoch schon gestoppt ist und die automatische Zündung aktiviert ist.

Das Programm besitzt drei Eingänge: die Linx Ressource, den Fehler Cluster und ein Zeiger zu den Zündungsfortschrittsbalken. Zudem besitzt die VI über zwei Ausgänge: Linx Ressource und den Fehler Cluster.

Die VI kontrolliert zuerst, ob die automatische Zündung aktiviert wurde, falls nicht wird die VI umgehend geschlossen. Danach wird kontrolliert, ob die Zündung bereits gestartet wurde, falls ja wartet das Programm solange bis die Zündung die gewünschte Zeit aktiv war und schaltet die Zündung anschließend ab. Ist die Zündung jedoch noch nicht aktiviert worden,

wartet das Programm solange bis das Zündungskriterium erreicht ist und schaltet anschließend die Zündung für die gewünschte Zeit ein.

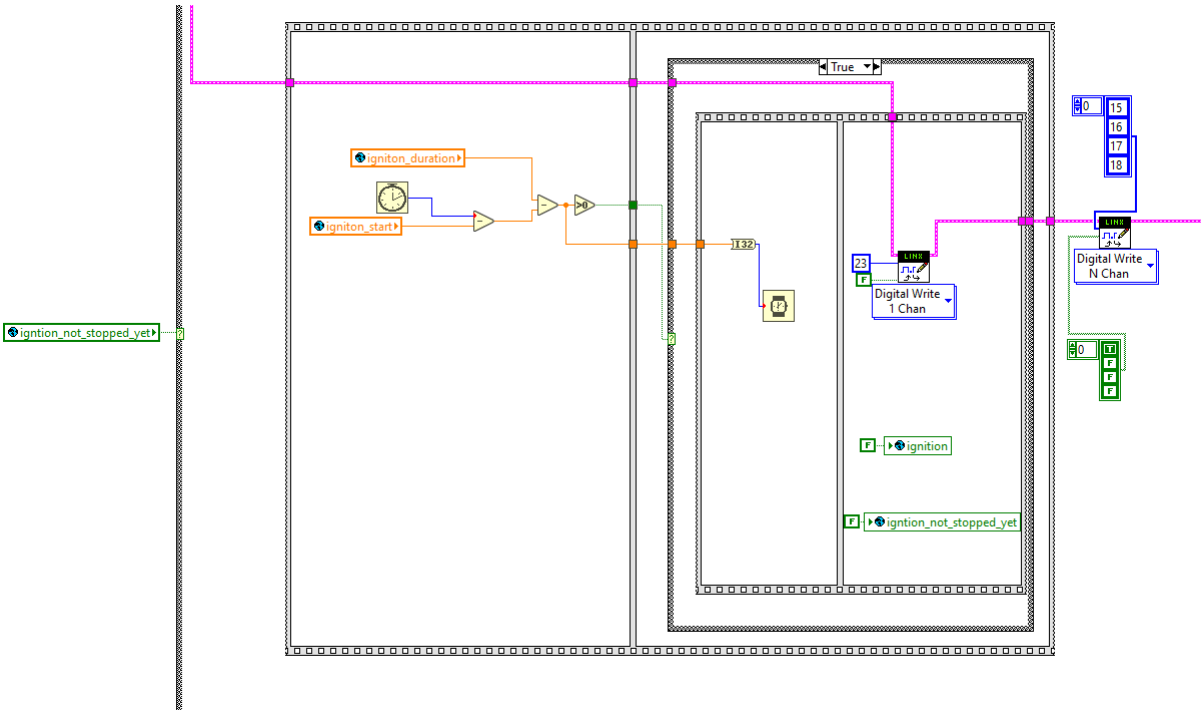


Abbildung 29:Programm Code für das Beenden der Zündung falls die Zündung bereits gestartet wurde und der Staubeintrag schon beendet wurde

In Abbildung 29 ist der Code für den Fall, dass die automatische Zündung aktiviert ist, die Zündung bereits während des Staubeintrages gestartet wurde jedoch nicht während des Staubeintrages beendet wurde. Im ersten Schritt wird die Differenz zwischen der gewünschten Zünddauer und der bereits verstrichenen Zeit seit dem Zündstart berechnet. Anschließend wird genau diese Zeit gewartet, bevor die Zündung abgeschaltet wird. Schließlich wird die Warnleuchte noch entsprechen geschalten, um die aktive Zündung zu signalisieren.

3.2.3.6 init_progressbar

Diese VI dient dazu, um die Fortschrittsbalken für die Zündung und Staubeintrag richtig zu skalieren. Dazu verfügt die VI über vier Eingänge, drei davon sind Zeiger, die auf die Fortschrittsbalken zeigen. Der vierte Eingang ist die gewählte Förderrate, diese wird benötigt, falls man ausrechnen möchte wie lange es dauert bis eine gewisse Masse eingetragen wird. Die VI besitzt keinen Ausgang.

Je nach gewähltem Zündungskriterium und Abbruchkriterium für den Staubeintrag werden die Fortschrittsbalken dementsprechend skaliert. Dies geschieht indem der Maximalwert sowie der Minimumwert der Fortschrittsbalken definiert wird.

3.2.3.7 Measurment

Wie schon erwähnt ist das Frontpanel dieser VI für den Nutzer sichtbar. Diese VI wird automatisch geöffnet. Die VI wartet solange bis die globale Variable „measurment“ auf TRUE gesetzt wird. Anschließend erstellt das Programm zwei neue Messdateien und speichert in diesen Files sämtliche Messwerte der Photodioden und der Thermoelemente. Das Programm schreibt alle Daten bis vier Sekunden nach der Zündung mit und geht anschließend wieder in Warteposition. Die VI besitzt keine Ein- und Ausgänge.

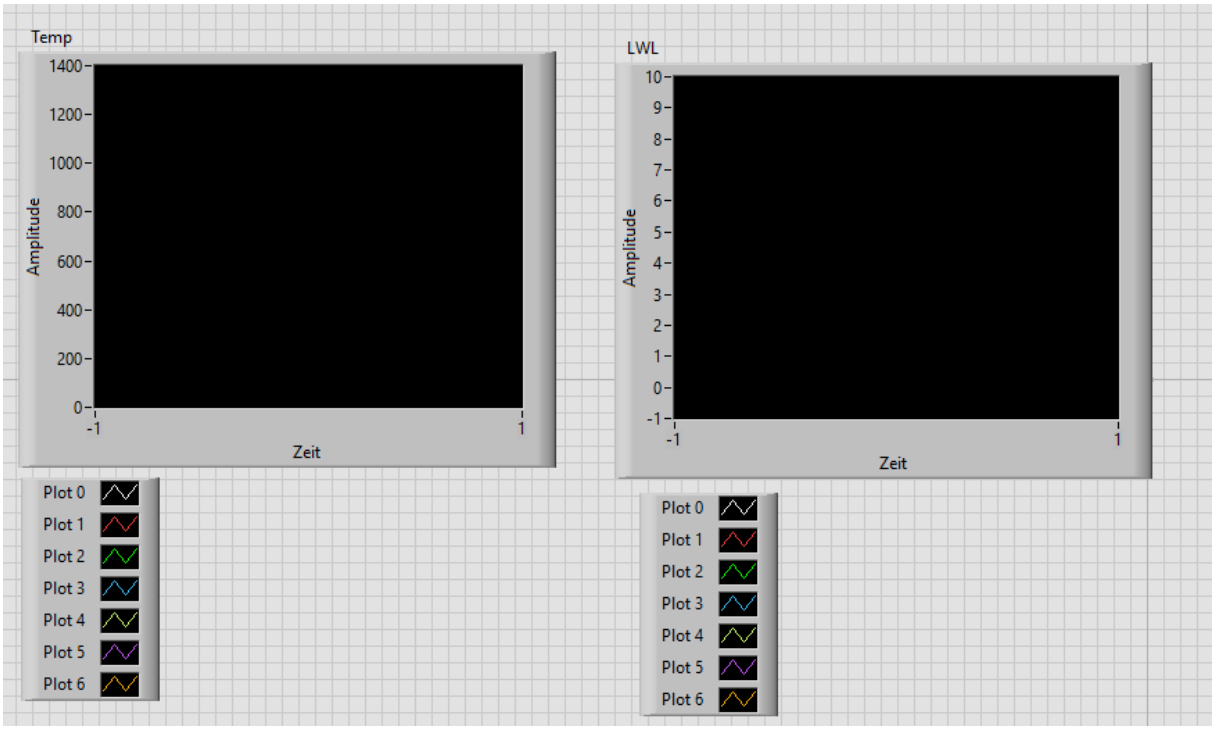


Abbildung 30: Frontpanel der Measurment VI

Abbildung 30 zeigt das Frontpanel der VI, das wie zu erkennen ist die Messsignale in Diagrammen dargestellt. Im linken Diagramm werden sämtliche Temperaturen der Thermoelemente dargestellt und im rechten werden die Werte der Photodioden dargestellt.

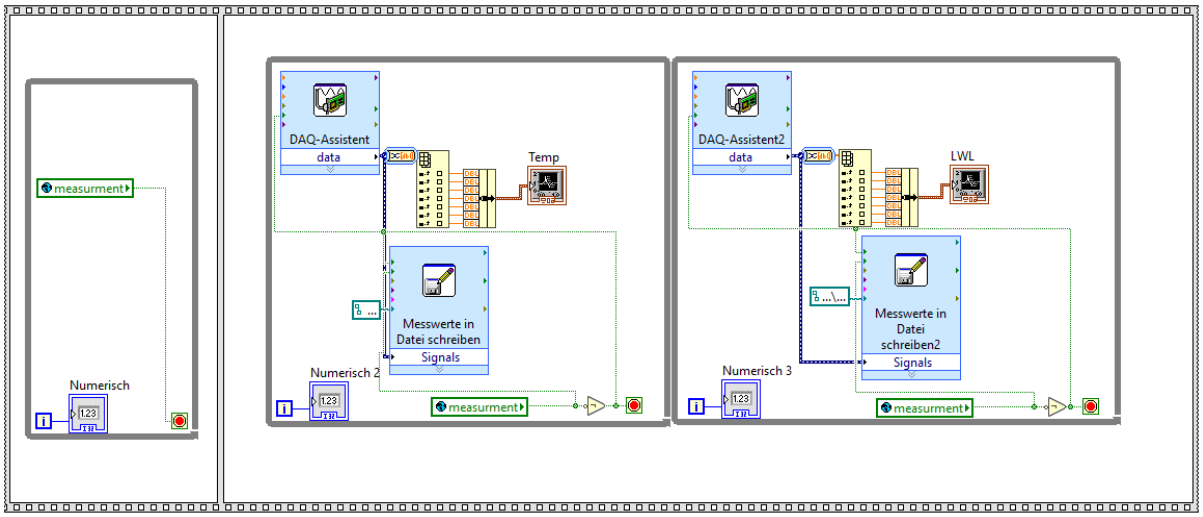


Abbildung 31: Teil des Codes der Measurment VI

Abbildung 31 zeigt den Code der VI, der sich in einer While-Schleife befindet, diese ist in der Abbildung nicht enthalten. Zu erkennen ist, dass das Programm solange wartet bis das Hauptprogramm die globale Variable „measurment“ auf TRUE setzt und anschließend die Daten aus den NI-Messkarten (National Instruments) ausliest und in Messfiles schreibt. Für die Thermoelemente wurde eine NI cDAQ-9171 – Messkarte verwendet. Die Daten werden dabei mit einer Rate von 100 Hertz abgetastet. Die Messkarte (NI USB-6211) für die Photodioden misst mit einer Abtastrate von 1000 Hertz. Bei beiden Messdateien ist die erste Spalte ein Zeitstempel. Die Daten werden jeweils während der Messung in die Dateien geschrieben und jeweils 100 Datensätze gesammelt bevor diese in die Files geschrieben werden.

3.2.3.8 prestart_check

Das Programm kann die Hardware nur dann richtig, ansteuern wenn diese korrekt mit dem Arduino und untereinander verschaltet ist. Da es bei fehlenden Verbindungen zu Fehlern im Programm kommen kann, sollten die Verbindungen vor jedem Start kontrolliert werden. Um sicherzustellen, dass alle Verbindungen angeschlossen sind, wird dies mittels einer Checkliste am Start des Programmes kontrolliert. Der User muss hier sämtliche Punkte ankreuzen, bevor das Programm gestartet werden kann. Zudem wird aus Sicherheitsgründen nachgefragt ob die Zündung ausgeschaltet ist.

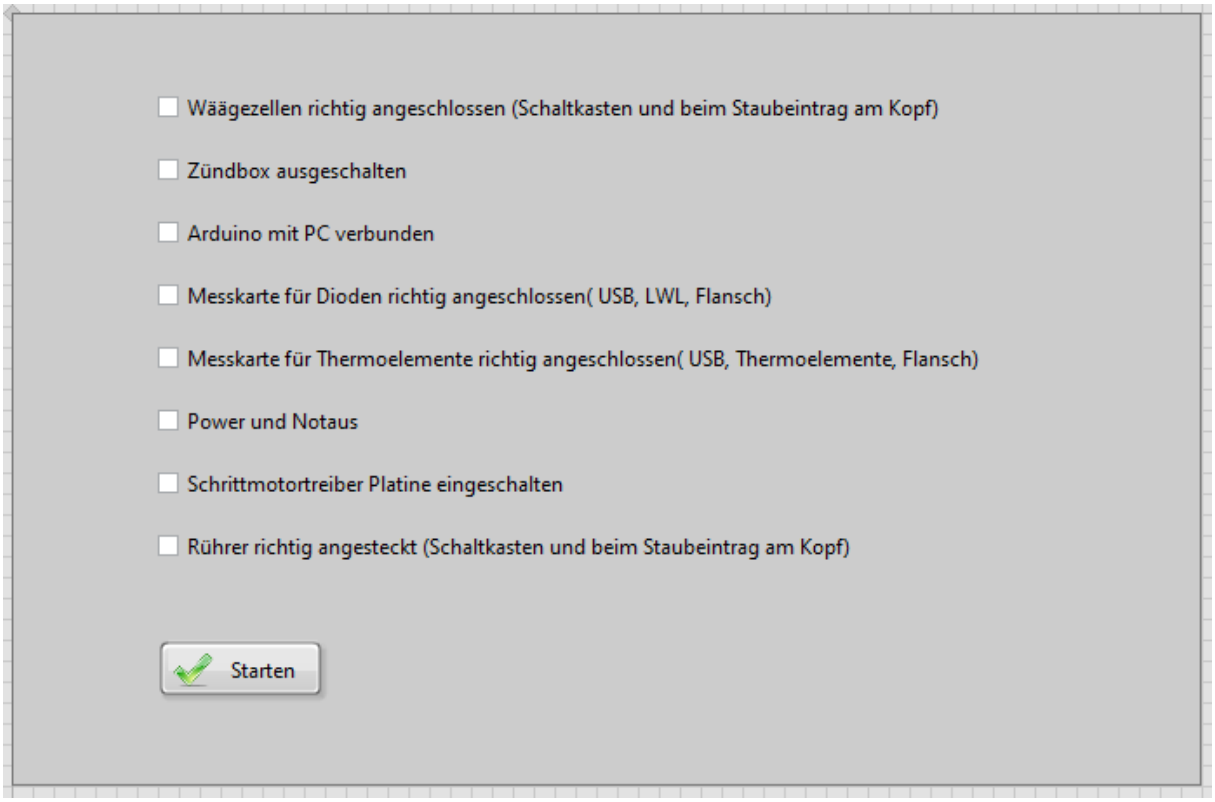


Abbildung 32: Checkliste am Start des Programmes

Abbildung 32 zeigt das Frontpanel der VI. Der Button „Starten“ wird, wenn das Programm läuft, nur angezeigt, falls alle Punkte vom User bestätigt wurden. Besonders wichtig ist der erste Punkt der Liste, da das Programm ständig versucht die Wägezellen auszulesen. Sind die Wägezellen nicht angeschlossen kommt es zu Fehlern im Programm. Die VI besitzt über keine Ein- und Ausgänge. Das Blockdiagramm dieser VI ist sehr einfach gehalten.

3.2.3.9 read_4_loadcells

Wie schon in Kapitel 3.1.4 erwähnt wurde, werden die Daten der ADC-Wandler mittels eines Custom Command in LabVIEW ausgelesen. Da die Werte der ADC-Wandler an mehreren Stellen des Programmes abgefragt werden, wurde eine VI dafür erstellt. Die VI besitzt über zwei Eingänge der LINX Ressource und dem Fehler Cluster. Die VI hat drei Ausgänge den LINX Ressource, den Fehler Cluster und einen Integer-Array. Das Integer-Array besteht aus den vier Werten der ADC-Wandler.

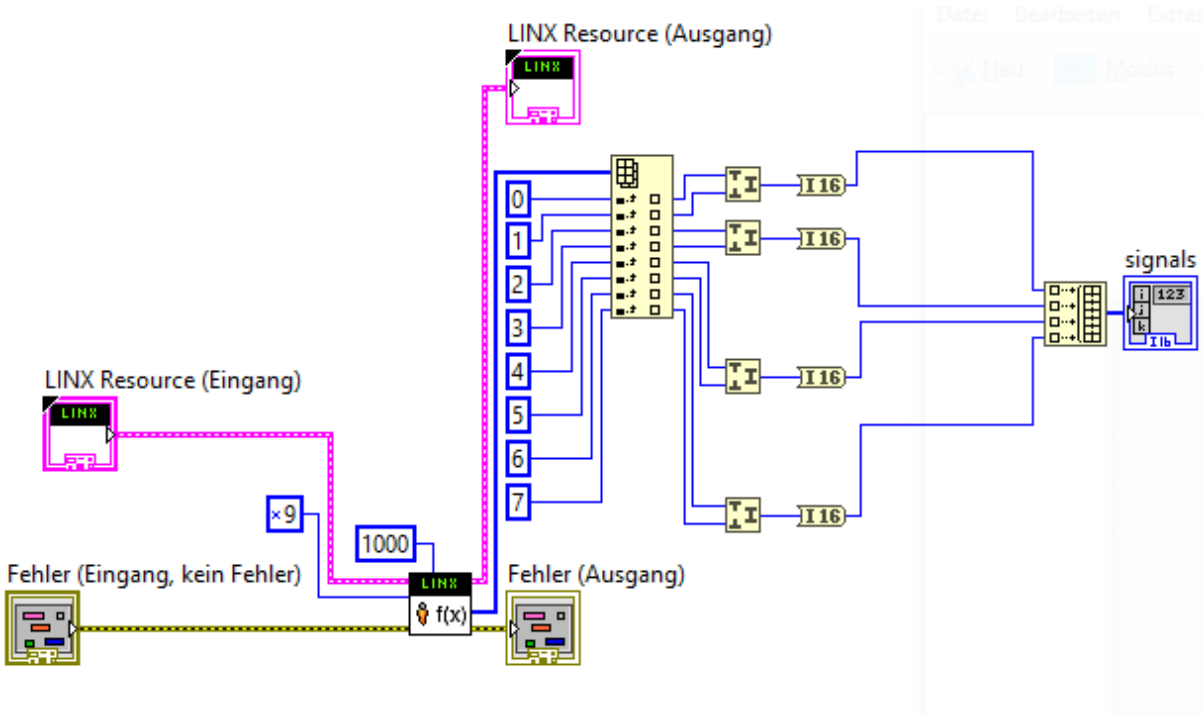


Abbildung 33: Code zum Auslesen der ADC-Wandler

Abbildung 33 zeigt das Blockdiagramm der VI, man sieht wie auf dem Costum Command zugegriffen wird, das Ergebnis ist ein Array bestehend aus 8 Bytes wobei es sich um die Daten der ADC-Wandler handelt. Anschließend werden jeweils zwei Bytes zu einem 16 Bit Integerwert zusammengefasst. Die vier Integerwerte werden zu einem Array zusammenfasst, bei welchen es sich um einen Ausgang handelt. Wie in Kapitel 3.1.4 gezeigt wurde wird im Code über die Hexadezimalzahl 9 auf die erstellte Custom Command zugegriffen.

3.2.3.10 Set_Global

Bevor der Staubeintrag gestartet wird, müssen diverse Variablen, die für den Staubeintrag oder die Zündung relevant sind, gesetzt werden. Zudem müssen einige globale Variablen auf bestimmte Werte gesetzt werden. Um den umfangreichen Code für den Staubeintrag übersichtlicher zu gestalten, wird dies in einer eigenen VI erledigt. Zudem wird der Wert des Timers gespeichert. Dieser Wert stellt den Startwert des Staubeintrages dar. Des Weiteren werden weitere globale Variablen für den Staubeintrag initialisiert. Die VI besitzt zwei Eingänge den Fehler Cluster und den Pfad zur Textdatei, in welcher die zu verwendeten Parameter abgespeichert sind. Bei dem einzigen Ausgang handelt es sich um den Fehler Cluster. Es wird unter anderem ausgelesen ob die automatische Zündung verwendet werden soll und was für ein Abbruchkriterium für den Staubeintrag verwendet werden soll.

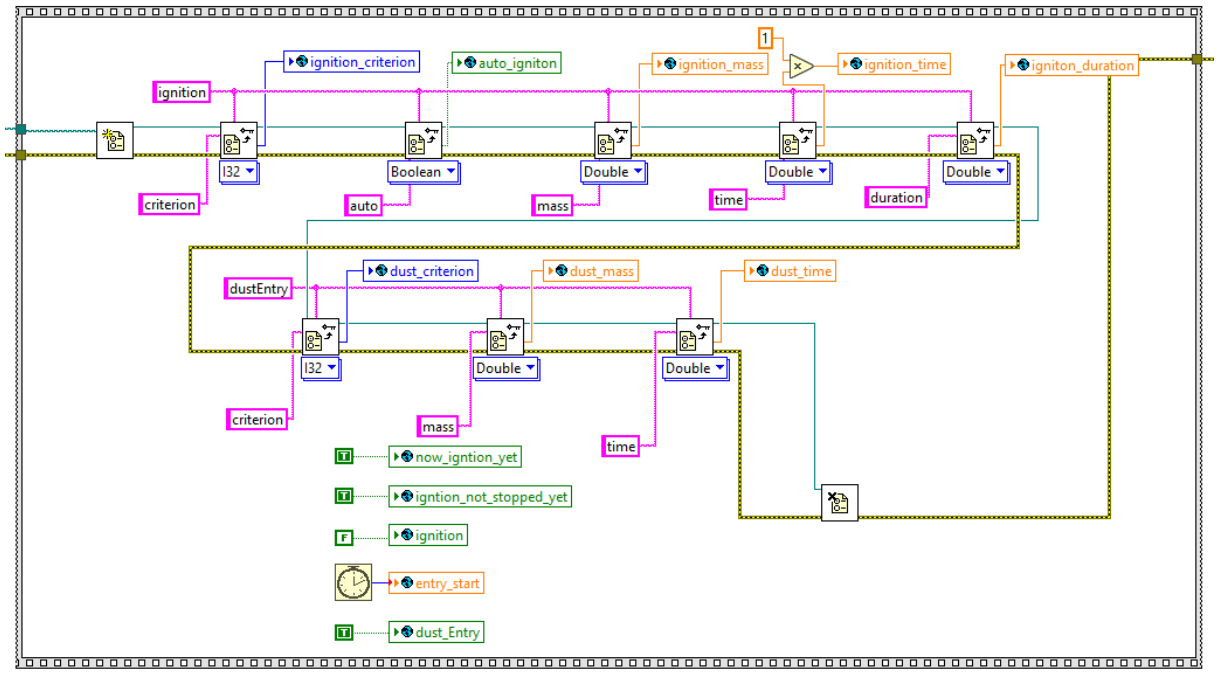


Abbildung 34: Blockdiagramm der set_global VI

In Abbildung 34 ist dargestellt wie Daten aus einer Textdatei ausgelesen werden. Zuerst wird die Datei über den Pfad geöffnet. Anschließend wird über den Sektionsabschnitt und der Variablennamen die Variable ausgelesen. Sind alle Variablen ausgelesen, wird die Datei schließlich geschlossen. Wichtig ist, dass beim Auslesen angegeben wird um, welchen Variablentyp es sich handelt. In der Abbildung 34 ist ebenfalls ersichtlich, wie die einzelnen globalen Variablen am Start des Staubeintrages gesetzt werden.

3.2.3.11 Settings

Hierbei handelt es sich um eine VI bei der das Frontpanel für den User sichtbar ist. Über dieses Frontpanel kann der User neue Daten über den Staubeintrag und Zündung erstellen, bearbeiten und alte Profile laden. Das Frontpanel ist auf zwei verschiedene Tabs aufgesplittet. Der Tab „Ignition“ ist für alle Variablen zuständig die für Versuche von Belang sind, mit welcher Förderrate eingerieselt werden soll, Verwendung des automatischen Staubeintrages und wie lange/wieviel Staub eingetragen werden soll. Je nach ausgewählten Kriterien werden verschieden Eingabefelder ein- und ausgeblendet. Es werden stets nur die Felder angezeigt, die von Relevanz sind. Der Tab „PID“ beinhaltet Parameter, die für die PID Regelung wichtig sind, welche von dem verwendeten Staub abhängen. Für verschiedene Versuchssequenzen mit demselben Staub sollte das gleiche PID- Profil verwendet werden. Einige dieser Parameter können direkt im Hauptprogramm eingeben werden. Die VI füllt die Felder standardmäßig mit den zuletzt gewählten Parametern aus.

Die VI besitzt über vier Eingänge, drei Dateipfade und den Daten Cluster des Hauptprogrammes. Zudem besitzt die VI über drei Ausgänge. Dabei handelt es sich um zwei Dateipfade sowie um einen booleschen Wert, der dazu dient, dass das Hauptprogramm in einen Stand-by-Modus übergeht während die Settings VI ausgeführt wird.

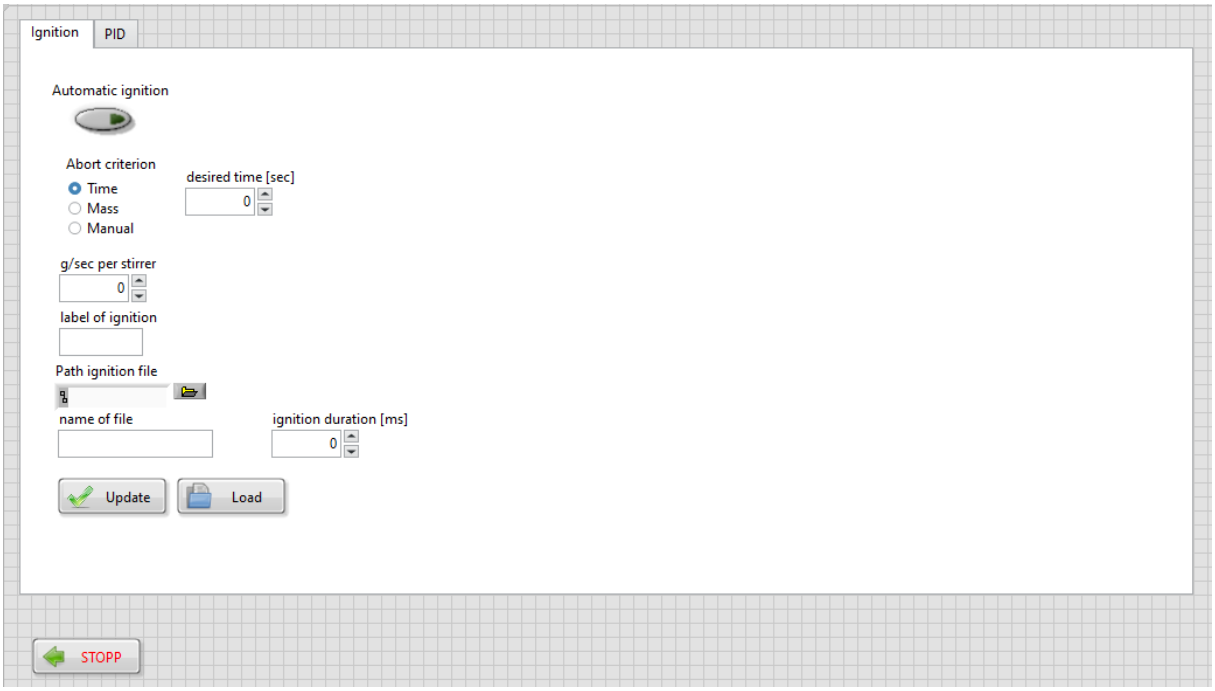


Abbildung 35: Frontpanel der Setting VI

Abbildung 35 zeigt das Frontpanel der VI. In der Abbildung ist die automatische Zündung nicht gewählt worden. Wählt man diese, erscheint die Option nach welchem Kriterium man die Zündung starten möchte. Wird die Zeit ausgewählt, ist es wichtig zu wissen, dass null Sekunden bedeutet, dass die Zündung gleichzeitig mit dem Staubeintrag erfolgen soll. Ein negativer Wert bedeutet, dass die Zündung vor dem Ende des Staubeintrages erfolgen soll.

Das Programm besteht im Grunde aus drei Sequenzen. In der ersten Sequenz werden sämtliche Parameter, die zuletzt gewählt wurden, in die Felder aus Default-Werten geschrieben. Damit wird dem User Arbeit erspart. Bei der nächsten Sequenz handelt es sich um eine Schleife, die solange durchlaufen wird, bis der Nutzer das Programm schließt. Die Schleifensequenz enthält eine Vielzahl von verschiedenen Fällen. Abbildung 36 zeigt einen dieser Fälle. In der letzten Sequenz werden schließlich alle gewünschten Parameter in die entsprechenden Textdateien gespeichert.

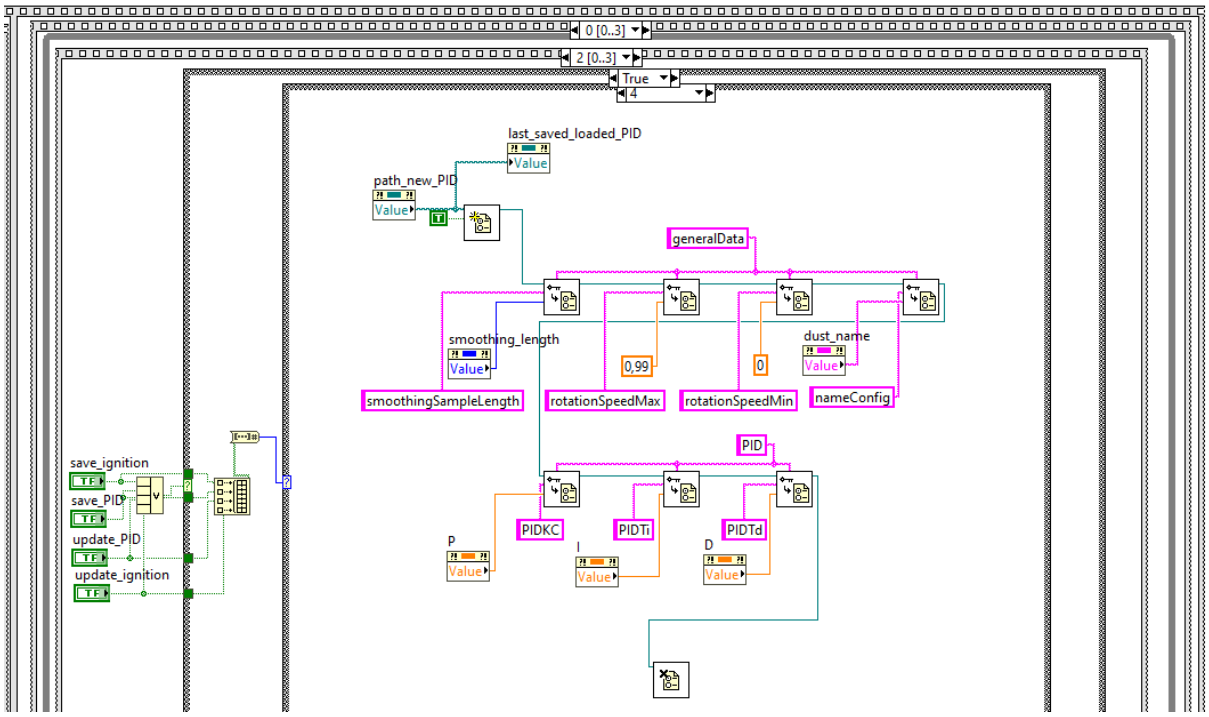


Abbildung 36:Teil des Blockdiagramms der VI

Der Code der in Abbildung 36 dargestellt ist, zeigt wie das Programm zuerst überprüft ob ein Speichern oder Update Button gedrückt wurde und anschließend je nach Fall einen unterschiedlichen Case ausführt. Um zu erkennen welcher Button gedrückt wurde, wird aus den einzelnen booleschen Werten ein Array erstellt, das anschließend in einen Integer Wert umgewandelt wird. Jeder gedrückte Button ergibt daher einen eindeutigen Integer Wert. Tabelle 10 stellt dar welcher Button welchen integer Wert zur Folge hat.

Tabelle 10: Zusammenhang zwischen gedrückten Button und dem erstellten Integer Wert

Gedrückter Button	Erstellter Integer Wert
Save Ignition	1
Save PID	2
Update PID	4
Update Ignition	8

Der in Abbildung 36 gezeigte Code zeigt den Case vier, sprich den Code, der ausgeführt wird, falls der Update PID Button gedrückt wurde. Der Code zeigt wie die einzelnen Parameter in die Datei gespeichert werden.

3.2.3.12 Smooth_Signals

Die Wägezellen haben leider ein gewisses Rauschen. Um den Einfluss des Rauschens zu minimieren wird dieses geglättet. Dazu wird das arithmetische Mittel einer bestimmten Anzahl von Messwerten gebildet. Durch das Bilden des Mittelwertes wird das ganze System träger, um dafür zu sorgen, dass der PID- Regler gut arbeitet, sollte die Anzahl der Mittelwerte so gering wie möglich gehalten werden, jedoch hoch genug, dass der PID nicht auf einzelne Ausreiser der Wägezellen reagiert. Als guter Kompromiss hat sich fünf als Anzahl der Messwerte für die Mittelwertbildung herausgestellt.

3.2.3.13 Unbundle

Wie erwähnt wurde, werden eine Vielzahl von Daten im Hauptprogramm in einem Cluster dargestellt. Da auf gewisse Werte dieses Cluster öfters zuggriffen wird und dies mit einem relativ aufwendigen Code einher geht, wurde dafür eine eigene VI erstellt.

Die VI verfügt über einen Eingang, der Eingang ist der Cluster. Zudem verfügt die VI über sieben Ausgänge. In Tabelle 11 werden die einzelnen Ausgänge aufgelistet.

Tabelle 11: Ausgänge der Unbundle VI

Ausgang	Beschreibung
setPoint	Hierbei handelt es sich um den gewünschten Staubeintrag pro Rührer in Gramm pro Sekunde.
slopes	Beinhaltet sämtliche Steigungen der einzelnen Wägezellen.
Intercepts	Beinhaltet sämtliche Ordinatenabschnitte der einzelnen Wägezellen.
smoothing_length	Gibt an über wie viele Werte der Mittelwert der Signale gebildet werden soll.
PID Values	Gibt die einzelnen Werte des PID Reglers an.
PID	Gibt die einzelnen Werte des PID Reglers an plus eine Bezeichnung für den gewählten Staub.
PWM	Gibt den Bereich an, in welchem die PWM Signale vom PID-Regler erstellt werden können.

Abbildung 37 zeigt das Aufschlüsseln des Clusters, wie zu sehen ist, werden einzelne Parameter anschließend wieder zu einem Cluster zusammengefasst. Da diese nicht als einzelne Parameter genutzt werden, so braucht zum Beispiel der PID Regler seine PID-Parameter und die Grenzen in einem Cluster.

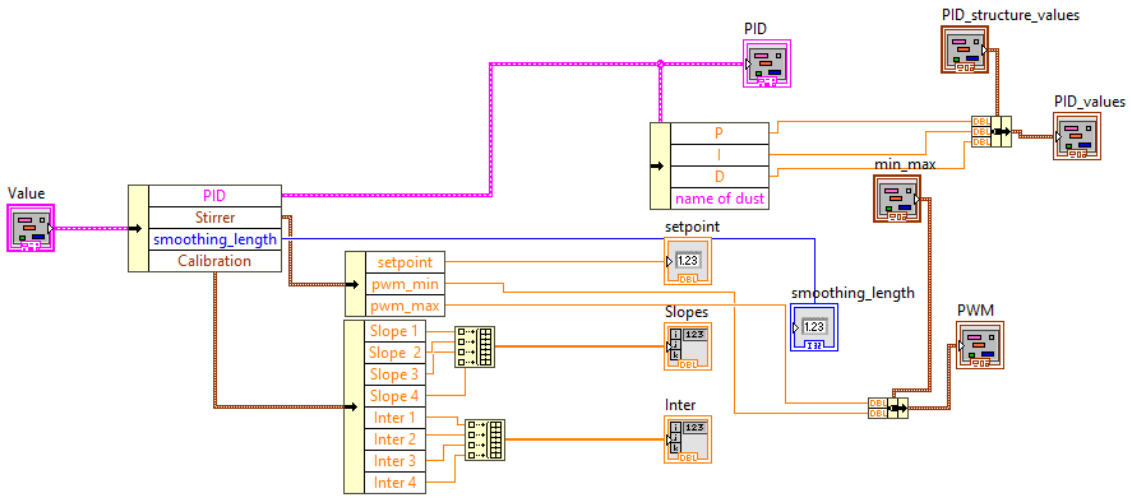


Abbildung 37: Blockdiagramm der Unbundle VI

3.2.4 Hauptprogramm

Sämtlicher Code des Hauptprogramm befindet sich in einer While-Schleife. Diese wird solange durchlaufen bis der Nutzer das Programm über den Button „Shut Down“ beendet. In der Schleife befindet sich eine Case Struktur, welche durch Enums angesteuert wird. Jeder Case muss am Ende den entsprechenden Enum setzen, um dafür zu sorgen, dass der richtige Case ausgeführt wird. Hierbei kann zwischen fünf Enums gewählt werden. In Tabelle 12 werden sämtliche Cases kurz beschrieben.

Tabelle 12: Auflistung sämtlicher Haupt-Cases des Hauptprogrammes.

Enum/Case	Beschreibung
Start	Dieser Case wird am Start des Programmes ausgeführt. Es wird unter anderem die Verbindung mit den Arduino hergestellt und Parameter werden initialisiert.
Dust_Entry	Wie der Name schon sagt, wird in diesen Case der Staub eingetragen und die automatische Zündung durchgeführt.
Wait	Dieser Case wird immer dann ausgeführt, wenn das Programm auf einen Befehl wartet.
Calibration	Dieser Case enthält den Code, um die Kalibration einer Wägezelle durchzuführen.
Ignition	Wird die Zündung per Fernbedienung gestartet, wird der Case ausgeführt, welche die Zündsequenz ausführt.
Exit	Beendet der User das Programm, muss die Verbindung zum Arduino ordnungsgemäß geschlossen werden, dies ist die Hauptaufgabe dieses Cases.

Im Folgenden wird nur auf den Dust_Entry und Wait Case näher eingegangen, da diese Cases für das Grundverständnis des Programmes fundamental sind.

Wait

Diesen Case kann man in gewisser Weise als Standard Case bezeichnen, da er mit Abstand der Case ist, welcher am häufigsten ausgeführt wird, zudem wird er nach jedem anderen Case ausgeführt.

Dieser Case besteht aus einer Eventstruktur mit einem Timeout von 10 Milisekunden. Bei sämtlichen Events handelt es sich um Wertänderung eines Buttons. Im Folgenden werden kurz die vier Events aufgelistet.

- Kalibration: Setzt das Enum, sodass als nächstes der Case für die Kalibration ausgeführt wird.
- Shut Down: Setzt das Enum, sodass als nächstes der Exit-Case ausgeführt wird.
- Settings: Ruft die VI „Settings“ auf, mithilfe welcher alle Parameter eingeben werden können.
- Dust Entry: Ruft den umfangreichsten Case auf, dieser Case enthält den Staubeintrag.

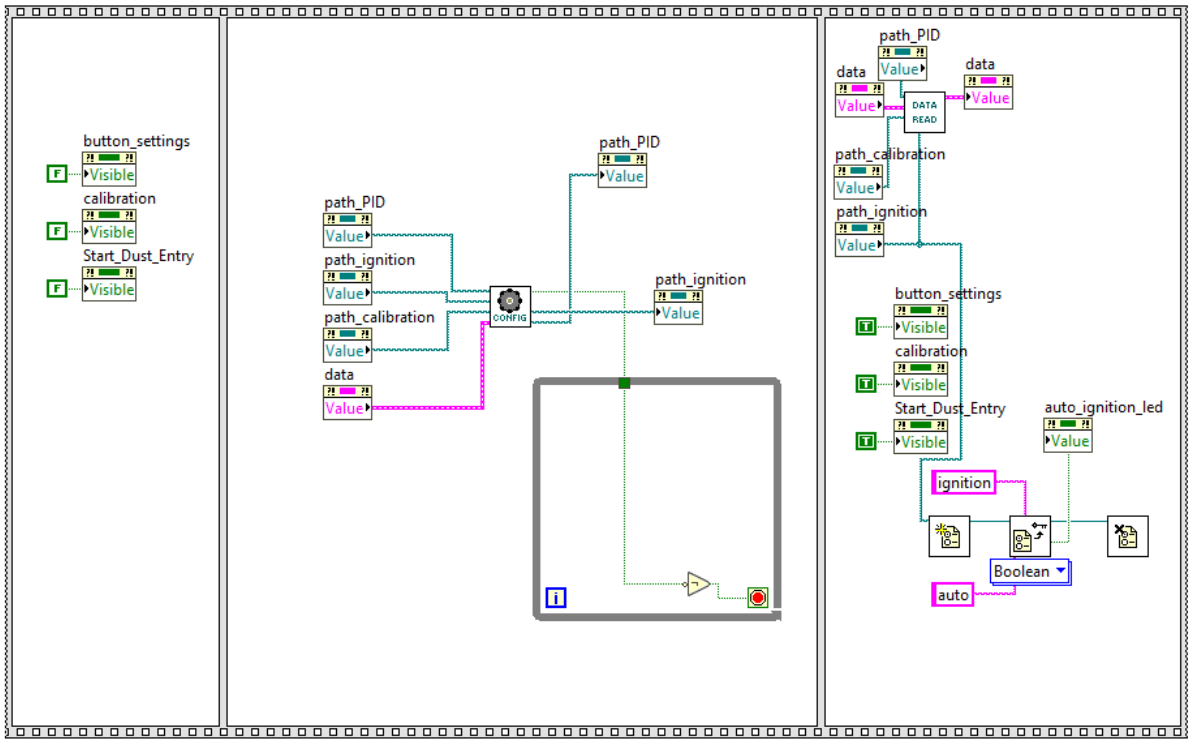


Abbildung 38: Code für den Event das der Settings-Button gedrückt wurde.

Abbildung 38 zeigt den Code, der ausgeführt wird, falls der User den Settings-Button gedrückt hat. Man kann gut erkennen wie zwei Sub-VIs aufgerufen werden und andere Button für ein- und ausgeblendet werden. Am Ende des Cases werden schließlich die gewünschten Parameter in das Hauptprogramm übernommen, mithilfe der Data Read VI.

Dust Entry

Wen der User auf ‚Start‘ klickt wird dieser Case ausgeführt. Der Case besteht aus drei Sequenzen. In der Start Sequenz werden sämtliche Button ausgeblendet die während des Staubeintrages nicht benötigt werden. Zudem werden diverse globale Variablen gesetzt und die Schrittmotoren werden aktiviert sowie die Warnleuchte wird entsprechend geschalten. Die mittlere Sequenz beinhaltet die Steuerung des Staubeintrages. Sie besteht aus einer Schleife, in welcher sich der Code befindet. Das Herzstück der Schleife stellt der PID-Regler da. Der

Regler sorgt dafür, dass die Drehzahlen der einzelnen Rührer so angepasst werden, sodass sich die gewünschte Förderrate ergibt. Die Schleife beinhaltet eine Menge von SubVIs. Je nach gewählten Einstellungen kann der Staubeintrag und die Zündung vollkommen autonom stattfinden. Die Schleife wird beendet sobald ein Staubeintragskriterium erfüllt ist. In der letzten Sequenz wird gegebenenfalls die Zündung abgeschlossen, diverse globale Variablen gesetzt und entsprechende Buttons eingeblendet.

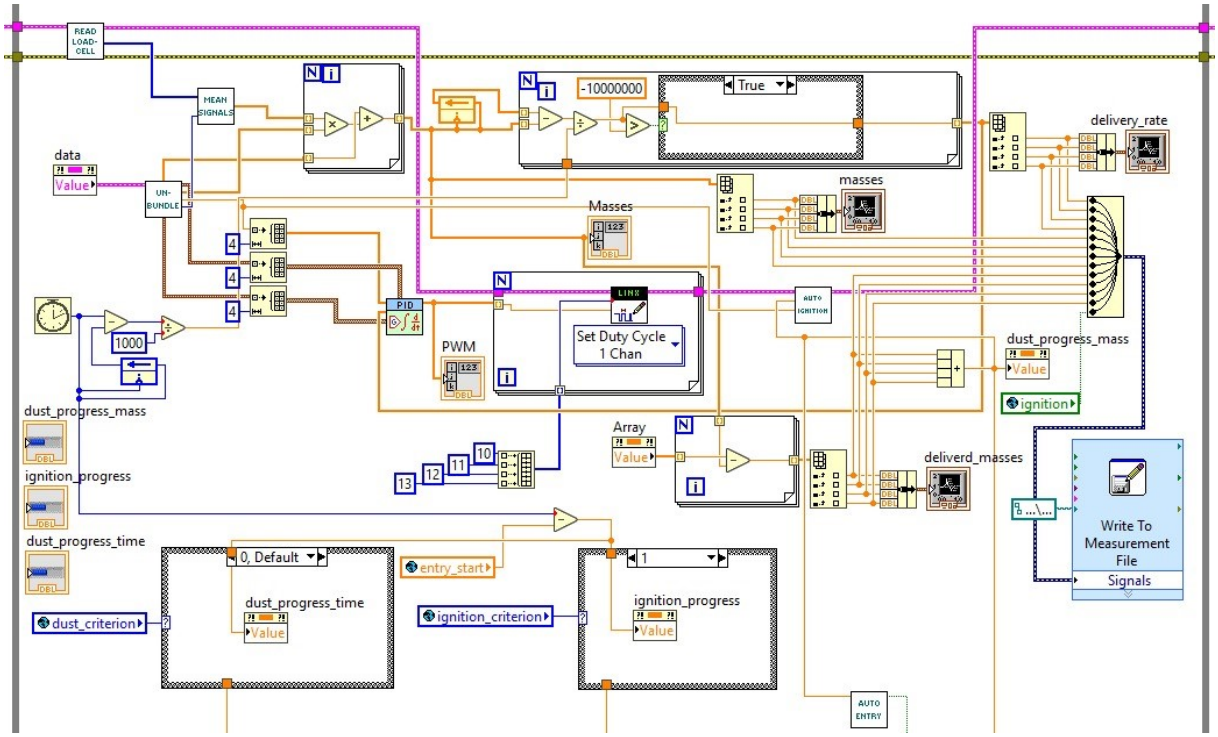


Abbildung 39: Code für den Staubeintrag

Abbildung 39 zeigt einen Großteil des Codes für den Staubeintrag. In jedem Schleifendurchlauf werden die aktuellen Werte der Wägezellen ausgelesen. Mit diesen Werten und jenen des vorigen Durchlaufs werden die einzelnen Förderraten berechnet. Zudem werden die einzelnen Diagramme aktualisiert und die Daten des Diagramms in eine Textdatei gespeichert. Dadurch kann der Versuch nachträglich optimal analysiert werden.

3.2.5 Dateien

Das Programm greift auf mehrere Dateien zu. Im Folgenden wird kurz auf diesen Dateizugriff und deren Aufbau eingegangen.

Textdateien

Es gibt folgende unterschiedliche Typen von Textdateien:

- PID: In dieser Datei werden alle relevanten Parameter für die PID-Regelung gespeichert. Die Datei ist in zwei Abschnitte unterteilt: Im ersten Abschnitt sind generelle Daten gespeichert. Im zweiten Abschnitt sind die PID-Parameter gespeichert. Abbildung 40 zeigt den Aufbau einer PID-Textdatei, gut zu erkennen sind die zwei Abschnitte der Datei.

```
[generalData]
smoothingSampleLength = 10    ;
rotationSpeedMax = 0,990000    ;
rotationSpeedMin = 0,000000    ;
setpoint = -10;
nameConfig = "sisco_28"

[PID]
PIDKC = 0,200000    ;
PIDTi = 0,010000    ;
PIDTd = 0,002000    ;
```

Abbildung 40: Aufbau einer PID-Textdatei

- Kalibration: Diese Dateien sind in zwei Abschnitte unterteilt. Im ersten Abschnitt sind sämtliche Steigungen der Wägezellen aufgelistet. Im zweiten Abschnitt sind sämtliche Ordinatenabschnitt der Wägezellen aufgelistet.
- Zündung/Staubeintrag: Dieser Dateityp enthält alle Informationen, die den Ablauf des Versuches bestimmen. Wie in Abbildung 41 zu erkennen ist, enthält dieser Datentyp Informationen, ob die automatische Zündung verwendet werden soll, welches Abbruchskriterium für den Staubeintrag verwendet werden soll und das Kriterium für die automatische Zündung.

```
[dustEntry]
criterion = 0    ;
mass = 200,000000    ;
time = 15,000000    ;
setpoint = 3,000000    ;

[ignition]
criterion = 1    ;
auto = FALSE    ;
mass = 150,000000    ;
time = -13,000000    ;
duration = 0    ;
```

Abbildung 41: Aufbau einer Zündung/Staubeintrag -Textdatei

Zudem gibt es eine weitere Textdatei, die sogenannte Path-Datei. In dieser sind die Pfade zu den zu verwendenden, jeweils eine Datei der obengenannten Dateitypen, Textdateien gespeichert.

LVM-Dateien

LVM-Dateien sind Messdateien von National Instruments. Dateien des Typs können mit beliebigen Texteditoren geöffnet werden. Dieser Dateityp wird dann verwendet, wenn viele Messdaten schnell in eine Datei gespeichert werden sollen, da LabVIEW diese Messdaten blockweise in die Datei schreibt. Im Programm werden drei verschiedene LVM-Dateien verwendet.

- Messung Thermoelemente: Sobald der Staubeintrag gestartet wird, startet die Aufzeichnung sämtlicher Thermoelementwerte. Die Messdaten werden blockweise in eine LVM-Datei gespeichert. In der ersten Spalte der Datei steht der jeweilige Zeitstempel der Messung. Danach folgen die Spalten mit den Messwerten der Thermoelemente.
- Messung Photodioden: Analog zur Messung der Thermoelemente findet auch die Aufzeichnung der Photodiodenwerte statt.
- Aufzeichnung Staubeintrag: Um den Verlauf der Förderrate und der eingerieselten Masse für jeden Versuch genau darstellen zu können, werden alle relevante Daten während des Staubeintrages abgespeichert. Wie bei den beiden anderen Dateien enthält auch die Datei in jeder Zeile einen Zeitstempel.

3.3 Integration der Zündung und Warnleuchte

Zu Beginn dieser Arbeit war die Zündung für das Flammenrohr schon entwickelt und hergestellt. Als Zündung dient ein Hochspannungs-Plasmabogen, welcher zwischen den beiden Elektroden gebildet wird. Die Zündung wird mit 12 Volt betrieben. Diese Spannung reicht nicht aus, darum wird die Spannung über einen Zeilentransformator verstärkt. Zudem wird vor dem Transformator die Gleichspannung in Wechselspannung umgewandelt. Für eine detailliertere Dokumentation möchte ich auf die Arbeit von Dipl.-Ing. Julian Glechner verweisen [2] .

3.3.1 Zündung

Im Zuge dieser Arbeit wurde die Ansteuerung der Zündung fertig erstellt und die Zündung in das Flammenrohr integriert. Die Zündung verfügt über drei Anschlüsse: zwei Anschlüsse sind für die Spannungsversorgung notwendig. Mit Hilfe des dritten Anschlusses kann die Zündung geschalten werden. Zum Zünden müssen an diesen Anschluss fünf Volt angelegt

werden. Die Zündung kann automatisch oder von Hand erfolgen. Die Zündung per Hand hat die Anforderung, dass sie aus sicherer Distanz erfolgen soll.

Die Anforderungen welche gewünscht sind:

- Die Ansteuerung sollte nicht kabelgebunden sein.
- Aus Sicherheitsgründen sollte die Zündung von zumindest zwei Signalen ausgelöst werden.
- Die Zündung sollte aus mindestens 15 Meter durchführbar sein.
- Die Ansteuerung sollte möglichst kostengünstig sein.

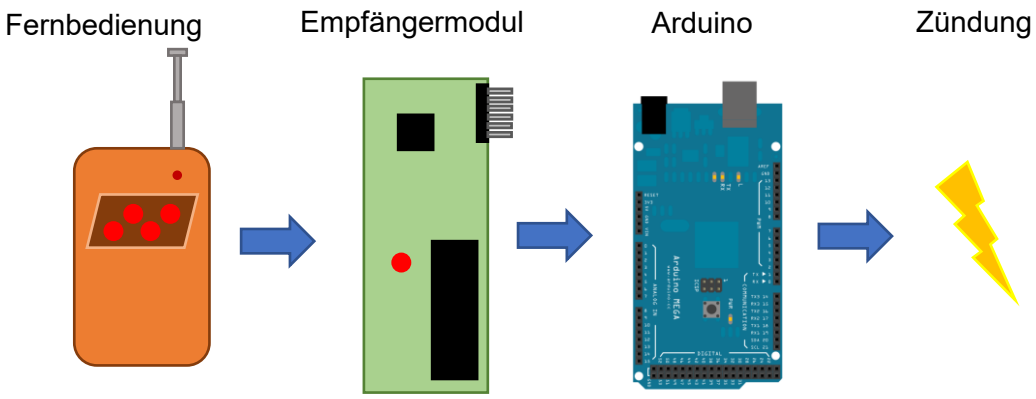


Abbildung 42: Schema der händischen Zündung

In Abbildung 42 ist skizziert wie die Zündung per Hand funktioniert. Der Bediener drückt zwei Tasten der Fernbedienung, am Empfänger gehen die Signale ein. Diese werden vom Arduino eingelesen und der Zündungsvorgang wird gestartet. Wichtig ist, dass der Schalter der Zündung auf ‚ON' gestellt ist. Dieser Schalter dient dazu die Versorgungsspannung für die Zündung zu schalten. Aus Sicherheitsgründen darf der Schalter nur knapp vor dem Versuch aktiviert werden.

Schließlich wurde eine Radiofrequenz Fernbedienung für die Zündung per Hand gewählt, welche über vier Kanäle verfügt. Bei der Fernbedienung handelt es sich um das Wireless Remote Control Module 2262/2272. Zwei der Kanäle werden verwendet, um die Zündung zu starten. Als Empfänger-Platine wurde eine PT2272-M4 basierte Platine verwendet. Die Empfängerplatine muss nur mit Spannung versorgt werden. Die Platine besitzt für jeden Kanal einen eigenen Ausgangspin, welcher auf fünf Volt gesetzt wird, falls das entsprechende Signal empfangen wird. Der Arduino Mega ist mit den zwei relevanten Pins des Empfängermoduls verbunden. Sind beide Kanäle aktiv wird die Zündung gestartet, dafür schaltet der Arduino über einen Transistor die Zündung ein. Dafür werden fünf Volt an den Steueranschluss der Zündung gelegt. Wird der Transistor nicht geschalten ist der Steueranschluss der Zündung mit

der Masse verbunden. Tabelle 13 zeigt wie der Arduino mit der Platine des Empfängermoduls verbunden ist.

Tabelle 13: Verschaltung des Hauptarduinos mit der Empfängerplatine

Pin Empfänger Modul	Pin Arduino Mega
D1	D2
D2	D3

3.3.2 Warnleuchte

Am Flammenrohr wurde eine Warnleuchte angebracht. Die Warnleuchte besteht aus drei LED-Leuchten und einer Signalsirene. Je nach Betriebszustand werden die Leuchten und die Sirene unterschiedlich geschalten. In Tabelle 14 sind die unterschiedlichen Betriebszustände und die dazugehörige Warnsignalkombination aufgelistet.

Tabelle 14: Ausgabe der Betriebszustände per Warnleuchte

Betriebszustand	LED-Grün	LED-Orange	LED-Rot	Sirene
Programm in Standby-Modus, automatische Zündung deaktiviert	AN	-	-	-
Programm in Standby-Modus, automatische Zündung aktiviert	-	AN	-	-
Staubeintrag ist im Gange	-	AN	AN	
Zündung findet in weniger als vier Sekunden statt.	-	-	AN	AN

Die Warnleuchte wird mit 24 Volt betrieben und verfügt über eine gemeinsame Masseleitung. Die drei LED-Leuchten und die Sirene werden mittels vier Leitungen separat mit 24 Volt versorgt. Für die Versorgung dient ein 24 Volt Gleichspannungsnetzgerät. Die Anforderungen an das Netzgerät sind:

- Robustheit, da das Netzgerät oft transportiert werden soll und während der Versuche erhöhter Staubkontamination und evtl. erhöhter Luftfeuchtigkeit ausgesetzt ist.
- Das Gerät muss 24 Volt liefern, um für die Versorgung geeignet zu sein.

- Das Gerät muss auf einer Hutschiene des Typs DELTA3F-ST-GEL-WC montierbar sein.
- Das Netzgerät sollte so klein wie möglich sein, um Platz im Schaltkasten zu sparen
- Der Preis sollte möglichst niedrig sein.

Schließlich wurde ein Netzgerät des Typs Ndr-120-24, der Firma Mean Well verwendet. Die Ansteuerung wurde zudem ein vier-Kanal- Relaismodul eingesetzt, welches sich auf derselben Platine befindet wie das Empfängermodul der Fernbedienung für die Zündung. Jeder Kanal schaltet dabei eine LED oder die Sirene. Jeder Kanal besitzt über einen Eingangspin. Wird dieser auf Masse gezogen schaltet das jeweilige Relais. Die einzelnen Eingangspins werden von dem Arduino Mega geschalten.

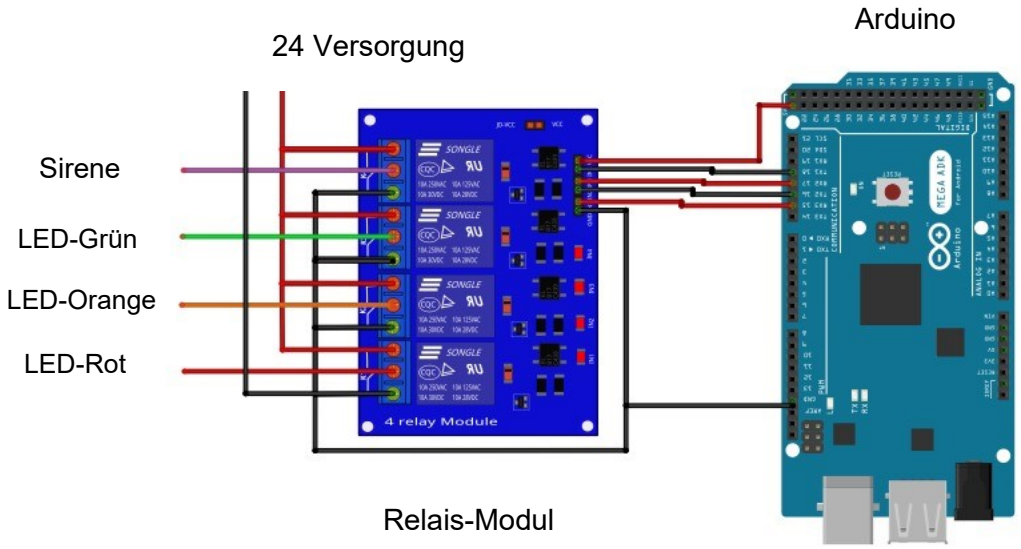


Abbildung 43:Verschaltung der Warnleuchte, des Relaismoduls und des Arduinos

Abbildung 43 zeigt wie die einzelnen Komponenten untereinander verbunden sind. Wie zu erkennen wird, werden die LEDs und die Sirene entweder auf 24 Volt oder Masse geschalten, je nach Stellung der Relais. Das Relaismodul wird über den Arduino mit fünf Volt versorgt. Wichtig zu erwähnen ist, dass die Relais schalten, wenn am entsprechenden Eingang Masse anliegt. Will man zum Beispiel die Sirene einschalten, muss man am Pin „In 1“ Masse anlegen. Gut zu erkennen ist auch, dass die Masse des Netzgeräts mit der Masse des Arduinos verbunden ist. Tabelle 15 listet genau auf wie die Pins der Komponenten genau untereinander verbunden sind.

Tabelle 15: Verschaltung des Arduino, des Relaimoduls und er Warnleuchte

Arduino Mega Pin	Relaimodul Pin	Kanal-Relaimodul	positive Versorgung
16	In 2	2	LED-Grün
17	In 3	3	LED-Orange
18	In 4	4	LED-Rot
15	In 1	1	Sirene

3.4 Messtechnik

Um die Flammengeschwindigkeit zu messen, werden zwei parallele Systeme verwendet. Einerseits erfolgt die Messung mittels Thermoelemente und andererseits mittels Photodioden. Diese Verfahren kommen bereits bei der kleinen Laborapparatur zur Anwendung. Diese Messtechniken wurden im Zuge dieser Arbeit so erweitert, dass sie für den großen Versuchsaufbau eingesetzt werden können.

3.4.1 Bestimmung der Flammengeschwindigkeit mittels Thermoelemente

Im Flammenrohr werden in gleichmäßigen Abständen 0,08 mm dünne Thermoelemente angebracht. Die dünnen Drähte gewährleisten eine äußerst geringe Trägheit, die entscheidend für ein schnelles Ansprechen auf einen Temperaturwechsel ist. Die verwendeten Thermoelemente wurden selbstangefertigt. Bei den zwei verwendeten Metallen handelt es sich um Nickel und um Nickel-Chrom, sprich Typ K Thermoelement. Ein wichtiges Kriterium bei Auswahl des Thermoelementleitungen ist, dass sie nicht brennbar sind. Bei den Thermoleitungen wurden die Modelle IEC-TFCY-003 und IEC-TFAL-003 der Firma OMEGA Engineering verwendet. Sämtliche Thermoelemente sind an eine NI-Messkarte angeschlossen. Die Software speichert die Messwerte in eine Messdatei, die danach ausgewertet werden kann. Die Daten werden mittels Maximalwert- und Flankenmethode ausgewertet. Hierfür wurde die bereits vorhandene Auswertesoftware von vier auf sieben Kanäle erweitert. Für eine genauere Dokumentation der Auswertungsmethode möchte ich auf die Projektarbeit von, Dipl.-Ing. Florian Toth [15] verweisen. Abbildung 44 zeigt den prinzipiellen Aufbau der Messtechnik.

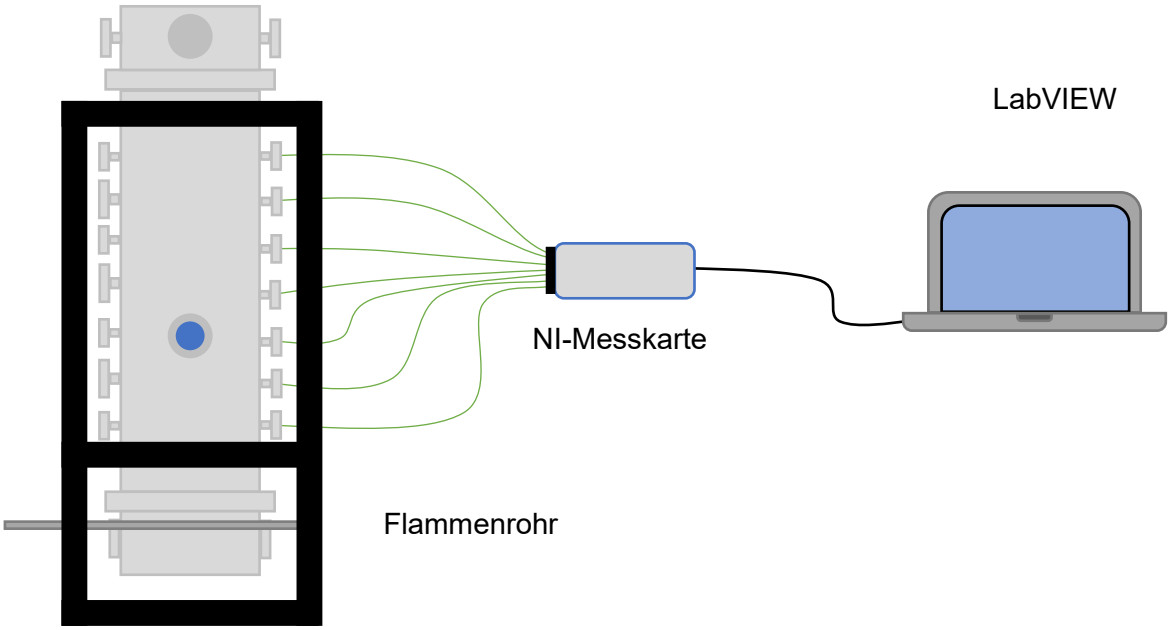


Abbildung 44: Prinzip für die Messung der Flammengeschwindig per Thermoelemente.

3.4.2 Bestimmung der Flammengeschwindigkeit mittels Photodioden

Die Flammengeschwindigkeit wird zusätzlich mit der Hilfe von Photodioden gemessen. Eine genauere Dokumentation der Messmethode ist in der Bachelorarbeit von, Dipl.-Ing. Florian Toth [16] ersichtlich. Die Messtechnik wurde so angepasst, dass diese in der großen Versuchsanlage verwendet werden kann. Die Photodioden befinden sich dabei nicht direkt im Flammenrohr, sondern auf einer Platine. Zusätzlich befindet sich auf der Platine noch die Messverstärker-Schaltung welche aus sieben Transimpedanzverstärker besteht. Das Ausgangsignal der Messschaltung wird zur Durchführung der Analog-Digital-Wandlung an eine NI- Karte des Typs Ni USB-6008 angeschlossen. Die Ni-Messkarte ist an einem PC angeschlossen, wo die Daten per LabVIEW verarbeitet werden. Abbildung 45 stellt das Prinzip der Messung dar.

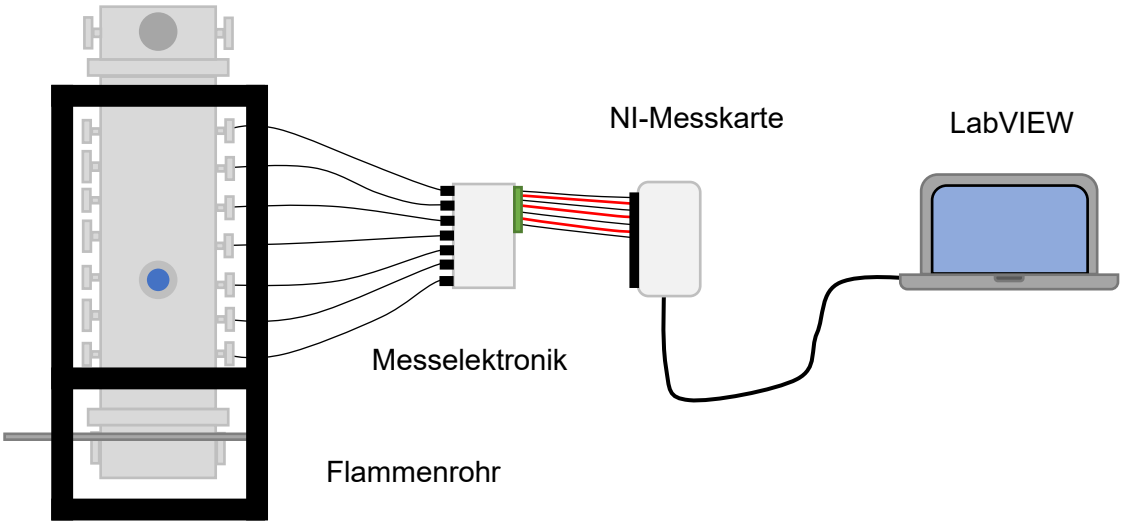


Abbildung 45: Prinzip für die Messung der Flammengeschwindig per Photodioden.

Lichtwellenleiter sind in gleichmäßigen Abständen im Flammenrohr montiert. Das jeweilige andere Ende des Lichtwellenleiter ist mit der Messplatine verbunden, welches dort als Lichtquelle für die Photodioden dient. Die Messchaltung konnte mit kleinen Änderungen aus dem kleinen Flammenrohr übernommen werden. Die Schaltung wurde von vier auf sieben Kanäle erweitert, zudem wurden die Operationsverstärker mit Stützkondensatoren ausgerüstet. Die Schaltung im kleinen Rohr ist zudem auf 10 V ausgelegt. Zur Spannungsversorgung für das große Rohr sind allerdings nur Netzgeräte mit 12 und 24 V verbaut. Bei dem Transimpedanzverstärker handelt es sich um eine Schaltung bestehend aus einem Operationsverstärker. Ein Operationsverstärker kann niemals eine Spannung mit exakt seiner Versorgungspannung ausgeben, sondern nur eine Spannung, die um einen gewissen Faktor kleiner ist als die Versorgungspannung. Dieser Faktor ist fast für alle Operationsverstärker kleiner als 1,3 V. Würde man die Schaltung für das kleine Rohr mit 12 V versorgen, wäre die maximale Ausgangspannung 10,3 V. Der AD- Wandler der verwendeten Messkarte hat allerdings eine maximale Eingangsspannung von 10 V. Da nur 12 V für die Versorgung zu Verfügung stehen, muss diese Spannung um minimiert werden. Dafür wurden drei in Serie geschaltete Zenerdioden verwendet. An den Zenerdioden fallen pro Zenerdiode 0,6 V an. Um Spannungsschwankungen auszugleichen, wurde nach den drei Zenerdioden ein Stützkondensator mit 10 Mikrofarad verbaut.

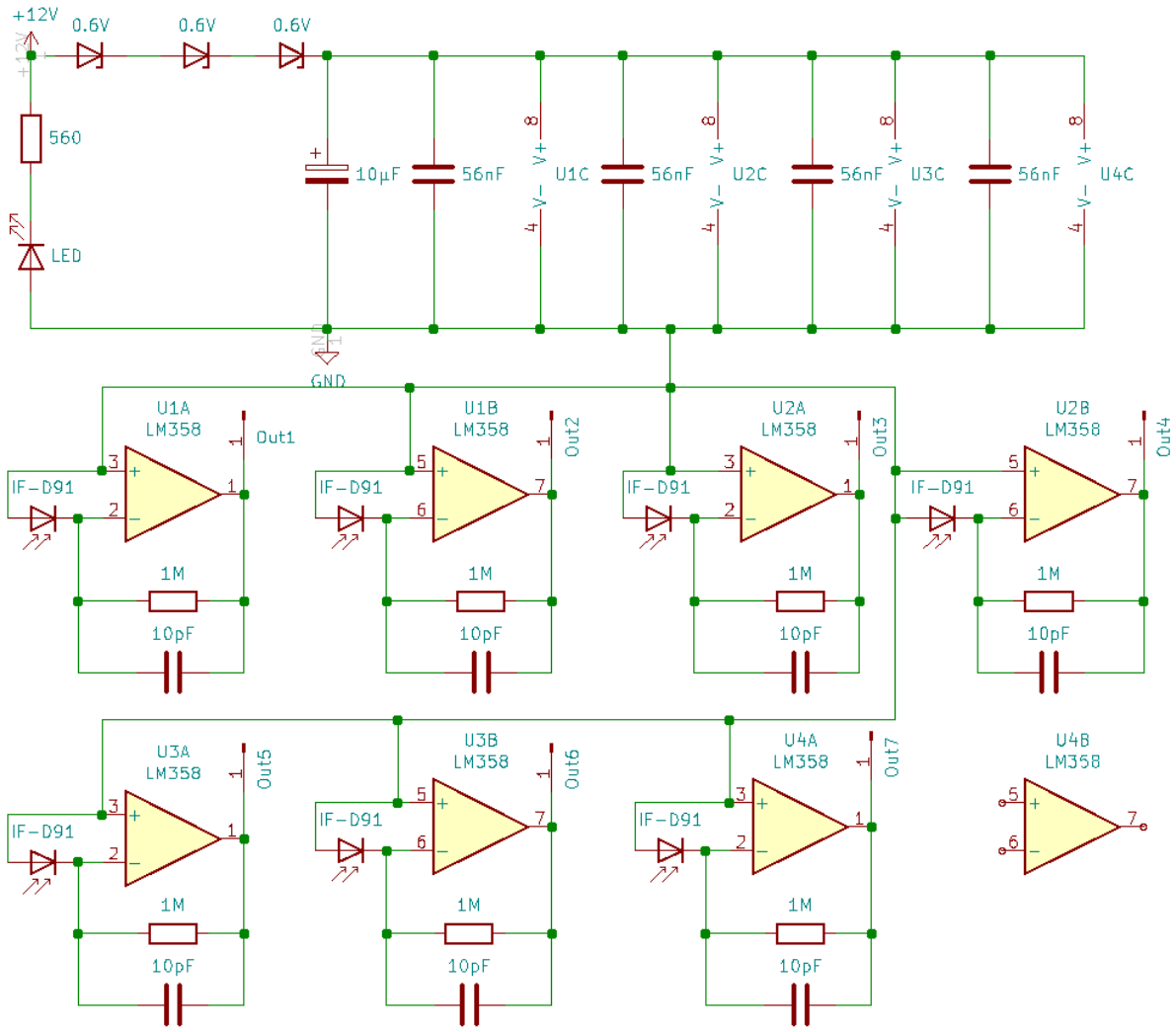


Abbildung 46: Schaltplan der Messverstärkerschaltung.

Abbildung 46 zeigt den Schaltplan für die erstellte Messverstärkerschaltung, der obere Bereich des Schaltplans zeigt die Spannungsversorgung der einzelnen Operationsverstärker. Wie zu erkennen ist, wurde zudem eine LED eingebaut, welche anzeigt ob die Platine mit Spannung versorgt wird.

3.5 Schaltkasten und Verkabelung

3.5.1 Energieversorgung

Da die Versuchsanlage im Freien betrieben wird, muss die Anlage durch einen mobilen Generator betrieben werden. Die Anforderungen an einen solchen Generator sind wie folgt:

- Der Generator muss genügend Leistung bieten, um die Anlage betreiben zu können.
- Der Generator muss wetterfest sein, da er im Freien ungeschützt eingesetzt wird.
- Für den Transport sollte der Generator möglichst klein und leise sein.
- Eine möglichst niedrige Geräuschkulisse ist wünschenswert.
- Die Spannungsqualität zum Betreiben von elektrischen Komponenten muss ausreichend sein.

Die beste Lösung stellt ein sogenannter Inverter Stromerzeuger da, zum Beispiel jenes Modell EU10i der Marke Honda.

3.5.2 Schaltkasten

Sämtliche elektronische Bauteile wurden in einen Schaltkasten auf Hutschienen montiert. Der Schaltkasten wird während eines Versuches am Flammenrohr montiert. Sämtliche Kabel können außen an den Schaltkasten angeschlossen werden. Dies ist nötig, um den Schaltkasten abmontieren zu können. Sämtliche Kabel und Steckverbindungen wurden so ausgelegt, dass sie mindestens IP 67 erfüllen. Der Schaltkasten wird über den Inverter Stromerzeuger mit Spannung versorgt.

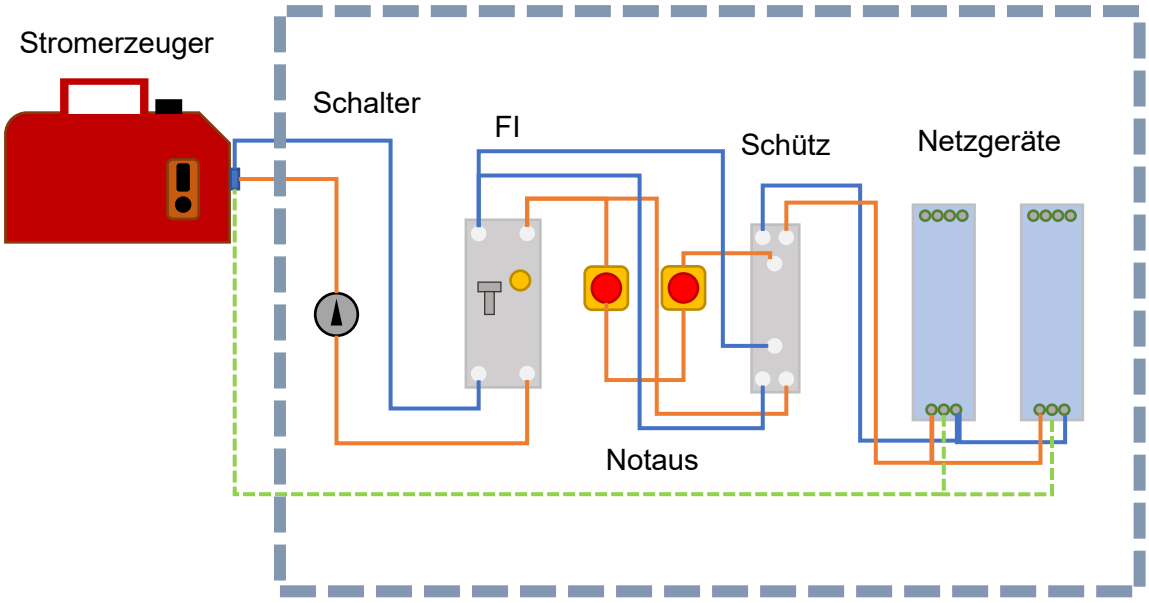


Abbildung 47: Prinzip der Stromversorgung

Abbildung 47 zeigt das Prinzip der Stromversorgung. In der Skizze ist die Erdung grün, die Phase braun und der Nullleiter blau gezeichnet. Sämtliche Bauteile, die im grau strichlierten Rechteck dargestellt sind, sind am oder im Schaltkasten montiert. Es ist ersichtlich, dass der

Ein/Auschalter die Phase zu- oder wegschaltet. Der FI-Schutzschalter ist vor den beiden Notaus-Schaltern eingebaut. Nur wenn beide Notaus-Schalter nicht gedrückt sind, schaltet der Schütz die Phase und den Nullleiter durch. Nach dem Schütz werden die beiden Netzgeräte mit Spannung versorgt. Sämtliche anderen Bauteile werden von diesen beiden Netzgeräten betrieben. Der Stromerzeuger ist per Erdungsspitze geerdet. Sämtliche Komponenten wurden im Schaltkasten, wie in dieser Arbeit und in der Diplomarbeit von Dipl.-Ing. Glechner beschrieben, mit Spannung versorgt und untereinander verschalten. [2] Für alle Platinen wurde ein Gehäuse im 3D-Druckverfahren aus PLA hergestellt.

4 Ergebnisse und Funktionstest

4.1 Förderrate

Nachdem die Software und die Hardware fertiggestellt waren, konnte die Förderrate des Staubeintrages getestet werden. Als Siebeinsatz wurde für jeden Rührer ein Sieb mit einer Maschenweite von 200 µm verwendet. Aus Sicherheitsgründen wurde zur Vermeidung einer explosionsfähige Atmosphäre als Teststaub Zeolith verwendet. Zeolith wird hauptsächlich in der Bauindustrie verwendet. [17]

Jeder Rührer wurde einzeln gemessen und die Ergebnisse anschließend miteinander verglichen. Die Tests wurden mit drei verschiedenen Förderraten durchgeführt. Bei den Förderraten, die über die Software eingeben wurden, handelt es sich um 0,25 Gramm/Sekunde, 0,5 Gramm/Sekunde und 1,0 Gramm/Sekunde. Idealerweise sollten sämtliche Kurven der Rührer ident sein.

Der Versuch wurde wie folgt durchgeführt. Der befüllte Rührer wird samt Wägezelle über einer Laborwaage befestigt. Die Wägezelle wird vor jedem Versuch kalibriert. Eine Stoppuhr

wird neben der Laborwaage positioniert. Eine Videokamera wird so ausgerichtet, dass sie die Skale der Waage und die Stoppuhr aufnimmt. Die Videokamera wird gestartet, gefolgt von der Stoppuhr. Schließlich wird der Staubeintrag aktiviert. Dieser Versuch wird anschließend mit verschiedenen Förderraten und Rührern wiederholt. Der Versuchsaufbau ist in Abbildung 48 skizziert.

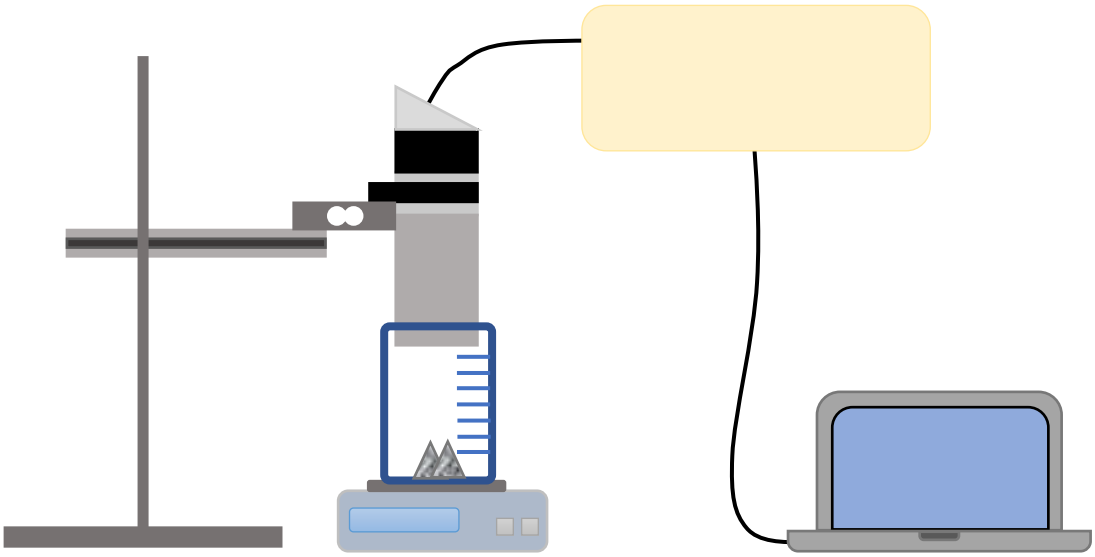


Abbildung 48:Versuchsaufbau für die Bestimmung der Förderrate

Nach der Versuchsdurchführung wird das Videomaterial ausgewertet. Die Ergebnisse der Auswertung sind in Abbildung 49 ersichtlich.

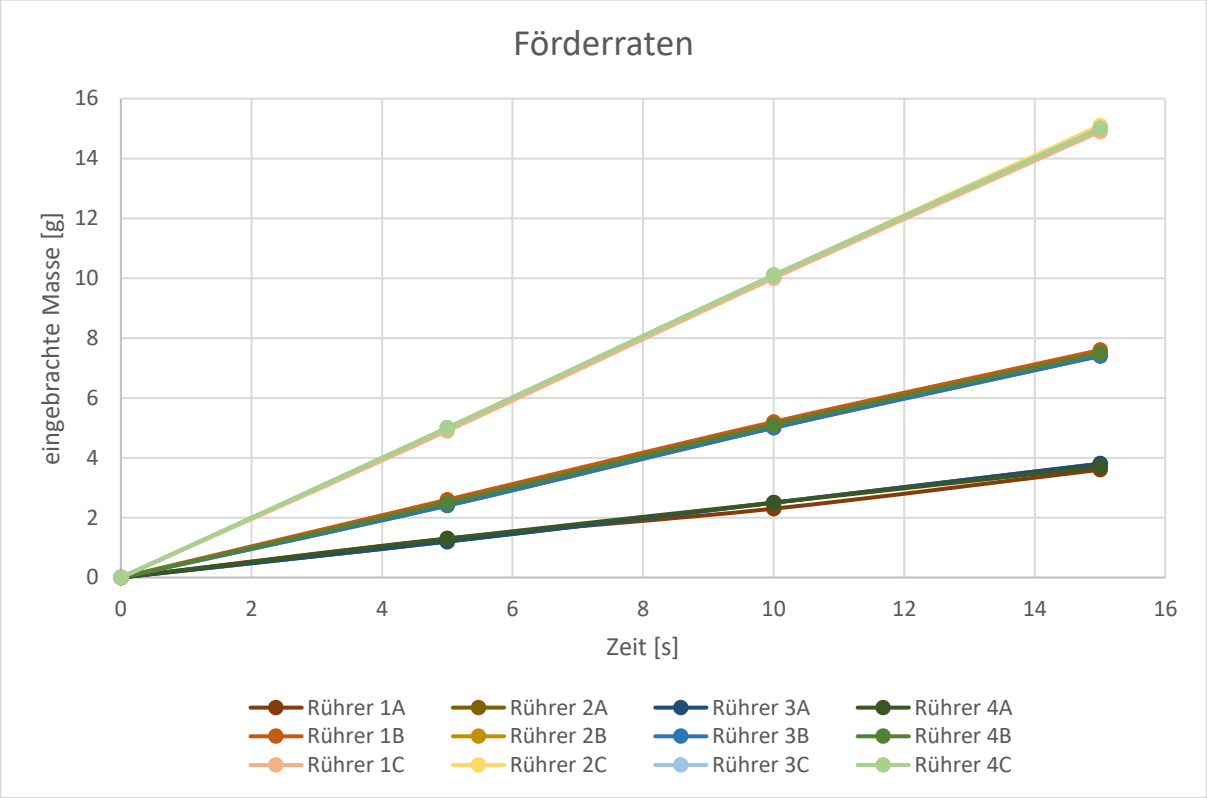


Abbildung 49: Geförderte Massen pro Rührer, A Sollwert 0,25 g/s, B Sollwert 0,5 g/s, C Sollwert 1 g/s

Wie in der Abbildung 49 ersichtlich ist, sind die Linien der einzelnen Rührer je Einstellung beinahe identisch. Der größte relative Fehler tritt bei einer Förderrate von 0,25 Gramm/Sekunde auf, der Unterschied zwischen dem Sollwert der geförderten Masse und der tatsächlichen Masse beträgt 1,3 %. Mit zunehmender Förderraten näheren sich die Geraden, der einzelnen Rührer immer mehr an. Bei größeren Förderraten wirkt sich das Rauschen der Wägezellen weniger stark aus. Zudem ist gut erkennbar, dass die Förderraten über die Zeit fast konstant sind. Mit abnehmender Füllhöhe der Rührer nimmt die Drehzahl der Rührer zu, bis die maximale Drehzahl erreicht ist.

4.2 Funktionstest

Nach dem Zusammenfügen sämtlicher Komponenten konnte mit der Testphase der Anlage, soweit möglich im „kalten“ Zustand, begonnen werden. Hierfür wurden alle Komponenten, Verbindungen und Verschaltungen gründlich erprobt. Nach Abschluss der Testphase wurde die Anlage zum Versuchsstandort am Zentrum am Berg (ZAB) am steirischen Erzberg transportiert.

Die erste „warme“ Inbetriebnahme und Testphase wurde am 9. Oktober 2019 durchgeführt. Am Versuchstag wurde Lycopodium-Staub als Versuchsmedium verwendet. Nach mehreren Funktionstests wurde schließlich der erste Versuch gestartet, welcher nicht gezündet hat. Nachdem die Förderrate und die Förderzeit angepasst wurden, konnte das Flammenrohr ein erstes Mal erfolgreich gezündet werden. Der anschließende zweite Versuch führte ebenfalls zur Zündung.

Bei beiden Versuche wurde die Anlage vollkommen autonom betrieben, sprich die Zündung und der Staubeintrag wurden vom Programm gesteuert.

Bei den zwei Versuchen handelt es sich hauptsächlich um Funktionsüberprüfungen. Anhand dieser wurde die Software angepasst werden konnte. So wurde in die Software unter anderem eingefügt, dass die Förderrate pro Rührer während des Staubeintrages in eine Messdatei gespeichert wird. Dies dient dazu, die einzelnen Versuche später genauer analysieren zu können.

Bei den Versuchen wurden im Flammenrohr zudem nur zwei Thermoelemente und zwei Lichtwellenleiter verwendet, da lediglich die Funktionalität der Messtechnik überprüft werden sollte.

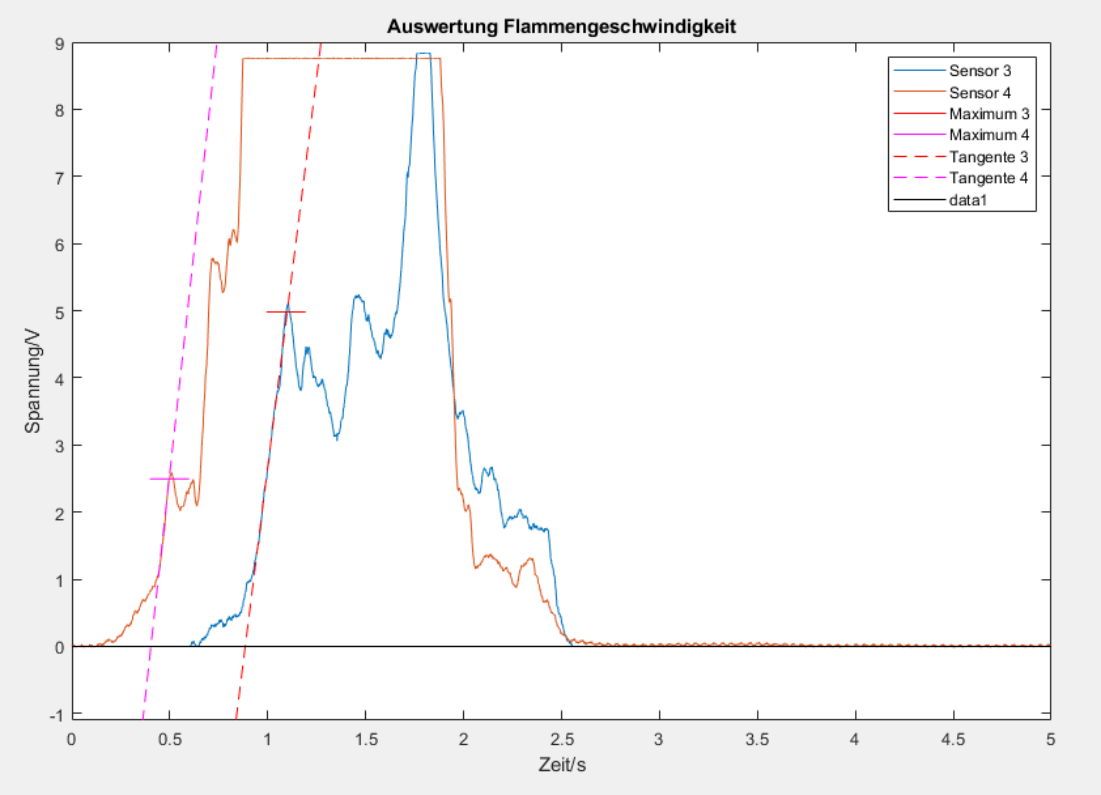


Abbildung 50: Auswertung der Flammengeschwindigkeit mittels Fotodioden vom ersten Testversuch

Abbildung 50 zeigte die Auswertung der gemessenen Photodioden Daten, mithilfe der von Dipl.-Ing Toth entwickelten Software. [16] Diese Software musste nur minimal angepasst werden. Wie zu erkennen ist, wurde die Flammenfront, wie gewünscht, von der Mess-Elektronik erfasst.

Abbildung 51 zeigt die Auswertung der gemessenen Daten der Thermoelemente. Es ist dargestellt, dass die Temperaturen wie gefordert aufgezeichnet wurden. Allerdings zeigten die Thermoelemente ein sehr träges Verhalten. Dieses Phänomen wurde erwartet, da nicht die selbst erstellten Thermoelemente verwendet wurden, sondern „normale“ mit relativ dicken Leitungen.

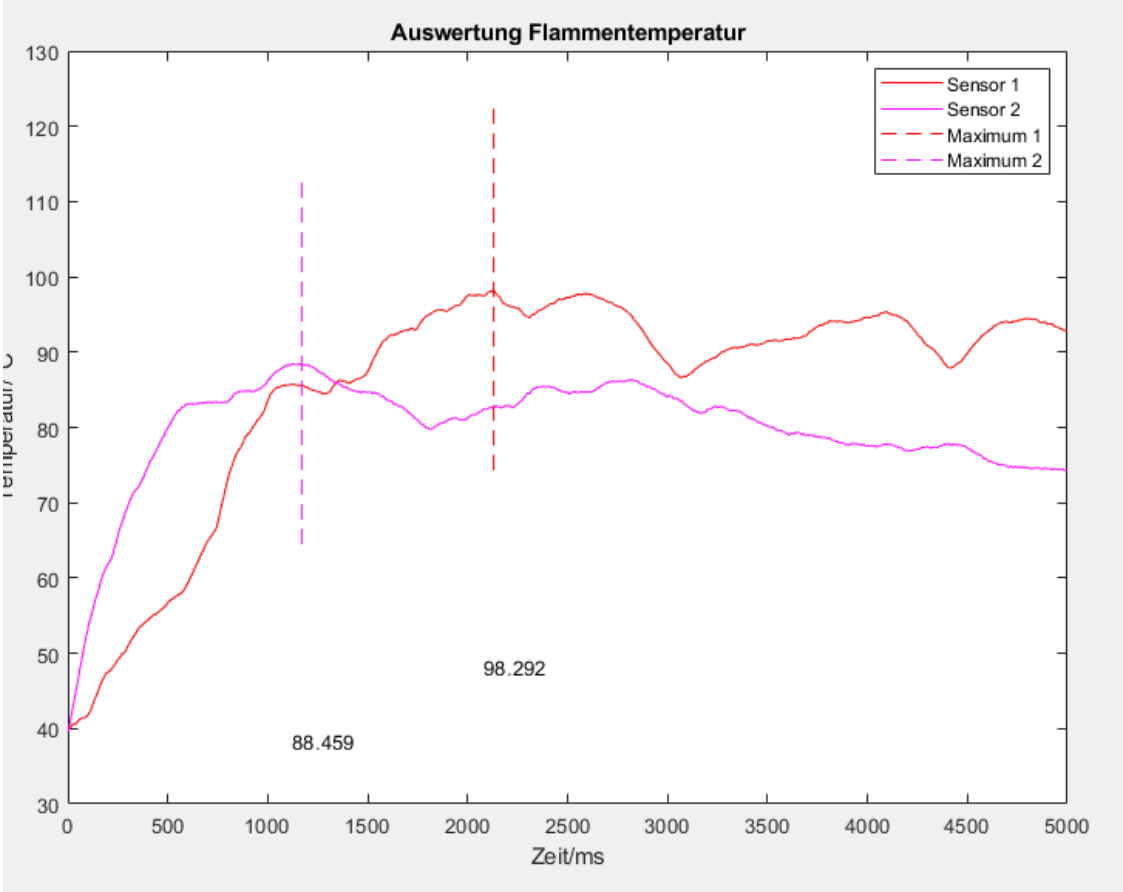


Abbildung 51: Auswertung der Flammengeschwindigkeit mittels Thermoelementen

Zusammengefasst kann gesagt werden, dass die Anlage den Funktionstest vollständig bestanden hat und somit betriebsbereit ist.

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

Im Zuge dieser Arbeit wurde eine Versuchsanlage zur Untersuchung der Flammengeschwindigkeit von brennbaren Staub-Luft-Gemischen so erweitert, dass diese einsatzbereit ist. Dafür wurden, der bereits vorhandene Staubeintrag und die Zündung verwendet und erweitert. Des Weiteren wurde eine vorhandene Messtechnik in auf den speziellen Fall angepasst du erweitert. Der Hauptteil dieser Arbeit war die Erstellung der Software für die Versuchsanlage.

Im ersten Schritt wurde der Staubeintrag erweitert, da die Förderrate nicht regelbar und die einzelnen Rührer nicht mit unterschiedlicher Drehzahl betrieben werden konnten. Für die Steuerung der Versuchsanlage wurde ein Arduino Mega 2560 verwendet. Dieser kann nur ein Rechtecksignal mit einer Frequenz ausgeben. Um die Rührer individuell ansteuern zu können, mussten vier individuelle Rechtecksignale erstellt werden. Die bereits vorhandene Hardware war dafür nicht geeignet, und wurde daher erweitert. Auf einer zusätzlichen Platine wurden vier Arduino Nano verbaut. Diese wurden jeweils an ein PWM-Signal angeschlossen, welches vom Arduino Mega generiert wird. Die Nanos transformieren dieses PWM Signal in ein Rechtecksignal, welches abhängig vom PWM-Signal ist. Dadurch kann über den Arduino Mega jeder Rührer individuell gesteuert werden, denn eine Ausgabe von vier individuellen PWM-Signalen ist möglich. Des Weiteren wurden sämtliche Rührer zur genauen Bestimmung der Förderrate auf Wägezellen montiert und Analog-Digital gewandelt. Um die Daten in LabVIEW verwenden zu können, musste die Firmware des Arduino erweitert werden.

Im zweiten Schritt wurde die Software für die Anlage entwickelt. Eine autonome Software ermöglicht eine hohe Reproduzierbarkeit der Versuche. Die Software wurde in LabVIEW erstellt, wobei es sich um eine grafische Programmiersprache handelt. Um die Übersichtlichkeit des Programmes zu erhöhen, besteht das Programm aus mehreren Unterprogrammen. Das Programm ermöglicht einen automatischen Staubeintrag und eine automatische Zündung.

Im dritten Schritt wurden sämtliche Komponenten zusammengefügt und in einem Schaltkasten verbaut, zudem wurde die Stromversorgung für die Versuchsanlage erstellt. Am

Schaltkasten wurde eine Warnleuchte mit eingebauter Sirene angebracht, über welche der Betriebszustand der Anlage ersichtlich ist.

Zuletzt wurde die Anlage am Versuchsstandort einer Funktionsüberprüfung und Inbetriebnahme erfolgreich unterzogen.

5.2 Fazit

Im Zuge dieser Arbeit konnte die Versuchsanlage soweit fertiggestellt werden, dass mit der Anlage Versuche durchgeführt werden können. Die Fertigstellung der Anlage war von vielen Rückschlägen und Herausforderungen begleitet. Die Implementierung der Wägezellen und die veränderte Ansteuerung der Schrittmotoren waren ein großer Schritt. Dadurch konnten die Rührer individuell geregelt werden.

Das Erstellen der Software gestaltete sich als sehr zeitaufwendig und lehrreich. Das Erproben der Software beanspruchte vielleicht sogar noch mehr Zeit. Allein für das Feintuning des Codes, zur Erreichung einer optimalen Förderrate, wurden Stunden verbracht. Mit dem Fortschreiten der Arbeit wurden auch immer mehr Kleinigkeiten in das Programm eingebaut, wie zum Beispiel der Fortschrittsbalken.

Die Konstruktion der Messtechnik gestaltete sich als vergleichsweise einfach. Die bereits vorhandene Messtechnik eines kleinen Rohres konnte zu großen Teilen übernommen und mussten nur leicht angepasst werden.

Mit der Fertigstellung und Zusammenfügung aller Komponenten stieg die Anspannung vor der „warmen“ Testphase, weshalb vor dem Transport zum Standort alle Komponenten unzählige Male „kalt“ getestet wurden. Schließlich konnte im Oktober 2019 der erste Funktionstest erfolgreich abgeschlossen werden und im Sommer 2020 wird voraussichtlich die erste Versuchsreihe mit der Anlage durchgeführt.

5.3 Ausblick

Die nötigen Komponenten für die erfolgreiche Durchführung eines Versuches sind fertiggestellt. Einer Verwendung der Anlage zu Versuchszwecken steht nichts im Wege. Folgende zukünftigen Änderungen werden als sinnvoll erachtet:

- Einbau einer funktionierenden Staubkonzentrationsmessung.

- Vergrößerung des Staubbehälters der Rührer, sodass diese nicht nach jedem zweiten Versuch neu befüllt werden müssen.
- Die Konstruktion eines höhenverstellbaren Einbaues für die Zündung.

6 Literaturverzeichnis

Literaturverzeichnis

[1] Kern, H., Explosible Dust/Air Mixtures, Investigations on flame propagation under non atmospheric conditions, Montanuniversität Leoben, 2013.

[2] Dipl. Ing. Glechner, Entwicklung eines Verfahrens für Staubeintrag und Staubkonzentrationsmessung sowie einer Zündmethode für eine Großversuchsanlage zur Untersuchung der Flammenfortpflanzung in Staub/Luft- Gemischen, Masterarbeit, Montanuniversität Leoben, Leoben, 2018.

[3] Wolfgang Bartknecht, Explosionsschutz, Springer-Verlag, 1993.

[4] Rolf K. Eckhoff, Dust Explosions in the Process Industries, Gulf Professional Publishing, 2003.

[5] Steen, H., Handbuch des Explosionsschutzes, Wiley-VCH, 2000.

[6] Hüttenbrenner, K., H. Kern, M. Stockinger and H. Raupenstrauch, Effects of melting inert particulate additives on ignition and flame propagation in dust/air, Hazards 30 der IChemE 2020 in press..

[7] Frank Bernhard, Handbuch der Technischen Temperaturmessung, Springer-Verlag, 2014.

[8] ABB Automation Products GmbH, Industrielle Temperatur-Messtechnik.

[9] Held Gilbert, Find in library Get price alert Sell this book Share This Page Introduction to Light Emitting Diode Technology and Applications, CRC Press, 2008.

[10] Kurt Reim, LabVIEW-Kurs, Vogel Business Media, 2017.

[11] National Instruments, LabVIEW TM, http://www.ni.com/pdf/manuals/373427j.pdf, abgerufen am 04. September 2019.

[12] National Instruments Corporation, Grundlagen zur LabVIEW-Umgebung, http://www.ni.com/getting-started/labview-basics/d/environment.

[13] National Instruments Corporation, Knoten, VI und Funktionen für Umgebungsvariablen Inhaltsverzeichnis , http://zone.ni.com/reference/de-XX/help/371361R-0113/lvcomm/shared_variable_reference/, abgerufen am 14. Februar 2020.

[14] National Instruments Corporation, Datenstrukturen in LabVIEW, http://www.ni.com/getting-started/labview-basics/d/data-structures, abgerufen am 17. Februar 2020.

[15] Dipl.-Ing- Florian Toth, Aufbau einer Hochgeschwindigkeits- temperaturmessung zur Ermittlung der Flammentemperatur in Staub/Luft Gemischen, Projektarbeit, Montanuniversität Leoben, 2019.

[16] Dipl.-Ing- Florian Toth, Aufbau eines photooptischen Sensorarrays zur Messung der Flammengeschwindigkeit in Staub/Luft - Gemischen, Bachelorarbeit, Montanuniversität Leoben, 2017.

[17] IARC Monographs on the Evaluation of Carcinogenic Risks to Humans, IARC Monographs on the Evaluation of Carcinogenic Risks to Humans, 1997.

7 Anhang

Die Datenblätter des FI-Schutzschalter, des Schütz, der Wägezelle und der Netzgeräte werden dieser Arbeit angehängt.

DATENBLATT - PKNM-16/1N/C/003-A-MW

Abbildung ähnlich

FI/LS, 16A, 30mA, LS-Kennline-C, 1p+N, FI-Char: A

Typ **PKNM-16/1N/C/003-A-MW**
Katalog Nr. **236217**

Powering Business Worldwide

Lieferprogramm

Grundfunktion			Kombinierte RCD/MCB-Geräte
Anzahl der Pole			1 Pol + N
Auslösecharakteristik			C
Anwendung			Schaltgeräte für Anwendungen im Wohnungsbereich und kommerzielle Anwendungen
Bemessungsstrom	I_n	A	16
Bemessungsschaltvermögen nach IEC/EN 61009		kA	10
Bemessungsfehlerstrom	$I_{\Delta N}$	A	0,03
Typ			Typ A
Auslösung		s	unverzögert
Sortiment			PKNM
Empfindlichkeit			Pulsstromempfindlich
Stoßstromfestigkeit			bedingt stoßstromfest 250 A

Technische Daten

Elektrisch

Empfindlichkeit			Pulsstromempfindlich

Daten für Bauartnachweis nach IEC/EN 61439

Technische Daten für Bauartnachweis			
Bemessungsstrom zur Verlustleistungsangabe	I_n	A	16
Verlustleistung pro Pol, stromabhängig	P_{vid}	W	0
Verlustleistung des Betriebsmittels, stromabhängig	P_{vid}	W	3.6
Verlustleistung statisch, stromunabhängig	P_{vs}	W	0
Verlustleistungsabgabevermögen	P_{ve}	W	0
Min. Betriebsumgebungstemperatur		°C	-25
Max. Betriebsumgebungstemperatur		°C	40
			0
Bauartnachweis IEC/EN 61439			
10.2 Festigkeit von Werkstoffen und Teilen			
10.2.2 Korrosionsbeständigkeit			Anforderungen der Produktnorm sind erfüllt.
10.2.3.1 Wärmebeständigkeit von Umhüllung			Anforderungen der Produktnorm sind erfüllt.
10.2.3.2 Widerstandsfähigkeit Isolierstoffe gewöhnliche Wärme			Anforderungen der Produktnorm sind erfüllt.
10.2.3.3 Widerstandsfähigkeit Isolierstoffe außergewöhnliche Wärme			Anforderungen der Produktnorm sind erfüllt.
10.2.4 Beständigkeit gegen UV-Strahlung			Anforderungen der Produktnorm sind erfüllt.
10.2.5 Anheben			Nicht zutreffend, da die gesamte Schaltanlage bewertet werden muss.
10.2.6 Schlagprüfung			Nicht zutreffend, da die gesamte Schaltanlage bewertet werden muss.
10.2.7 Aufschriften			Anforderungen der Produktnorm sind erfüllt.
10.3 Schutzart von Umhüllungen			Nicht zutreffend, da die gesamte Schaltanlage bewertet werden muss.
10.4 Luft- und Kriechstrecken			Anforderungen der Produktnorm sind erfüllt.
10.5 Schutz gegen elektrischen Schlag			Nicht zutreffend, da die gesamte Schaltanlage bewertet werden muss.
10.6 Einbau von Betriebsmitteln			Nicht zutreffend, da die gesamte Schaltanlage bewertet werden muss.
10.7 Innere Stromkreise und Verbindungen			Liegt in der Verantwortung des Schaltanlagenbauers.
10.8 Anschlüsse für von außen eingeführte Leiter			Liegt in der Verantwortung des Schaltanlagenbauers.
10.9 Isolationseigenschaften			

DATENBLATT - PKNM-16/1N/C/003-A-MW

Abbildung ähnlich

FI/LS, 16A, 30mA, LS-Kennline-C, 1p+N, FI-Char: A

Typ	**PKNM-16/1N/C/003-A-MW**
Katalog Nr.	**236217**

Powering Business Worldwide

Lieferprogramm

Grundfunktion			Kombinierte RCD/MCB-Geräte
Anzahl der Pole			1 Pol + N
Auslösecharakteristik			C
Anwendung			Schaltgeräte für Anwendungen im Wohnungsbereich und kommerzielle Anwendungen
Bemessungsstrom	I_n	A	16
Bemessungsschaltvermögen nach IEC/EN 61009		kA	10
Bemessungsfehlerstrom	$I_{\Delta N}$	A	0,03
Typ			Typ A
Auslösung		s	unverzögert
Sortiment			PKNM
Empfindlichkeit			Pulsstromempfindlich
Stoßstromfestigkeit			bedingt stoßstromfest 250 A

Technische Daten

Elektrisch

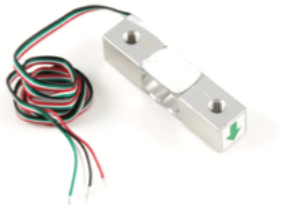
Empfindlichkeit			Pulsstromempfindlich

Daten für Bauartnachweis nach IEC/EN 61439

Technische Daten für Bauartnachweis			
Bemessungsstrom zur Verlustleistungsangabe	I_n	A	16
Verlustleistung pro Pol, stromabhängig	P_{vid}	W	0
Verlustleistung des Betriebsmittels, stromabhängig	P_{vid}	W	3.6
Verlustleistung statisch, stromunabhängig	P_{vs}	W	0
Verlustleistungsabgabevermögen	P_{ve}	W	0
Min. Betriebsumgebungstemperatur		°C	-25
Max. Betriebsumgebungstemperatur		°C	40
			0
Bauartnachweis IEC/EN 61439			
10.2 Festigkeit von Werkstoffen und Teilen			
10.2.2 Korrosionsbeständigkeit			Anforderungen der Produktnorm sind erfüllt.
10.2.3.1 Wärmebeständigkeit von Umhüllung			Anforderungen der Produktnorm sind erfüllt.
10.2.3.2 Widerstandsfähigkeit Isolierstoffe gewöhnliche Wärme			Anforderungen der Produktnorm sind erfüllt.
10.2.3.3 Widerstandsfähigkeit Isolierstoffe außergewöhnliche Wärme			Anforderungen der Produktnorm sind erfüllt.
10.2.4 Beständigkeit gegen UV-Strahlung			Anforderungen der Produktnorm sind erfüllt.
10.2.5 Anheben			Nicht zutreffend, da die gesamte Schaltanlage bewertet werden muss.
10.2.6 Schlagprüfung			Nicht zutreffend, da die gesamte Schaltanlage bewertet werden muss.
10.2.7 Aufschriften			Anforderungen der Produktnorm sind erfüllt.
10.3 Schutzart von Umhüllungen			Nicht zutreffend, da die gesamte Schaltanlage bewertet werden muss.
10.4 Luft- und Kriechstrecken			Anforderungen der Produktnorm sind erfüllt.
10.5 Schutz gegen elektrischen Schlag			Nicht zutreffend, da die gesamte Schaltanlage bewertet werden muss.
10.6 Einbau von Betriebsmitteln			Nicht zutreffend, da die gesamte Schaltanlage bewertet werden muss.
10.7 Innere Stromkreise und Verbindungen			Liegt in der Verantwortung des Schaltanlagenbauers.
10.8 Anschlüsse für von außen eingeführte Leiter			Liegt in der Verantwortung des Schaltanlagenbauers.
10.9 Isolationseigenschaften			

Datasheet

3135 - Micro Load Cell (0-50kg) - CZL635



Contents

What do you have to know?

A load cell is a force sensing module - a carefully designed metal structure, with small elements called strain gauges mounted in precise locations on the structure. Load cells are designed to measure a specific force, and ignore other forces being applied. The electrical signal output by the load cell is very small and requires specialized amplification. Fortunately, **the 1046 PhidgetBridge will perform all the amplification and measurement of the electrical output.**

Load cells are designed to measure force in one direction. They will often measure force in other directions, but the sensor sensitivity will be different, since parts of the load cell operating under compression are now in tension, and vice versa.

How does it work - For curious people

Strain-gauge load cells convert the load acting on them into electrical signals. The measuring is done with very small resistor patterns called strain gauges - effectively small, flexible circuit boards. The gauges are bonded onto a beam or structural member that deforms when weight is applied, in turn deforming the strain-gauge. As the strain gauge is deformed, it's electrical resistance changes in proportion to the load.

The changes to the circuit caused by force is much smaller than the changes caused by variation in temperature. Higher quality load cells cancel out the effects of temperature using two techniques. By matching the expansion rate of the strain gauge to the expansion rate of the metal it's mounted on, undue strain on the gauges can be avoided as the load cell warms up and cools down. The most important method of temperature compensation involves using multiple strain gauges, which all respond to the change in temperature with the same change in resistance. Some load cell designs use gauges which are never subjected to any force, but only serve to counterbalance the temperature effects on the gauges that measuring force. Most designs use 4 strain gauges, some in compression, some under tension, which maximizes the sensitivity of the load cell, and automatically cancels the effect of temperature.

Installation

This Single Point Load Cell is used in small jewelry scales and kitchen scales. It's mounted by bolting down the end of the load cell where the wires are attached, and applying force on the other end **in the direction of the arrow.** Where the force is applied is not critical, as this load cell measures a shearing effect on the beam, not the bending of the beam. If you mount a small platform on the load cell, as would be done in a small scale, this load cell provides accurate readings regardless of the position of the load on the platform.

Calibration

A simple formula is usually used to convert the measured mv/V output from the load cell to the measured force:

*Measured Force = A * Measured mV/V + B* (offset)

It's important to decide what unit your measured force is - grams, kilograms, pounds, etc.

This load cell has a rated output of 1.0±0.15mv/v which corresponds to the sensor's capacity of 50kg.

To find A we use

*Capacity = A * Rated Output*

A = Capacity / Rated Output

A = 50 / 1.0

A = 50

Since the Offset is quite variable between individual load cells, it's necessary to calculate the offset for each sensor. Measure the output of the load cell with no force on it and note the mv/V output measured by the PhidgetBridge.

Offset = 0 - 50 * Measured Output

Product Specifications	
Mechanical	
Housing Material	Aluminum Alloy
Load Cell Type	Strain Gauge
Capacity	50kg
Dimensions	55.25x12.7x12.7mm
Mounting Holes	M5 (Screw Size)
Cable Length	550mm
Cable Size	30 AWG (0.2mm)
Cable - no. of leads	4
Electrical	
Precision	0.05%
Rated Output	1.0±0.15 mv/V
Non-Linearity	0.05% FS
Hysteresis	0.05% FS
Non-Repeatability	0.05% FS
Creep (per 30 minutes)	0.1% FS
Temperature Effect on Zero (per 10°C)	0.05% FS
Temperature Effect on Span (per 10°C)	0.05% FS
Zero Balance	±1.5% FS
Input Impedance	1130±10 Ohm
Output Impedance	1000±10 Ohm
Insulation Resistance (Under 50VDC)	≥5000 MOhm
Excitation Voltage	5 VDC
Compensated Temperature Range	-10 to ~+40°C
Operating Temperature Range	-20 to ~+55°C
Safe Overload	120% Capacity
Ultimate Overload	150% Capacity

Installationschütz J20-20 230

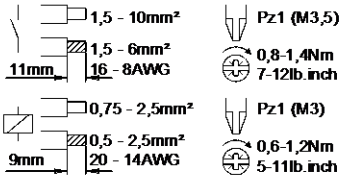
Schaltbild

Gewicht etwa 0,13 kg
Steuerspannung U_s 220-240V 50Hz, 240V 60Hz

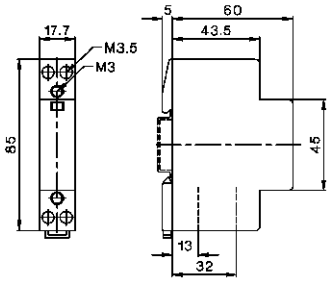
Technische Daten nach IEC/EN60947-4-1, IEC/EN60947-5-1, VDE 0660

Typ			J20..
Hauptschaltglieder			
Bemessungsisolationsspannung U_i		V~	440
Bemessungsbetriebsspannung U_e		V~	440
Zul. Schalthäufigkeit z AC1, AC3		1/h	300
Mech. Lebensdauer		S x 10^6	1
Gebrauchskategorie AC1			
Bemessungsbetriebsstrom I_e (=I_{th}) offen	bei 60°C	A	20
Schaltstücklebensdauer		S x 10^6	0,1
Niedrigste Schaltspannung		V/mA	24/100
Kurzzeitstromfestigkeit 10s-Strom		A	72
Verlustleistung pro Pol bei I_e/AC1		W	2
Gebrauchskategorie AC5b Schalten von Motoren			
Bemessungsleistung Motor	230V	kW	1,1
Gebrauchskategorie AC5a Schalten von Leuchtstofflampen			
1-phasig 1-polig 220-240V		A	kW
unkompensiert		10	1,1
parallelkompensiert		2,3	0,4
Duoschaltung		16	3,0
Gebrauchskategorie AC5b			
Glühlampen		6	1,4
Kurzschlußschutz			
Sicherung Koordinationstyp "1" gL (gG)		A	35
Bemessungskurzschlußstrom	"r"	kA	3
	"Iq"	kA	3
Schaltzeiten bei Steuerspannung U_s ±10%			
	Schließverzug	ms	7 - 16
	Öffnungsverzug	ms	6 - 12
	Lichtbogendauer	ms	10 - 15
Leistung der Magnetspulen			
wechselstrombetätigt	Einschalten	VA	7 - 9
	Halten	VA	2,2 - 4,2
		W	0,8 - 1,6
Arbeitsbereich der Magnetspulen in Vielfachen von U_s	(-40°C bis +40°C)		0,85 - 1,1

Anschlußquerschnitte Schraubendreher, Drehmomente



Maße



07 2006 Änderungen vorbehalten D807-1

MEAN WELL

120W Single Output Industrial DIN RAIL

NDR-120 series

TPTC004 EN62368-1 IEC62368-1

Features

- Universal AC input / Full range
- Protections: Short circuit / Overload / Over voltage / Over temperature
- Cooling by free air convection
- Can be installed on DIN rail TS-35/7.5 or 15
- UL 508 (industrial control equipment) approved
- EN61000-6-2(EN50082-2) industrial immunity level
- 100% full load burn-in test
- 3 years warranty

Applications

- Industrial control system
- Semiconductor fabrication equipment
- Factory automation
- Electro-mechanical apparatus

Description

NDR-120 is one economical slim 120W DIN rail power supply series, adapt to be installed on TS-35/7.5 or TS-35/15 mounting rails. The body is designed 40mm in width, which allows space saving inside the cabinets. The entire series adopts the full range AC input from 90VAC to 264VAC and conforms to EN61000-3-2, the norm the European Union regulates for harmonic current.

NDR-120 is designed with metal housing that enhances the unit's power dissipation. With working efficiency up to 89%, the entire series can operate at the ambient temperature between -20℃ and 70℃ under air convection. It is equipped with constant current mode for over-load protection, fitting various inductive or capacitive applications. The complete protection functions and relevant certificates for industrial control apparatus (UL508, TUV EN62368-1, and etc.) make NDR-120 a very competitive power supply solution for industrial applications.

Model Encoding

NDR - 120 - 12

- 12 — Output voltage
- 120 — Output wattage
- NDR — Series name

File Name:NDR-120-SPEC 2020-01-04

MEAN WELL 120W Single Output Industrial DIN RAIL **NDR-120 series**

SPECIFICATION

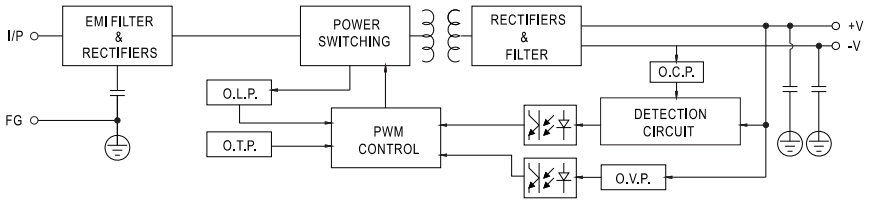
MODEL		NDR-120-12	NDR-120-24	NDR-120-48
OUTPUT	DC VOLTAGE	12V	24V	48V
	RATED CURRENT	10A	5A	2.5A
	CURRENT RANGE	0 ~ 10A	0 ~ 5A	0 ~ 2.5A
	RATED POWER	120W	120W	120W
	RIPPLE & NOISE (max.) Note.2	100mVp-p	120mVp-p	150mVp-p
	VOLTAGE ADJ. RANGE	12 ~ 14V	24 ~ 28V	48 ~ 55V
	VOLTAGE TOLERANCE Note.3	±2.0%	±1.0%	±1.0%
	LINE REGULATION	±0.5%	±0.5%	±0.5%
	LOAD REGULATION	±1.0%	±1.0%	±1.0%
	SETUP, RISE TIME	1200ms, 60ms/230VAC 2500ms, 60ms/115VAC at full load		
	HOLD UP TIME (Typ.)	16ms/230VAC 10ms/115VAC at full load		
INPUT	VOLTAGE RANGE Note.6	90 ~ 264VAC 127 ~ 370VDC [DC input operation possible by connecting AC/L(+), AC/N(-)]		
	FREQUENCY RANGE	47 ~ 63Hz		
	EFFICIENCY (Typ.)	85.5%	88%	89%
	AC CURRENT (Typ.)	2.25A/115VAC 1.3A/230VAC		
	INRUSH CURRENT (Typ.)	20A/115VAC 35A/230VAC		
	LEAKAGE CURRENT	<1mA / 240VAC		
PROTECTION	OVERLOAD	105 ~ 130% rated output power		
		Protection type : Constant current limiting, recovers automatically after fault condition is removed		
	OVER VOLTAGE	14 ~ 17V	29 ~ 33V	56 ~ 65V
		Protection type : Shut down o/p voltage, re-power on to recover		
	OVER TEMPERATURE	Shut down o/p voltage, re-power on to recover		
ENVIRONMENT	WORKING TEMP.	-20 ~ +70℃ (Refer to "Derating Curve")		
	WORKING HUMIDITY	20 ~ 95% RH non-condensing		
	STORAGE TEMP., HUMIDITY	-40 ~ +85℃, 10 ~ 95% RH		
	TEMP. COEFFICIENT	±0.03%/℃ (0 ~ 50℃)		
	VIBRATION	Component:10 ~ 500Hz, 2G 10min./1cycle, 60min. each along X, Y, Z axes; Mounting: Compliance to IEC60068-2-6		
SAFETY & EMC (Note 4)	SAFETY STANDARDS	UL508, TUV EN62368-1, EAC TP TC 004 approved;(meet EN60204-1)		
	WITHSTAND VOLTAGE	I/P-O/P:3KVAC I/P-FG:2KVAC O/P-FG:0.5KVAC		
	ISOLATION RESISTANCE	I/P-O/P, I/P-FG, O/P-FG:>100M Ohms / 500VDC / 25℃/ 70% RH		
	EMC EMISSION	Compliance to EN55032 (CISPR32), EN61204-3 Class B, EN61000-3-2,-3, EAC TP TC 020		
	EMC IMMUNITY	Compliance to EN61000-4-2,3,4,5,6,8,11, EN55024, EN61000-6-2 (EN50082-2), EN61204-3, heavy industry level, criteria A, EAC TP TC 020		
OTHERS	MTBF	456.3K hrs min. MIL-HDBK-217F (25℃)		
	DIMENSION	40*125.2*113.5mm (W*H*D)		
	PACKING	0.6Kg; 20pcs/13Kg/1.16CUFT		
NOTE	1. All parameters NOT specially mentioned are measured at 230VAC input, rated load and 25℃ of ambient temperature. 2. Ripple & noise are measured at 20MHz of bandwidth by using a 12" twisted pair-wire terminated with a 0.1uf & 47uf parallel capacitor. 3. Tolerance : includes set up tolerance, line regulation and load regulation. 4. The power supply is considered a component which will be installed into a final equipment. The final equipment must be re-confirmed that it still meets EMC directives. 5. Installation clearances : 40mm on top, 20mm on the bottom, 5mm on the left and right side are recommended when loaded permanently with full power. In case the adjacent device is a heat source, 15mm clearance is recommended. 6. Derating may be needed under low input voltage. Please check the derating curve for more details. 7. The ambient temperature derating of 3.5℃/1000m with fanless models and of 5℃/1000m with fan models for operating altitude higher than 2000m(6500ft).			

File Name:NDR-120-SPEC 2020-01-04

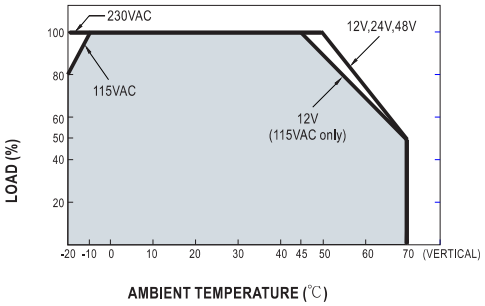
MEAN WELL 120W Single Output Industrial DIN RAIL **NDR-120 series**

■ Block Diagram

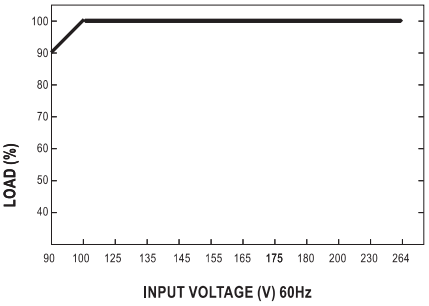
fosc : 70KHz



■ Derating Curve



■ Static Characteristics

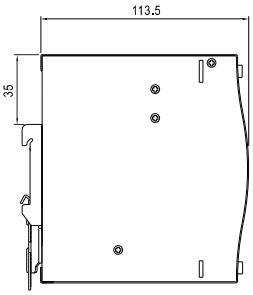


File Name:NDR-120-SPEC 2020-01-04

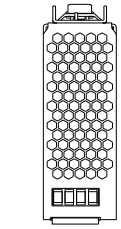
MEAN WELL 120W Single Output Industrial DIN RAIL **NDR-120 series**

■ Mechanical Specification

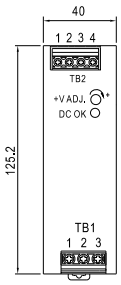
Case No.992D Unit:mm



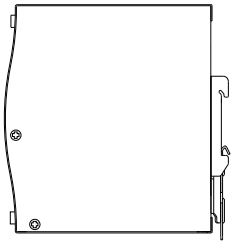
Side View



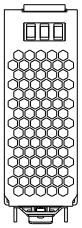
Top View



Front View



Side View



Bottom View

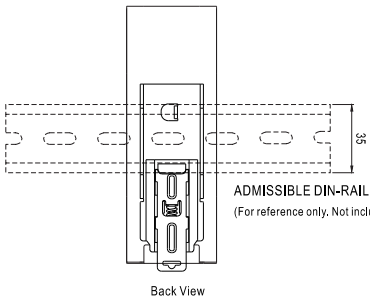
Terminal Pin No. Assignment (TB1)

Pin No.	Assignment
1	FG ⏚
2	AC/N or DC -
3	AC/L or DC +

Terminal Pin No. Assignment (TB2)

Pin No.	Assignment
1,2	DC OUTPUT -V
3,4	DC OUTPUT+V

■ Installation Instruction



ADMISSIBLE DIN-RAIL:TS35/7.5 OR TS35/15
(For reference only. Not included with unit.)

Back View

This series fits DIN rail TS35/7.5 or TS35/15.
For installation details, please refer to the Instruction manual.

■ Installation Manual

Please refer to : http://www.meanwell.com/manual.html

File Name:NDR-120-SPEC 2020-01-04