Bachelorthesis

Auslageexemplar

Name:	Vorname:
von Hebel	**Markus**
Studiengang:	vorgelegt am:
Produktionstechnik	**21.02.2011**
Erstprüfer:	Zweitprüfer:
Prof. Dr.-Ing Peter Chr. Hornberger	**Dipl. Ing. Benjamin Remmers**

Thema:

Erstellung einer an der Messung des Nachlaufs einer hydraulischen Presse erprobten Treiber und Programmdatenbank mit Programmieranleitung

Bachelorthesis

Erstellung einer an der Messung des Nachlaufs einer hydraulischen Presse erprobten Treiber und Programmdatenbank mit Programmieranleitung

Von

Markus von Hebel

Matrikelnummer 1857904

Institut: Hochschule für angewandte Wissenschaften Hamburg

Erstprüfer: Prof. Dr. Peter Chr. Hornberger

Zweitprüfer: Dipl.-Ing. Benjamin Remmers

Vorgelegt am: 21.02.2011

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Formelzeichen und Abkürzungen

Abkürzung	Einheit	Beschreibung
C	mm	Eindringabstand
d	mm	Auflösung eines Lichtvorhangs
ds	mm	Abstand zwischen zwei Messpunkten
dt	ms	zeitlicher Abstand zwischen zwei Messpunkten
ExpressVI		komplexes und konfigurierbares Unterprogramm eines VIs
HBM		Hottinger Baldwin Messtechnik
K	m/s	Annäherungsgeschwindigkeit
S	mm	Sicherheitsabstand
s_1	mm	ältere Stempelposition
s_2	mm	neuere Stempelposition
SubVI		Unterprogramm eines VIs
t_1	ms	älterer Messzeitpunkt
t_2	ms	neuerer Messzeitpunkt
$t_{gerät}$	ms	Geräteschaltzeit
t_{mittel}	ms	mittlere gemessene Schaltzeit
t_{schalt}	ms	Gesamtschaltzeit
v	mm/s	Stempelgeschwindigkeit
v_{max}	mm/s	maximale Stempelgeschwindigkeit
VI		virtuelles Instrument. Bezeichnung für ein LabVIEW-Programm

Kurzreferat

In dieser Arbeit wurden Bausteine, Beispiele und eine Anleitung erstellt, mit denen auch ungelernte Programmierer in der Programmierumgebung LabVIEW 2010 von National Instruments Programme erstellen können, die Sensoren über einen Messverstärker vom Typ Spider8 der Firma Hottinger Baldwin Messtechnik auslesen bzw. Steuerungssignale über die I/O-Buchse des Messverstärker senden und empfangen können. Dazu wurde der LabVIEW-Treiber, der von Hottinger Baldwin Messtechnik angeboten wird, weiterentwickelt.

Diese Baustein-Bibliothek beinhaltet einen Baustein für jede relevante Treiberfunktion und je eine vorgefertigte Sequenz, die aus diesen Treiber-Funktionen erstellt wurde, zur Messung von Einzelwerten für Steuerungsaufgaben oder von einer festgelegten Anzahl von Werten mit hoher Messrate sowie zur Durchführung einer kontinuierlichen Messung mit hoher Messrate, zum Initialisieren des Geräts, zum Setzen und Lesen von Signalen der I/O-Buchse und zum vollständigen Herunterfahren des Geräts.

Zusätzlich wurden auch ein Programm zur Übersetzung von Fehlercodes des Spider8 in Englisch sowie ein Baustein zur Erstellung eines Befehls zur Protokollerstellung ergänzt, das bei der Nutzung von DIAdem 2010 von National Instruments als Auswertungsprogramm relevant ist.

Dazu wurden diese Bausteine getestet und die minimal erreichbare Schaltzeit sowie die maximal erreichbare Messrate ermittelt, die bei der Verwendung dieser Bausteine in der vorliegenden PC-Konfiguration auftreten.

Zusätzlich wurde mit diesen Bausteinen ein Programm zur Messung und Auswertung des Nachlaufs einer Presse erstellt, das nach DIN EN ISO 13855 funktioniert. Die Ergebnisse, die von diesem Messprogramm erstellt worden sind, und die Vorgehensweise, mit der das Programm arbeitet, wurden mit einem bestehenden System zur Nachlaufwegmessung verglichen.

Zusätzlich wurden Untersuchungen mit dem Ziel vorgenommen, die optimalen Parameter für eine Nachlaufwegmessung beim vorliegenden Aufbau zu bestimmen. Diese Parameter sind die Geschwindigkeit, bei der der Not-Aus ausgelöst wird, und die optimale Messrate beim Messen des Weg-Zeit-Verlaufs..

Einleitung

Die Aufnahme und Digitalisierung von Messwerten ist ein signifikanter Bestandteil der Messtechnik. Die Messprogramme, die von Herstellern von A-D-Wandlern oder Messverstärkern erstellt worden sind, bestehen dabei in der Regel aus einem Kompromiss aus Benutzerfreundlichkeit und Einstelloptionen. Dieser Kompromiss kann allerdings nicht allen möglichen Versuchsaufbauten gerecht werden, sodass häufig spezialisierte Software nötig ist. Diese zu erstellen erfordert allerdings ein hohes Maß an Programmier-, Hardware- und Treiberkenntnis, die nicht immer verfügbar oder leicht erwerbbar ist. Um auch weniger spezialisierten Anwendern das Erstellen von solchen Messprogrammen zu ermöglichen, soll hier eine Bibliothek aus Treiberfunktionen und Sequenzen erstellt werden, die eine intuitives Programmieren ermöglicht Dazu wurde die Bibliothek auf den Messverstärker und A-D-Wandler des Typs Spider8, der von der Firma Hottinger Baldwin Messtechnik hergestellt und vertrieben wird, spezialisiert und für die Programmierumgebung LabVIEW 2010 erstellt, dass aufgrund seiner Darstellung von Programmquellcode sowie seiner Programmierphilosophie auch für Anfänger einen schnellen Einstieg ins Programmieren ermöglicht.

1 Einführung in Labview

1.1 Datenfluss-Modell

LabVIEW ist eine Programmierumgebung, die von National Instruments entwickelt wurde. Dabei handelt es sich nicht um eine übliche, Text basierende Programmiersprache, sondern um eine grafische Umgebung, die nach dem Datenfluss-Modell arbeitet. Das bedeutet, dass die Daten von einer oder mehreren Datenquellen über Verarbeitungsblöcke, in LabView Knoten genannt, zu einer oder mehreren Datensenken fließen[1].

Eine Datenquelle kann zum Beispiel ein beliebiges Eingabefeld, eine Konstante, eine Datei oder ein Messgerät sein und liefert einen Datensatz in einem bestimmten Datentyp. Sie ist also mit einer Variablen in einer textbasierten Programmiersprache vergleichbar.

Diese Datenquellen sind über Verbindungslinien, die den Datenfluss und damit den Programmablauf bestimmen, mit den Knoten oder Datensenken verbunden. Dabei ist die Fließrichtung der Verbindungslinie zu beachten, denn Daten fließen von der Quelle zur Senke.

[1] Vgl: Georgi, Wolfgang/Metin, Ergun: Einführung in LabVIEW, 3. Auflage, München, Carl Hanser Verlag, 2007 S. 21, 34f

Die verfügbaren Knoten sind sehr zahlreich und umfassen Logik-Bausteine wie eine Und-Bedingung, mathematische Operationen wie einen Multiplikationsknoten, Programmfunktionen wie einen Dateiaufruf oder selbst erstellte Unterprogramme.

Die Datensenken können Anzeigefelder, Dateien oder andere Knoten sein.

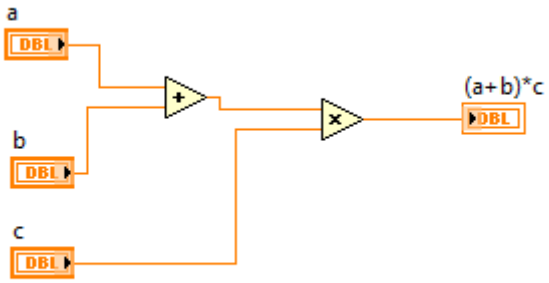


Abbildung 1: Beispiel eines simplen Datenflusses

In Abbildung 1 ist ein simpler Datenfluss dargestellt, der die Daten von den Datenquellen „a“, „b“ und „c“ über zwei Knoten gemäß der Gleichung (1) verarbeitet und an der Datensenke „(a+b)*c“ abgegeben wird.

$$(a+b)\cdot c \qquad (1)$$

Schleifen, Alternativen oder Sequenzen werden in LabVIEW Strukturen genannt. Diese werden als Rahmen dargestellt und können den Datenfluss verändern oder unterbrechen, indem Teile des Datenflusses ersetzt oder mehrfach ausgeführt werden. Zusätzlich können sie den Datenfluss übersichtlicher gestalten, indem nicht benötigte Teile ausgeblendet werden. Um Daten innerhalb der Struktur zu verarbeiten, deren Quelle außerhalb der Struktur liegt, werden in der Regel sog. Tunnel verwenden. Eine Struktur wird erst dann bearbeitet, wenn an allen eingehenden Tunneln Daten vorliegen und Daten werden erst durch einen Tunnel aus der Struktur herausgegeben, wenn die Struktur vollständig abgearbeitet wurde. Im Falle einer Schleife werden die Daten also erst nach dem letzten Schleifendurchgang weitergegeben[2].

Ein wichtiger Unterschied zu einer Text basierenden Programmiersprache ist, dass hier die Reihenfolge, in der die einzelnen Programmbestandteile abgearbeitet werden, nicht durch die Anordnung der Quellen, Senken und Knoten sondern durch deren Verbindung, also dem Datenfluss, vorgegeben wird. Das Prinzip macht das Datenflussmodell sehr verständlich bei der Programmierung

[2] Vgl.: Georgi, Wolfgang/Metin, Ergun: Einführung in LabVIEW, 3. Auflage, München, Carl Hanser Verlag, 2007 S.55ff

von Messprogrammen, denn werden die Messergebnisse erst über Treiberfunktionen geladen, danach an einen Verarbeitungsteil weitergegeben und als letztes angezeigt oder abgespeichert. Aber dieses Prinzip erfordert einen höheren Aufwand, um die einzelnen Programmbestandteile zu synchronisieren. Genau wie eine Struktur wird ein Knoten erst dann verarbeitet, wenn an allen Eingängen Daten angekommen sind und sie geben ihre Daten erst weiter, wenn der Knoten vollständig abgearbeitet wurde. Außerdem werden die eingehenden Daten nur zu Beginn der Ausführung des Knotens abgefragt. Ein Knoten kann also nicht auf geänderte Eingaben reagieren, solange sie über Eingänge an den Knoten gegeben werden[3].

Programme, die mit LabVIEW erstellt worden sind, nennt man virtuelle Instrumente oder auch VI. Alle VIs können auch als Unterprogramm genutzt werden wodurch sie zu sog. SubVIs[4] werden, wodurch der Datenfluss übersichtlicher wird. Jedes VI verfügt über ein Frontpanel, das die Benutzeroberfläche darstellt, und ein Blockdiagramm, das den Programmcode abbildet. Zusätzlich gibt es noch ExpressVIs. Diese können nur als Unterprogramm arbeiten und sind bereits komplexe und häufig kompilierte Programme. Beim Setzen eines solchen VIs wird automatisch ein Konfigurationsmenü geöffnet, indem das Express-VI angepasst werden kann. Ein weiteres Anpassen eines dieser VIs außerhalb dieses Konfigurationsmenüs ist nicht möglich, wenn es bereits kompiliert worden ist, was häufig bei urheberechtlich geschützten Express-VIs der Fall ist. Zudem enthalten sie häufig viel mehr Quellcode als für die Aufgabe benötigt wird, was das Programm langsamer machen kann[5].

[3]Vgl.: Georgi, Wolfgang/Metin, Ergun: Einführung in LabVIEW, 3. Auflage, München, Carl Hanser Verlag, 2007 S. 34

[4] Vgl.: National Instruments Corporation: LabVIEW-Hilfe, National Instruments Corporation, 2010, Glossar

[5] Vgl.: National Instruments Corporation: LabVIEW-Hilfe, National Instruments Corporation, 2010, Express-VIs

1.2 Frontpanel

Im Frontpanel eines VI werden alle Bedien und Steuerelemente sowie alle Anzeigeelemente angeordnet. Es stellt also die Benutzeroberfläche des Programms dar. Dabei stehen eine Vielzahl von Elementen zur Verfügung wie z.B. numerische Eingabe- bzw. Anzeigeelemente oder Schaltflächen. All diese Elemente werden ebenfalls als Datenquelle bzw. Datensenke im Blockdiagramm dargestellt[6].

Zusätzlich können die Elemente im Frontpanel auch Anschlüssen zugeordnet werden, für den Fall, dass das VI als SubVI funktionieren soll[7]. Diese Anschlüsse ermöglichen eine Weitergabe von Daten an die zugeordneten Bedienelemente und damit in das SubVI. Wenn einem Anzeigeelement ein Anschluss zugeordnet wird, werden die Daten aus dem SubVI heraus in das eigentliche VI weitergegeben. Diese Anschlüsse ähneln also Tunneln bei Strukturen. In Abbildung 2 ist ein Beispiel eines Frontpanels zu sehen.

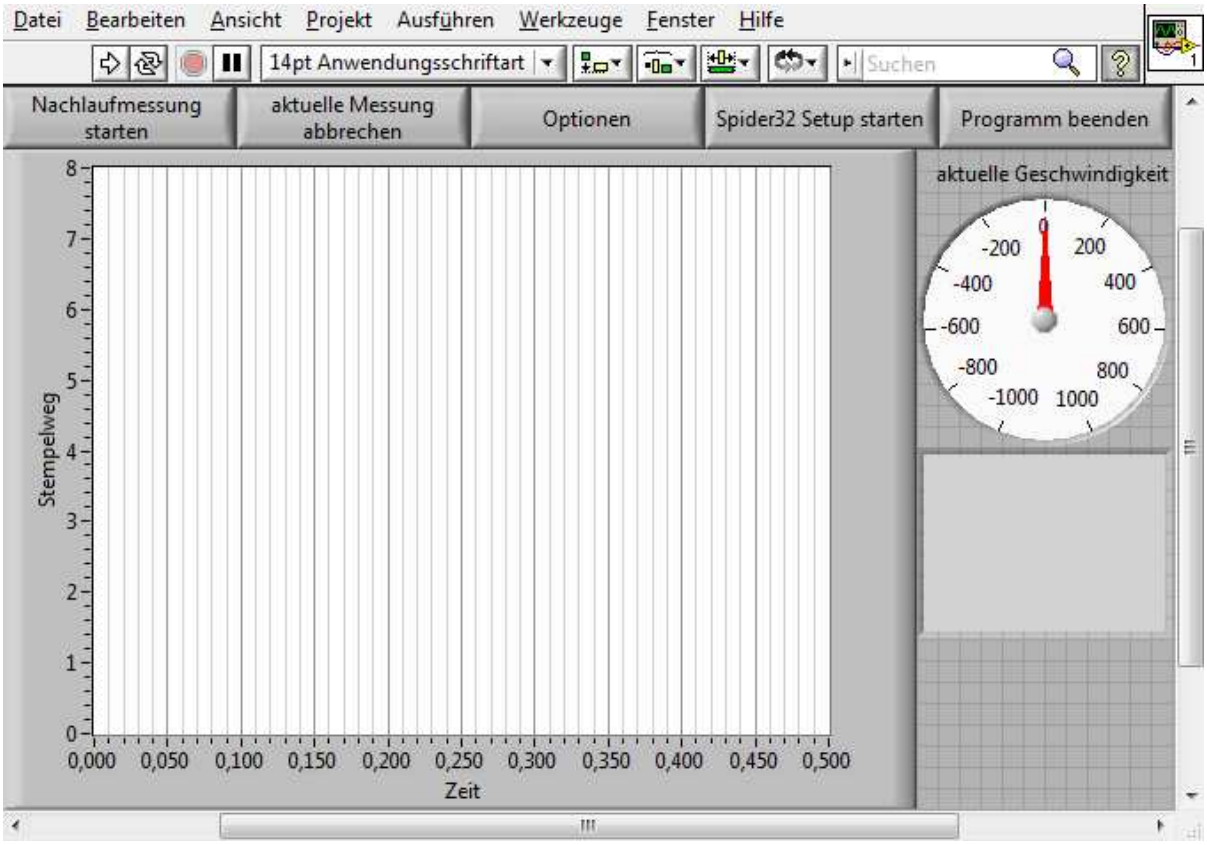


Abbildung 2: Beispiel eines Frontpanels

[6] Vgl.: National Instruments Corporation: LabVIEW Basic I: Einführung/Kurshandbuch, Version 8.6, National Instruments Corporation, 2008, S. 2-14ff

[7] Vgl.: National Instruments Corporation: LabVIEW Basic I: Einführung/Kurshandbuch, Version 8.6, National Instruments Corporation, 2008, S. 7-9

1.3 Blockdiagramm

Das Blockdiagramm entspricht dem Quellcode bei einer auf Text basierenden Programmiersprache. Hier werden die Datenquellen bzw. Datensenken mit Verarbeitungsblöcken verbunden und Strukturen erstellt, wodurch der Programmablauf definiert wird. Dies ist also die Darstellung des Datenflusses des Programms[8]. Obwohl die Annordnung der einzelnen Objekte im Blockdiagramm keine Auswirkung auf die Ausführung haben, empfiehlt es sich den Datenfluss gemäß der Leserichtung, also von links oben nach rechts unten, zu gestalten, um das Blockdiagramm leichter verständlich zu machen. Außerdem sollten Blockdiagramme nach Möglichkeit nicht zu groß werden, damit die Übersicht nicht verloren geht. In Abbildung 3 ist ein Beispiel eines Blockdiagramms dargestellt.

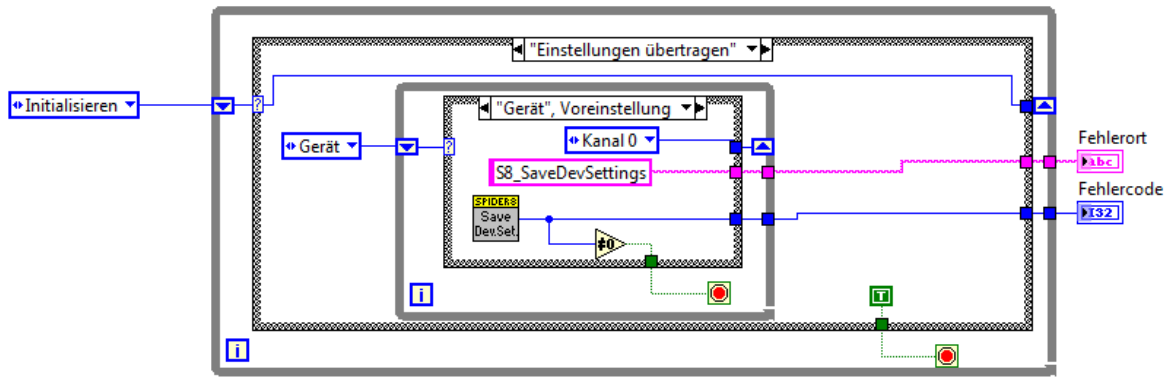


Abbildung 3: Beispiel eines Blockdiagramms

[8] Vgl.: Georgi, Wolfgang/Metin, Ergun: Einführung in LabVIEW, 3. Auflage, München, Carl Hanser Verlag, 2007 S. 33

1.4 In LabVIEW verwendete Datentypen

Der Umgang mit Datentypen funktioniert in LabVIEW etwas anders als in einer auf Text basierenden Programmiersprache. Grundsätzlich gelten zwar dieselben typischen Datentypen also z. B. 32bit long Integer für ganze Zahlen oder 64bit Float für Fließkommazahlen sowie string für Textdaten und „True“ bzw. „False“ für boolesche Werte. Je nach Datenquelle lässt sich der Datentyp auswählen oder er ist bereits vorgegeben. So kann ein numerisches Eingabeelement je nach Einstellung unterschiedliche Datentypen verwenden, aber immer nur Datentypen für numerische Daten. Zudem kann LabVIEW den Datentyp auch automatisch innerhalb bestimmter Grenzen anpassen, sodass der Umgang etwas intuitiver ist. LabVIEW passt z. B. den Datentyp von numerischen Daten automatisch an, sodass Zahlen im Long-Format ohne Probleme mit Zahlen im Float-Format verrechnet werden können. In solchen Fällen wird der genauere Datentyp verwenden[9].

Außer den gängigen Datentypen, kommen in LabVIEW auch einige LabVIEW eigene Datentypen zur Anwendung, wobei hier nur die für Messprogramme wichtigsten genannt werden. Einer dieser Datentypen ist das „Signal“-Format, welches für Messwerte geeignet ist. Hier werden die Y-Daten einer Messung einer Messungsstartzeit und einem konstanten Zeitabstand zwischen den Messungen zugeordnet. Die Startzeit ist dabei im LabVIEW eigenen „Zeitstempelformat“, welcher Datum und Uhrzeit enthält[10].

Auch Pfadangaben zu Dateien verfügen über ein eigenes Format, in dem der Pfad als absoluter Pfad angegeben sein muss. Wenn ein relativer Pfad benötigt wird, ist die Verwendung von Dateikonstanten nötig. Diese Datenquellen geben dann z. B. den Pfad des VI an. Aus diesen kann nun mit einem relativen Pfad ein gültiger absoluter Pfad erstellt werden.

Um mehrere Daten zu einem Datenfluss zusammenzufassen, können entweder Arrays mit beliebiger Dimension oder Cluster verwendet werden. Bei einem Array werden Daten gleichen Typs zusammengefasst, was z. B. bei Messdaten sinnvoll ist. Bei Cluster können Daten beliebigen Typs zusammengefasst werden. Dies kann z. B. bei Messparametern oder Programmeinstellungen sinnvoll sein. Allerdings ist der Umgang mit Clustern nicht so intuitiv wie der Umgang mit Arrays, sodass Arrays bevorzugt werden sollten[11].

---

[9] Vgl.: National Instruments Corporation: LabVIEW Basic I: Einführung/Kurshandbuch, Version 8.6, National Instruments Corporation, 2008, S. 4-10

[10] Vgl.: Georgi, Wolfgang/Metin, Ergun: Einführung in LabVIEW, 3. Auflage, München, Carl Hanser Verlag, 2007 S.140

[11] Vgl.: National Instruments Corporation: LabVIEW Basic I: Einführung/Kurshandbuch, Version 8.6, National Instruments Corporation, 2008, S. 5-2ff

Schließlich findet noch der Datentyp „dynamisch“ Verwendung. Hierbei handelt es sich um einen sehr flexiblen Datentyp, denn er kann Zahlen, Signalverläufe, boolesche Werte oder Arrays enthalten[12]. Da er aber in der Regel von ExpressVIs erzeugt wird, ist der genaue Aufbau dieser Daten häufig unklar.

Der Datentyp Enum, der z. B. für Zustandsautomaten wichtig ist, ist hier ein Sonderfall. In der Literatur wird er häufig als eigener Datentyp aufgelistet, aber er erzeugt numerische Daten im 32Bit Long-Integer-Format. Der Unterschied besteht darin, dass hier kurze Text-Daten aufsteigenden Zahlen zugeordnet werden. So könnte man eine Zuordnung gemäß Abbildung 4 vornehmen. Dort ist z. B. dem Text „Messen“ die Zahl 1 zugeordnet[13].

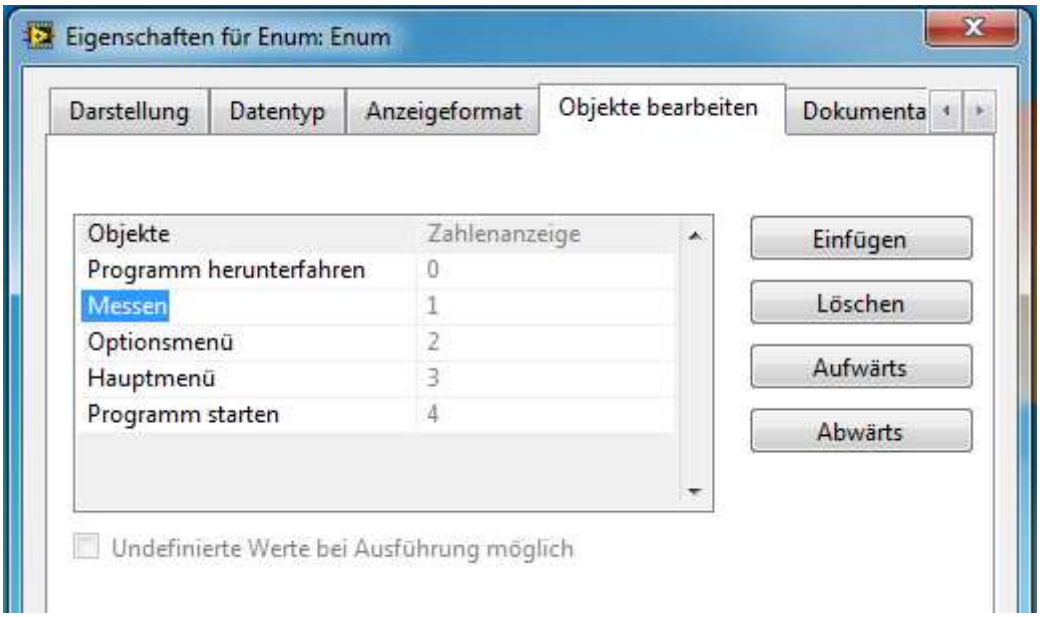


Abbildung 4: Definition eines Enum-Elements

[12] Vgl.: National Instruments Corporation: LabVIEW Basic I: Einführung/Kurshandbuch, Version 8.6, National Instruments Corporation, 2008, S. 4-16

[13] Vgl.: National Instruments Corporation: LabVIEW Basic I: Einführung/Kurshandbuch, Version 8.6, National Instruments Corporation, 2008, S. 4-15

Nach der Definition erscheint das Enum-Element als ein Text-Eingabe- oder Anzeige-Element mit Dropdown-Menü, bei dem je nach Textauswahl die zugeordnete Zahl weitergegeben wird bzw. im Falle eines Anzeigeelements je nach Zahl der zugeordnete Text angezeigt wird. Das ist in Abbildung 5 dargestellt.

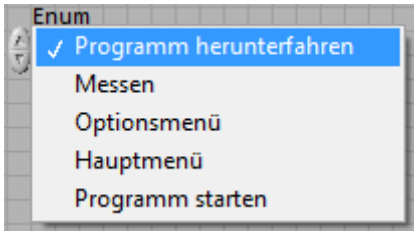


Abbildung 5: DropDown-Menü eines Enum-Elements

Wenn an einer „Case“-Struktur ein Enum-Element als Bedingung angeschlossen wird, erkennt LabVIEW diese und benennt die Fälle gemäß den Text-Daten. Aber auch hier werden die Fälle über numerische Daten ausgewählt. Es ist lediglich eine Vereinfachung für den Programmierer.

Zusätzlich lassen sich auch eigene Datentypen in LabVIEW festlegen die dann LabVIEW-Klassen genannt werden[14].

1.5 Zustandsautomat

Da eine große Herausforderung beim Arbeiten mit dem Datenfluss-Model darin besteht, die Reihenfolge, mit der die einzelnen Programmbestandteile abgearbeitet werden, festzulegen, werden Strategien benötigt, um dies zu gewährleisten. Eine davon ist die Verwendung des „Zustandsautomaten“ auch „Moore-Automat“ genannt. Dabei gibt eine Steuervariable den aktuellen Zustand des Programms an wie z. B. „herunterfahren“ oder „Daten speichern“. Diese Steuervariable, auch Zustandsvariable genannt, bestimmt in einer „Case“-Struktur den auszuführenden Fall. Der Fall führt daraufhin die ihm zugeordneten Programmbestandteile durch und legt den neuen Wert der Zustandsvariablen fest. Dies wird durch eine Schleife bis zum Programmende immer wieder durchgeführt, wodurch je nach Wert der Zustandsvariablen ein anderer Fall ausgeführt wird. Dieser Aufbau schafft Übersicht im Blockdiagramm, denn nur die Blöcke eines Falls werden gleichzeitig dargestellt, und er erlaubt ein sehr schnelles Modifizieren der Sequenz, indem z. B. die Zustandsvariable geändert wird. Außerdem kann so eine nicht statische Sequenz erzeugt werden, die je nach Eingabe oder Ergebnis einen anderen Verlauf hat. Dazu wird die Zufallsvariable von den Ergebnissen oder der Eingabe abhängig gemacht. Ein sehr simpler Zustandsautomat ist in Abbildung

[14] Vgl.: National Instruments Corporation: LabVIEW-Hilfe, National Instruments Corporation, 2010, Verwendung von LabVIEW-Klassen in einer Applikation

6 dargestellt. Das Programm startet mit dem Startzustand und geht dann in den Zustand über, der hier „nächster Zustand“ genannt wird.

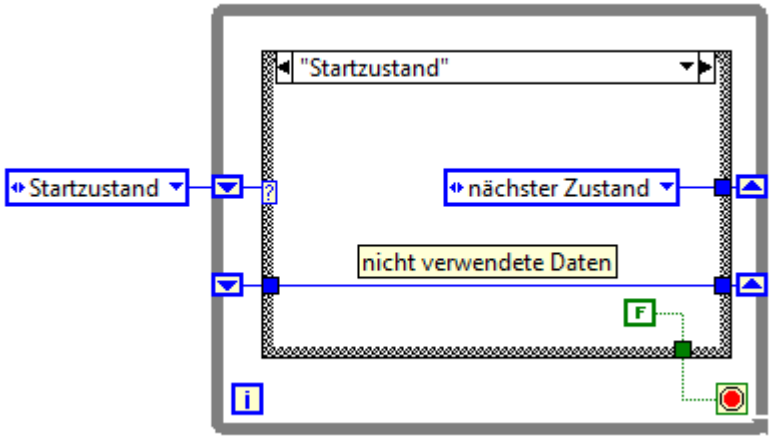


Abbildung 6: Grundsätzlicher Aufbau eines Zustandsautomaten.

In der „Case“-Struktur muss allerdings jedem Fall für jeden Ausgang aus der Schleife und für jeden Wert, der in den nächsten Schleifendurchgang übernommen werden soll, ein definierter Wert angegeben werden, da diese Werte immer eindeutig sein müssen. Dies trifft selbst dann zu, wenn ein Zustand keine passenden Ausgabewerte produziert. Im Beispiel aus Abbildung 6 ist ein solcher Datenfluss aus nicht verwendeten Daten dargestellt. Der Startzustand erzeugt weder diese Daten noch benötigter er diese. Trotzdem muss hier ein definierter Wert übergeben werden. In diesem Fall wird der letzte Wert aus dem Schieberegister an den nächsten Schleifendurchgang übergeben[15].

1.6 Ansprechen von Geräten aus LabView

Grundsätzlich können LabVIEW-Programme auf drei Varianten auf ein Gerät zugreifen, je nachdem wie der Treiber vorliegt. Die für den Anwender einfachste Methode ist, wenn der Treiber als ExpressVI vorliegt. Dann übernimmt dieses VI das Initialisieren, Konfigurieren, Ausmessen und Herunterfahren des Geräts. Allerdings erlaubt es auch nur sehr wenig Kontrolle über den Vorgang, denn die komplette Sequenz ist im ExpressVI vordefiniert. Alternativ können Geräte, die über GPID, USB, Ethernet, PXI, VXI oder einen seriellen Anschluss mit dem PC verbunden sind, auch über VISA angesteuert werden. Dies ist ein Programm, das ebenfalls von National Instrument entwickelt worden ist und von LabVIEW heraus angesprochen werden kann. In VISA wird das entsprechende Gerät eingerichtet und konfiguriert und kann danach in LabVIEW über VISA-VIs angesprochen werden. Wie auch bei der Verwendung von ExpressVIs ist die Kontrolle über den eigentlichen

[15] Vgl.: National Instruments Corporation: LabVIEW Basic I: Einführung/Kurshandbuch, Version 8.6, National Instruments Corporation, 2008, S. 10-5ff

Messvorgang begrenzt. Die letzte Möglichkeit ist die für den Anwender komplizierteste Variante. Hier wird der Treiber direkt angesprochen und jeder Befehl des Treibers ist ein eigener Knoten im VI. Dies ist die Variante mit der z. B. ein Spider8 von HBM angesprochen wird. Das erlaubt vollständige Kontrolle über das Gerät. In der Regel hat man keine Wahl, wie das Gerät angesprochen werden soll, denn der Hersteller des Treibers bestimmt die Variante.[16]

1.7 TDM(S)-Dateien

Das TDM(S)-Dateiformat wurde ebenfalls von National Instruments entwickelt und ist dahingehend optimiert, große Menge an numerischen Daten schnell les- und verarbeitbar zu machen. Dabei bezeichnet die Dateiendung TDMS eine neuere Version des älteren TDM-Format. Beide Formate arbeiten grundsätzlich auf drei Ebenen: der Datensatz-Ebene, der Kanalgruppen-Ebene und der Kanäle-Ebene. Die Datensatz-Ebene ist die oberste Ebene und fasst alle Kanalgruppen der Datei zusammen. Jede Datei enthält nur einen Datensatz, der aber aus mehreren Gruppen bestehen kann, die wiederum ebenfalls aus mehreren Kanälen bestehen können. Kanäle sind Gruppen aus numerischen Werten, wie beispielsweise die Ergebnisse einer Messung. Dabei wird zwischen reinen numerischen Kanälen, die aus Aneinanderreihungen von Zahlen bestehen, und Wellenform-Kanälen, die einen Graphen mit konstantem dt enthalten, unterschieden. Indem man einen numerischen Kanal als den X-Kanal eines Graphen und einen anderen numerischen Kanal als den Y-Kanal eines Graphen definiert, können auch in diesen Kanälen Graphen abgespeichert werden.

Jedem Kanal, jeder Gruppe oder jedem Datensatz könne weitere Eigenschaften zugeordnet werden, die zusätzliche Informationen enthalten, wie den Prüfernamen oder der Messbeginn. All diese Daten werden binär in der Datei gespeichert, wodurch ein schneller Zugriff auf die Daten, selbst bei großer Anzahl an Werten, möglich ist.[17]

[16] Vgl.: National Instruments Corporation: LabVIEW Basic I: Einführung/Kurshandbuch, Version 8.6, National Instruments Corporation, 2008, S. 9-16ff

[17] Vgl.: National Instruments Corporation: LabVIEW-Hilfe, National Instruments Corporation, 2010, The TDM Data Model

2 Einführung in DIAdem

2.1 Navigator

DIAdem ist ein Auswertungsprogramm, welches von National Instruments entwickelt worden ist, mit dem sich Daten manuell oder automatisiert auswerten lassen. Dazu ist das Programm in der Standard-Installation in fünf Module aufgeteilt. Das erste Modul ist der sog. „Navigator“. Hier können eine oder mehrere Dateien mit Messwerten geöffnet werden. Dabei wird im sog. Datenportal auf der rechten Bildschirmseite eine Struktur generiert, die analog ist zur der Struktur einer TDM(S)-Datei. Also enthält das Datenportal auch wieder die Datensatz-Ebene, die Kanalgruppen-Ebene und die Kanäle-Ebene. Sobald die Daten hier geladen worden sind, können sie in den anderen Abschnitten benutzt werden[18]. Das Datenportal ist in Abbildung 7 zu sehen. Dort enthält der Datensatz test2 eine Gruppe mit der Bezeichnung Unbenannt, die wiederum mehrere Kanäle wie z. B. Messung 0 Y enthält.

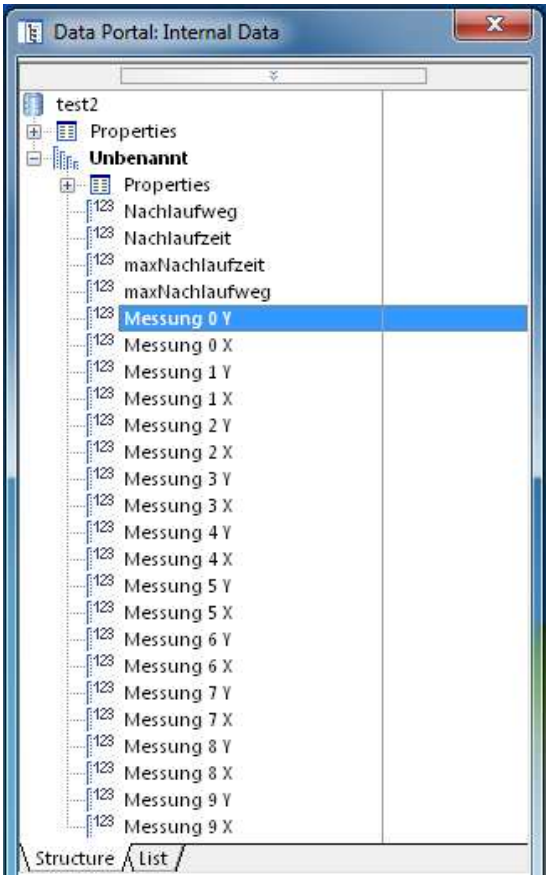


Abbildung 7: Datenportal mit einer geladenen TDMS-Datei

[18] Vgl.: National Instruments Corporation: DIAdem Schnupperkurs/Effizient von Daten zu Ergebnissen, S.16f

2.2 View

Im Modul „View“ kann nun der Inhalt der Kanäle als Graph oder als Tabelle angezeigt werden. Dazu lässt sich die Benutzer-Oberfläche individuell anpassen um z. B. einzelne Graphen anzuzeigen oder nur Ausschnitte. Dabei können nach Bedarf weitere Fenster geöffnet werden und Daten zwischen diesen Festern per „Drag and Drop“ übertragen werden. Auch können hier Messwerte verändert werden oder Kanäle aus vorher markierten Teilen eines anderen Kanals erstellt werden[19].

2.3 Analysis

Das Modul Analysis ermöglicht verschiedene mathematische Operationen und Analysen vorzunehmen. Dies beinhaltet z. B. die Suche nach Maxima oder die Berechnung der Standardabweichung. Auch lassen sich ganze Kanäle miteinander verrechnen. Alle Ergebnisse lassen sich dann als neuer Kanal in der TDM(S)-Datei abspeichern[20].

2.4 Report

Wenn die gewünschte Analyse der Messwerte abgeschlossen ist, kann im Modul „Report“ ein Protokoll erstellt werden, in dem die Messwerte und die Rechenergebnisse in aufbereiteter Form dargestellt werden können. Dieses Protokoll lässt sich entweder als PDF mit eingetragenen Ergebnissen oder als TDR-Vorlage abspeichern. Bei letzterem werden die Namen der Kanäle und Eigenschaften, sofern sie für das Protokoll relevant sind, als Variable abgespeichert, sodass das Protokoll beim Laden von anderen Werten automatisch ausgefüllt wird. Das ermöglicht eine automatische Protokollerstellung. Es werden allerdings keine Rechenoperationen abgespeichert, sodass diese vor dem Laden der Vorlage durchgeführt werden muss[21].

2.5 Script

Das letzte Modul ermöglicht es alle Funktionen von DIAdem zu automatisieren und wird als „Script“ bezeichnet. Hier kann über die textbasierte Programmiersprache DIAdem Script, welche stark mit „Visual Basic Script“ verwandt ist, eine Sequenz zur Auswertung erstellt werden. Dazu steht eine Aufnahme-Funktion zur Verfügung, die die Arbeitsschritte bei einer manuellen Auswertung speichert und in ein DIAdem Script übersetzt. Danach kann diese Sequenz immer wieder zur automatischen Auswertung genutzt werden[22].

---

[19] Vgl.: National Instruments Corporation: DIAdem Schnupperkurs/Effizient von Daten zu Ergebnissen, S. 21

[20] Vgl.: National Instruments Corporation: DIAdem Schnupperkurs/Effizient von Daten zu Ergebnissen, S. 23

[21] Vgl.: National Instruments Corporation: DIAdem Schnupperkurs/Effizient von Daten zu Ergebnissen, S. 24

[22] Vgl.: National Instruments Corporation: DIAdem Schnupperkurs/Effizient von Daten zu Ergebnissen, S. 27

2.6 LabVIEW und DIAdem

Es ist möglich, DIAdem-Funktionen aus LabVIEW heraus aufzurufen und zu starten, um ein VI eine immer gleiche Auswertung von gemessenen Daten durchführen zu lassen. Dazu kann entweder ein Express-VI, das in der Standard-Installation von LabVIEW verfügbar ist, oder das sog. DIAdem Connectivity Toolkit verwendet werden. Letztes ist auf der Internetseite von National Instruments verfügbar und ermöglicht es, jede DIAdem-Funktion in einem LabVIEW VI auszuführen. In beiden Fällen kann ein DIAdem-Script, welches z. B. mithilfe der Aufnahmefunktion erstellt worden ist, gestartet werden und anschließend eine TDR-Vorlage geladen werden, um die Protokollerstellung vollständig zu automatisieren.

3 Nachlaufwegmessung nach DIN EN ISO 13855

Der Aufbau einer berührungslos wirkenden Schutzeinrichtung wird in der DIN EN ISO 13855 beschrieben. Danach wird aus der Nachlaufzeit, einer angenommenen Annährungsgeschwindigkeit sowie dem Aufbau der Schutzeinrichtung und der Auflösung der Sensoren ein minimaler Abstand zwischen Schutzeinrichtung und Gefahrenbereich errechnet. Des geschieht gemäß folgender Formel.

$$S = (K \cdot T) + C \tag{2}$$

Dabei entspricht S dem Sicherheitsabstand, K der Annäherungsgeschwindigkeit, T der Nachlaufzeit des Systems und C dem Eindringabstand.

Für einen Lichtvorhang aus Sensoren mit einer Auflösung von ≤ 40mm, der orthogonal zur Eindringrichtung aufgestellt ist, gelten folgende Annahmen. K beträgt 2000 mm/s und C errechnet sich gemäß dieser Gleichung:

$$C = 8 \cdot (d - 14) \tag{3}$$

Dabei entspricht d der Auflösung des Lichtvorhangs. Bei einem Wert für C von unter 0mm, wird 0mm angenommen.

T ist die Summe aus zwei Komponenten: der Schaltzeit des Lichtvorhangs und der gemessenen Nachlaufzeit des Systems. Bei einer Nachlaufzeitmessung wird ein Notfall simuliert. Dazu wird bei maximaler Stempelgeschwindigkeit der Not-Aus der Maschine betätigt und die Zeit zwischen dem Auslösen des Not-Aus und dem Stillstand des Stempels gemessen. Diese Messung wird zehnmal wiederholt und, je nachdem welcher Wert höher ist, entweder der höchste Wert oder der Mittelwert + drei Standardabweichungen als gemessene Nachlaufzeit in die Formel eingesetzt[23].

[23] Vgl.: Norm DIN EN ISO 13855:2010-10, S. 11f

Sollte der resultierende Sicherheitsabstand größer sein als 500mm, gilt K=1600mm/s und die Rechnung wird mit den neuen Parametern wiederholt. Das Ergebnis wird danach für den Fall, dass es kleiner als 500mm ist, auf 500mm gesetzt[24].

Laut der Berufsgenossenschafts-Regel „Betreiben von Arbeitsmitteln“ Kapitel 2.3 „Pressen der Metallbe- und -verarbeitung“ (BGR 500) ist eine Prüfung der Schutzeinrichtungen einer hydraulischen Presse regelmäßig vorzunehmen[25]. Dabei verweist die BGR 500 auf die ZH 1/281 „Sicherheitsregeln für berührungslos wirkende Schutzeinrichtungen an kraftbetriebenen Pressen der Metallbearbeitung“[26], die ergänzende Sicherheitsregeln zur veralteten VBG 7n 5.2 „Durchführungsanweisungen zur Unfallverhütungsvorschrift Hydraulische Pressen“ enthält. Letztere schreibt für alle hydraulischen Pressen eine automatische Steuereinrichtung vor, die die Steuerung der Presse abschaltet, sobald der Grenzwert des Stößelnachlaufs überschritten wird. Für Pressen, die vor dem 1. April 1987 betreiben wurden, gilt dort als Ausnahmeregelung, dass eine solche Schutzeinrichtung nicht notwendig ist, wenn der Nachlauf halbjährig kontrolliert wird[27].

[24] Vgl.: Norm DIN EN ISO 13855:2010-10, S. 15ff

[25] Vgl.: Norm BGR 500/Kapitel 2.3, 2007, Carl Heymanns Verlag, S. 7

[26] Vgl.: Norm ZH1/281, 2006, Carl Heymanns Verlag, S. 3

[27] Vgl.: Norm VBG 7n5.2, 2005, Carl Heymanns Verlag, S.6 und S. 10

4 Aufgabenstellung

Es ist die Kommunikation zwischen der Programmieroberfläche LabVIEW 2010 und einem Spider8 Messverstärker herzustellen. Dazu ist, neben der Installation aller nötigen Treiber, eine Anleitung und eine dazugehörige VI-Bibliothek zur Erstellung von LabVIEW-Programmen zu erstellen, die einem Benutzer die Steuerung und das Auslesen eines Spider8 Messverstärkers ermöglichen soll und folgende VIs enthalten muss:

- ein geeignetes Hilfsmittel zum Fehlerhandling
- die Erzeugung eines Messwerte-Arrays mit aktuellen Messwertung und dazugehörigen Messzeitpunkten für Steuerungsaufgaben
- die Erzeugung eines Messwerte-Arrays nach Eingabe eines Kanalarray, Messrate und Anzahl an zu messenden Messwerten für nichtkontinuierliche Messaufgaben
- die Erzeugung eines Messwerte-Arrays nach Eingabe eines Kanalarray und Messrate für kontinuierliche Messungen mit hoher Messrate
- Herauslesen der I/O-Buchse und ausgeben der Ergebnisse in einen Cluster
- Setzen der Steuersignale nach Eingabe eines Clusters
- die vollständige Startsequenz für den Spider8-Messverstärker
- die vollständige Herunterfahrsequenz für den Spider8-Messverstärker

Mithilfe dieser VI-Bibliothek ist danach ein Programmbeispiel zur Messung des Nachlaufweges einer hydraulischen Presse nach DIN EN ISO 13855 zu erstellen. Dieses Programm soll nach dem Starten der Presse bei Erreichen einer bestimmten Geschwindigkeit den Not-Aus betätigen und den Nachlauf bestimmen. Diese Messung soll zehnmal wiederholt werden (mit der Option zwischen den Messungen die einzelne Messung zu wiederholen oder den gesamten Versuch abzubrechen). Danach sollen die Ergebnisse in eine TDMS-Datei abgelegt werden und es ist ein automatisches Messprotokoll mit Auswertung zu erstellen. Die Messung und Auswertung ist nach DIN 13855 vorzunehmen.

Dieses Programmbeispiel ist mit dem bestehenden System zur Nachlaufwegmessung zu vergleichen und die Grenzen der mit LabVIEW erstellen Software zu ermitteln. Dabei ergeben sich diese Grenzen aus der maximale Messrate und der minimale Schaltzeit.

5 Versuchsaufbau

5.1 Messkettenbeschreibung

In allen während dieser Arbeit durchgeführten Versuchen wird die Stempelposition einer hydraulischen Presse zu unterschiedlichen Zeitpunkten gemessen und die Messergebnisse weiterverarbeitet um z. B. die Stempelgeschwindigkeit zu messen oder ein Weg-Zeit-Diagramm zu erstellen. Dazu ist an der Presse ein Ortsensor des Typs WayCon MAB-A-A-850-N angebracht. Dieser liefert ein Spannungssignal, das zwischen 0V und 10V liegt und vom Spider8 aufbereitet und digitalisiert wird. Danach werden die Messergebnisse je nach eingestellter Messrate bzw. je nach Messprogramm über eine USB-Verbindung an den PC übertragen und ggf. weiterverarbeitet. Die Messkette ist in Abbildung 8 dargestellt.

Abbildung 8: Messkette zur Ortsmessung des Stempels

Im Falle der Nachlaufwegmessung wird zu diesem Aufbau noch eine Relais-Schaltung ergänzt, die die I/O-Buchse des Messverstärkers ausliest und mit dem Not-Aus-Kreis der Presse verbunden ist. Bei einem entsprechenden Signal wird der Not-Aus-Kreis geschlossen und somit die Presse freigegeben. Um den genauen Schaltzeitpunkt des Relais aufzunehmen, wird gleichzeitig über ein weiteres baugleiches Relais ein externer Trigger ausgelöst, womit die Messung gestartet wird. Der genaue Aufbau der Relais-Schaltung ist unter 8.3 beschrieben.

5.2 Geräte-Liste

5.2.1 WayCon MAB-A-A-850-N

Grundlegend arbeitet der Sensor nach dem Prinzip der Magnetostriktion. Das bedeutet, dass ein ferromagnetischer Körper sich aufgrund einer Wechselwirkung mit magnetischen Feldern verformt. In diesem Fall wird das Magnetfeld innerhalb des Sensors durch ein Kupferrohr erzeugt. Der Messwertgeber besteht hier aus einem Permanentmagneten, dessen Wirklinien sich unter einem Winkel von 90° zu den Wirklinien des vom Kupferrohr erzeugten Magnetfeld befindet. Die dabei entstehenden Wechselwirkungen führen zu einer Verformung, deren Impuls in ein elektrisches Signal umgewandelt wird[28].

5.2.2 Spider8

Der Spider8 ist sowohl ein A-D-Wandler als auch ein Messverstärker und wird von der Firma Hottinger Baldwin Messtechnik hergestellt und vertrieben. Das vorliegende Gerät verfügt über fünf Messkanäle, die durch weitere Module auf bis zu acht Messkanäle erweiterbar sind[29]. Dabei kann je nach Konfiguration eine Voll- bzw. Halbbrückenschaltung, eine Spannungsmessung, eine Frequenz bzw. Periodenmessung oder ein Impulszähler realisiert werden[30]. In der vorliegenden Konfiguration wird ein Dehnungsmessstreifen in einer Brückenschaltung zur Messung des hydraulischen Drucks in der Presse benutzt, um daraus die aktuell wirkende Stempelkraft zu bestimmen. Dieser Sensor befindet sich am Kanal 0. Für diese Arbeit ist allerdings nur der Ortsensor vom Typ WayCon MAB-A-A-850-N relevant, der am Kanal 1 angeschlossen ist und ein Spannungssignal erzeugt. Jeder Kanal lässt sich dabei in 21 Stufen einstellbar mit minimal 1Hz und maximal 9600Hz ausmessen[31]. Zusätzlich wird die I/O-Buchse als Kanal 8 bezeichnet. Diese dient dazu Steuersignale zu senden und zu empfangen und verfügt über acht reine Inputs und acht In/Outputs, die einzeln angesteuert werden können[32].

---

[28] Vgl.:WayCon Positionsmesstechnik GmbH: MAGNETOSTRIKTIV/Magnetostriktiver Wegaufnehmer Serie MAB, www.waycon.de/fileadmin/pdf/Magnetostriktive_Geber_MAB.pdf, 01.02.2011, S. 2

[29] Vgl: Hottinger Baldwin Messtechnik: PC Messelektronik/Spider8 Spider8-30 Spider8-01, Darmstadt, Hottinger Baldwin Messtechnik, S. A-6

[30] Vgl: Hottinger Baldwin Messtechnik: PC Messelektronik/Spider8 Spider8-30 Spider8-01, Darmstadt, Hottinger Baldwin Messtechnik, S. F-1

[31] Vgl: Hottinger Baldwin Messtechnik: PC Messelektronik/Spider8 Spider8-30 Spider8-01, Darmstadt, Hottinger Baldwin Messtechnik, S. F-1

[32] Vgl: Hottinger Baldwin Messtechnik: PC Messelektronik/Spider8 Spider8-30 Spider8-01, Darmstadt, Hottinger Baldwin Messtechnik, S. B-12

5.2.3 USB-Adapter

Der Spider8 verfügt lediglich über eine parallele IEEE-1284 und eine serielle RS-232-C-Schnittstelle zur Verbindung mit dem PC. Da sich allerdings die USB-Schnittstelle durchgesetzt hat, bietet Hottinger Baldwin Messtechnik einen Adapter an, der über die IEEE-1284-Schnittstelle eine USB-Verbindung mit dem PC ermöglicht. Dieser Adapter ist konfigurationslos.

5.2.4 PC

Zur Ausführung aller erstellten Messprogramme sowie LabVIEW 2010 und DIAdem 2010 wird ein herkömmlicher PC verwendet, der über Intel Pentium 4 Prozessor mit 2,66Ghz, einem Arbeitsspeicher von 1,5GB sowie über eine Grafikkarte vom Typ GeForce FX 5200 verfügt. Darauf wird Windows 7 in der 32Bit Enterprise Version ausgeführt. LabVIEW und DIAdem werden in der Version 2010 verwendet.

5.2.5 Hydraulische Presse

Mit dem erstellten Programm wird der Nachlauf einer ölhydraulischen, dreifach wirkenden, 10000t Doppelständer Ziehpresse vom Typ RZU 400 der Firma Lauffer Pressen verwendet.

6 Erstellen der VI-Bibliothek

6.1 Beschreibung des Treibers

6.1.1 Die Treiber-Dateien

Der eigentliche Treiber des Spider8 von Hottinger Baldwin Messtechnik GmbH liegt in kompilierter Form auf der Installations-CD zum Spider8 als folgenden vier Dateien vor:

- Interlink.dll
- Intfac32.dll
- Papo32.dll
- Spider32.dll

Die als Spider8 LabVIEW Treiber veröffentlichen VIs rufen lediglich Befehle auf, die in diesen DLL-Dateien gespeichert sind und werden lediglich in der Version 2.2 für LabVIEW 7.x angeboten, die von der aktuellen Version von LabVIEW nicht geöffnet werden kann. Zudem wird ein Anschließen des Spider8 über USB zwar vom Treiber unterstützt aber die VIs können nicht auf die dafür benötigten Treiberfunktionen zugreifen, da die Eingabe-Optionen fehlen. Um eine Kompatibilität mit LabVIEW 2010 herzustellen, werden die VIs in LabVIEW 8.2.x geöffnet und dort für diese Version gespeichert. Danach können die VIs in LabVIEW 2010 verwendet werden. Um ein Kompilieren der VIs von der LabVIEW Version 8.2.x auf LabVIEW 2010 bei jedem Aufruf der VIs zu vermeiden, werden die VIs für LabVIEW 2010 gespeichert. Damit sind sie allerdings nur mit dieser LabVIEW Version verwendbar.

6.1.2 Strukturvariablen

Die Treiberbefehle benötigen zur Ausführung eigene Variablen, die nur im Arbeitsspeicher vorhanden sind. Diese nennt man Strukturvariablen und ihr Inhalt ist in Tabelle 1 und Tabelle 2 aufgelistet. Dabei werden die Strukturvariablen in Geräteeinstellungen und Kanaleinstellungen unterteilt. Einige Werte werden im sog. IDS-Code in die Variablen eingetragen. Das bedeutet, dass hier jedem möglichen Wert ein spezifischer numerischer Wert zugeordnet wurde, der in der Hilfe-Datei nachgeschlagen werden kann[33]. Bei der Erstellung von Programmen in LabVIEW, stößt man allerdings nicht auf den IDS-Code, denn in der VI-Bibliothek sind VIs vorhanden, die diese Werte automatisch übersetzen und diese „Convert“-VIs werden in den VIs, die auf Treiberfunktionen zugreifen, bereits verwendet.

[33] Vgl.: Hottinger Baldwin Messtechnik: Spider8 DLL Help, Hottinger Baldwin Messtechnik, 2004, Enumeration constants

Tabelle 1: Gerätestrukturvariablen

Variable	Bedeutung
NumChan	Enthält die Anzahl der Kanäle
MeasRate	Enthält die Messrate im IDS-Code
FilterT	Enthält den Filtertyp im IDS-Code
FilterF	Enthält die Filterfrequenz im IDS-Code
SerNum	Enthält die Seriennummer des Spider8
SWVers	Enthält die Versionsnummer der Firmware des Spider8
Name	Enthält den Gerätenamen des Spider8
UserName	Enthält einen benutzerdefinierten Namen für den Spider8
Mode	Enthält den Operationsmodus im IDS-Code

Tabelle 2: Kanalstrukturvariablen

Variable	Bedeutung
ChanType	Enthält die Art des Kanals im IDS-Code
Name	Enthält einen benutzerdefinierten Namen für den Kanal
Unit	Enthält die Einheit in der gemessen wird
IsActive	Enthält den Aktivierungsstatus des Kanals
Mode	Enthält die Art des Sensors (Halbbrücke, etc.) im IDS-Code
MRange	Enthält den Messbereich im IDS-Code
Filter	Enthält die Filterauswahl im IDS-Code
NennVal	Enthält den Nennwert des Kanals
MeasValue	Enthält den zuletzt gemessenen Messwert (nicht bei Kanal 8)
TareValue	Enthält den Tarawert
DigIO	Enthält den Wert der I/O-Buchse (nur bei Kanal 8)

Beim Initialisieren der Verbindung zum Spider8 müssen diese Variablen mit gültigen Werten erstellt werden, da es sonst zu Fehlern beim Aufruf der Treiberfunktionen kommt. Dazu können entweder die aktuellen Einstellungen von Gerät herunter geladen werden, die Einstellungen manuell erzeugt werden oder die Einstellungen aus einer Einstellungsdatei geladen werden. Letzterer Weg ist der empfehlenswerte Weg, da zum Erstellen der Einstellungsdatei das Programm Spider32 Setup, das sich auf der Installations-CD des Spider8 befindet, zur Verfügung steht. Von einem manuellen Erstellen der Strukturvariablen sollte nach Möglichkeit abgesehen werden, da bei einigen Variablen, wie z. B. die Filterfrequenz, in Abhängigkeit von anderen Variablen nicht alle Werte gültig sind, die über die Treiberfunktion eingegeben werden können. Es müsste also eine Überprüfung der Einstellungen ergänzt werden, um Fehler zu vermeiden.

6.1.3 Generelle Sequenz bei der Benutzung des Treibers

Die Treiberfunktionen benötigen zwingend eine bestimmte Sequenz, damit sie wie vorgesehen funktionieren. Diese Sequenz ist in Abbildung 9 dargestellt.

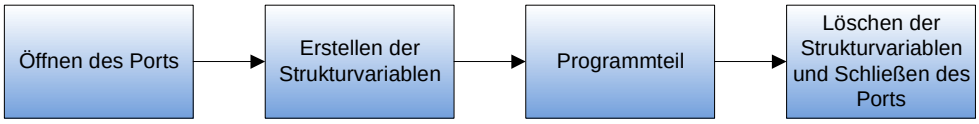


Abbildung 9: Schematische Darstellung der generellen Sequenz

Als erster Schritt muss immer der Port, also die Verbindung zum Spider8, geöffnet werden. Dazu ist je nach Aufbau eine andere Treiberfunktion notwendig. In den meisten Fällen genügt allerdings die Funktion S8_InitAll, die nur über zwei Anschlüsse verfügt und daher sehr einfach konfiguriert werden kann.

Sobald der Port geöffnet wurde, müssen die Strukturvariablen erstellt werden. Dazu gibt es wie unter 7.1.2 beschrieben drei Möglichkeiten. Das Herunterladen der aktuellen Einstellungen, die manuelle Erzeugung der Strukturvariablen und das Laden der Einstellungen aus einer Datei. Wenn die Einstellungen nicht vom Spider8 herunter geladen worden sind, müssen sie danach an das Gerät übertragen werden, was für das Gerät und für jeden Kanal einzeln erfolgt.

Nun kann der eigentliche Programmteil ausgeführt werden. Je nach Messaufgabe ist dieser Teil der Sequenz sehr individuell. Trotzdem gilt für die meisten Messungen, dass zuerst der Messkanal fürs Messen aktiviert werden muss, dann mit den benötigten Messparametern die Messung gestartet wird und als letztes die gemessenen Werte vom Spider8 herunter geladen werden.

Das Löschen der Strukturvariablen und das Schließen des Ports sind extrem wichtige Komponenten der Sequenz, denn wenn diese nicht ausgeführt werden, kann das zu Fehlern in später ausgeführten Programmen führen, obwohl diese vielleicht nicht fehlerhaft sind. Deswegen sollte man auch im Fehlerfall diesen Teil der Sequenz immer ausführen. Dabei werden die Strukturvariablen aus dem Arbeitsspeicher gelöscht und der geöffnete Port wird geschlossen. Wenn bei einem Programm wiederholt Fehler auftreten, deren Ursache nicht bestimmbar ist, kann ein Ausführen dieses Teils der Sequenz das weitere Auftreten der Fehler beenden.

6.2 Auflistung und Erläuterung der modifizierten VIs

Eine Auflistung aller Treiberfunktionen und ihr Syntax ist der Programmieranleitung im Anhang dieser Arbeit zu entnehmen. Generell geben alle Treiberfunktionen des Treibers für den Spider8 immer einen Fehlercode aus, der, wenn kein Fehler aufgetreten ist, 0 beträgt[34]. Aufgrund dieser generellen Regel kann man den Datenfluss der Fehlermeldungen nutzen, um die Reihenfolge zu steuern, in der das Messprogramm ausgeführt wird. Dazu müssen den ursprünglichen VIs mit den Treiberfunktionen und jeder im Rahmen dieser Arbeit erstellten Sequenz ein Fehlereingang, ein Fehlerausgang und eine „Case"-Struktur ergänzt werden. In der „Case"-Struktur sind lediglich zwei Fälle vorhanden. Wenn kein Fehler am Fehlereingang vorliegt, also der Wert 0, wird der Fall „False" aktiviert und die Sequenz oder die Treiberfunktion normal durchgeführt. Danach wird der Fehlercode und der Name der zuletzt ausgeführten Treiberfunktion am Fehlerausgang an das nächste VI weitergegeben. Sollte am Fehlereingang ein Fehler vorliegen, wird der Fall „True" aktiviert und die Treiberfunktion oder die Sequenz wird nicht ausgeführt. Stattdessen wird der am Fehlereingang vorliegende Fehler am Fehlerausgang des VIs zum nächsten VI weitergegeben. Dadurch kann man mit dem Fehlercluster des VIs eine bestimmte Sequenz erstellen, indem die VIs in der Reihenfolge, in der sie abgearbeitet werden sollen, an den Fehlerein- bzw. ausgängen verbindet. Wenn nun irgendwo eine Fehlermeldung auftritt, wird diese durch alle VIs bis zum Ende der Sequenz weitergegeben aber es wird keine weitere Treiberfunktion ausgeführt. Dabei stellen die Funktionen S8_ClosePort und S8_CloseDevice eine Ausnahme dar, da diese VIs so aufgebaut werden, dass immer die Treiberfunktion ausgeführt wird. Trotzdem wird auch hier im Fehlerfall ein Fehler vom Fehlereingang zum Fehlerausgang weitergereicht. In Abbildung 10 ist ein simples Messprogramm dargestellt, das durch den Datenfluss des Fehlercodes gesteuert wird. Diese Modifikation der ursprünglichen VIs wird bei allen VIs in der Bibliothek gemacht.

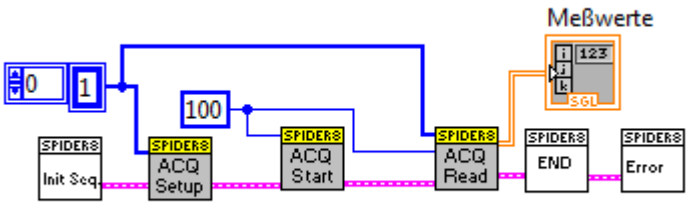


Abbildung 10: Ein simples Messprogramm, das durch den Datenfluss des Fehlers gesteuert wird

[34] Vgl.: Hottinger Baldwin Messtechnik: Spider8 DLL Help, Hottinger Baldwin Messtechnik, 2004, Example for Spider8 initialization and data acquisition

Um nun eine Verbindung über USB zu ermöglichen, müssen die VIs S8_InitAll.vi und S8_OpenPort.vi gemäß der Hilfedatei modifiziert werden[35]. In beiden VIs wird je nach Auswahl am Anschluss „Port“ die Verbindung mit dem Spider8, also z. B. COM1 oder LTP1, ausgewählt und je nach Auswahl ein anderer Wert an den Treiber des Spider8 übergeben. Diese Auswahl geschieht über eine „Case“-Struktur. Beim Anschluss „Mode“, der für den Modus der Kommunikation bei einem parallelen Anschluss und für die Baud-Rate bei einem seriellen Anschluss steht, wird ebenfalls über eine „Case“-Struktur ein Wert ausgewählt und an den Treiber übertragen. Welcher Wert für welchen Anschluss, Modus bzw. Baud-Rate steht, kann der Hilfe-Datei entnommen werden. Im VI geschieht die Auswahl über ein Enum-Element, welches Text-Daten mit numerischen Daten verknüpft. Der Benutzer wählt also über ein Auswahlmenü den Text aus, der für die gewünschte Auswahl steht, das Enum-Element gibt daraufhin eine Zahl aus, die für den Fall in der „Case“-Struktur steht, der mit der Text-Auswahl verknüpft ist und die „Case“-Struktur übergibt eine Zahl an den Treiber, die, gemäß der Hilfedatei, der gewünschten Auswahl entspricht. Dort stellen die mit „Port“ bzw. „Mode“ beschrifteten Datenquellen die Enum-Elemente dar, die Rahmen die „Case“-Strukturen und der orange Knoten die Anbindung des Treibers dar. Die Überschrift an der „Case“-Struktur gibt den gerade ausgewählten Fall an. So wird z. B. für eine Auswahl von COM1 der Wert 1 an den Treiber übergeben.

In den VIs muss daher nur für die Enum-Elemente und in den „Case“-Strukturen jeweils der Fall „USB“ ergänzt werden, der danach mit den für eine Verbindung über USB benötigten Werte gemäß der Hilfe gefüllt wird. Das ist bei Port der Wert 200 und bei Mode der Wert 0[36]. Dies ist in Abbildung 11 dargestellt.

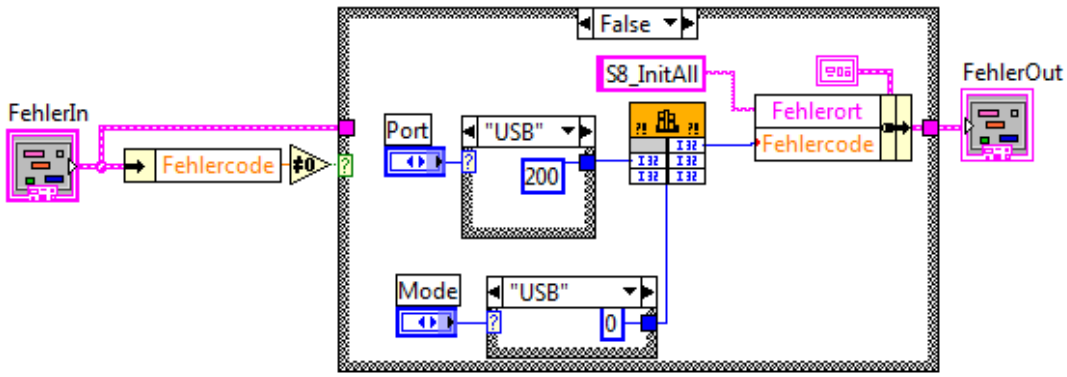


Abbildung 11: Blockdiagramm von S8_InitAll.vi mit USB-Unterstützung

[35] Vgl.: Hottinger Baldwin Messtechnik: Spider8 DLL Help, Hottinger Baldwin Messtechnik, 2004, S8_InitAll, S8_OpenPort

[36] Vgl.: Hottinger Baldwin Messtechnik: Spider8 DLL Help, Hottinger Baldwin Messtechnik, 2004, S8_InitAll, S8_OpenPort

Das VI S8_ACQStart.vi startet eine Messung mit den angeschlossenen Parametern. Dabei war in der ursprünglichen Version eine Eingabe der Messrate nötig. Dies kann aber zu Problemen führen, da beim Ändern der Messrate unter Umständen weitere Geräteeinstellungen, wie die Einstellungen des Filters, automatisch geändert werden[37]. In diesem VI wird über ein Enum-Element aus einer Text-Auswahl eine Zahl erstellt, welche von dem VI „Convert MeasRate to Value“ in einen für den Treiber verwertbare Wert umgewandelt wird. Damit die zuletzt gewählte Einstellung für die Messrate wählbar wird, wird eine „Case“-Struktur ergänzt, die bei entsprechender Auswahl beim Enum-Element, den Wert „-1“ an den Treiber übergibt. Dieser Wert steht für die aktuell eingestellte Messrate. Zudem wird der Anschluss des VIs „MeasRate“ von einem erforderlichen Anschluss zu einem empfohlenen Anschluss geändert und der Wert „aktuelle Messrate“ als Standardwert eingestellt. Dadurch muss der Anwender, wenn er das VI als SubVI verwenden will, diesen Anschluss nicht mehr zwingend belegen. Das modifizierte VI wird in Abbildung 12 dargestellt.

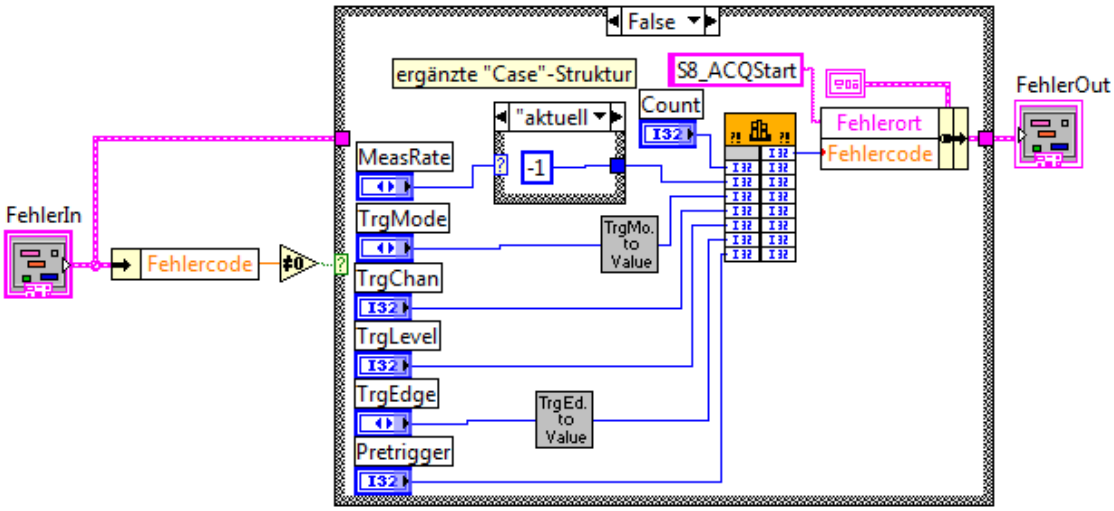


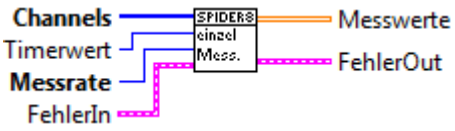
Abbildung 12: Modifiziertes Blockdiagramm von S8_ACQStart.vi

[37] Vgl.: Hottinger Baldwin Mess- und Systemtechnik: Spider8 Setup Hilfe, Version 3.00 041099, Hottinger Baldwin Mess- und Systemtechnik, 1999, Meßrate

6.3 Vorgefertigte Sequenzen und neu erstellte Programme

Zusätzlich zu den nun aktualisierten VIs, wird die Bibliothek um weitere VIs ergänzt, die Sequenzen aus den aktualisierten VIs beinhalten. Diese stehen für häufig benötigte Fälle, wie z. B. das Auslesen der I/O-Buchse. All diesen Sequenzen ist gemeinsam, dass sie sobald ein Fehler auftritt sofort beendet werden und den Fehlercode sowie den Namen der Funktion, die den Fehler verursacht hat, ausgeben. Es folgt eine Auflistung und Beschreibung aller erstellten Sequenzen.

S8_Einzelmessung



Dieses VI misst für eine beliebige Anzahl Kanäle den aktuellen Wert sowie den Zeitpunkt relativ zum Beginn der Messung in ms und gibt diese Werte aus. Die Kanäle werden als 1D-Array aus numerischen Werten im Long-Format am Anschluss „Channels“ angegeben. Um Zeiten in LabVIEW in ms zu messen, stehen die Funktionen der Timing-Palette zur Verfügung. Dort kann über die Funktion „Timerwert“ die seit Programmstart vergangene Zeit in ms abgerufen werden. Diese Funktion ist in der Sequenz verwendet worden, um mit dem „Timerwert“ zum Beginn einer kontinuierlichen Messung und dem „Timerwert“ zum Messzeitpunkt einer Messung die relativen Zeitwerte des einzelnen Messpunkts zu berechnen. Dazu muss „Timerwert“ mit dem Messbeginn am Anschluss „Timerwert“ an das VI übergeben werden.

Mit diesem VI sind auch kontinuierliche Messungen möglich, indem das VI in einer Schleife wiederholt ausgeführt wird. Dabei kann eine Zeitdifferenz in ms angegeben werden, um eine bestimmte Messrate zu erreichen. Das Einhalten der Messrate übernimmt das Programm selbst. Die maximale Messrate ist allerdings durch aktuelle Prozessorlast und maximale Kommunikationsgeschwindigkeit über USB begrenzt und kann bei schnellen Messraten schwanken. Genaueres hierzu kann unter 9.2 nachgelesen werden.

Außerdem erzeugt das Programm pro Schleifendurchgang nur einen aktuellen Satz Werte. Das Speichern der vorherigen Werte muss außerhalb des SubVIs sichergestellt werden.

Die Messwerte werden als 2D-Array herausgegeben, wobei die Spalten für die Kanäle stehen und die letzte Spalte den Messzeitpunkt in ms enthält.

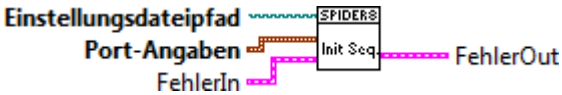
Diese Sequenz wird durch den Fehlercluster gesteuert. Dabei wird zuerst der Messzeitpunkt und direkt danach mit der Funktion S8_MeasOneVal die aktuellen Werte an allen Kanälen gemessen und

in den Strukturvariablen im Arbeitsspeicher abgelegt. Diese Werte werden nun über eine Schleife einzeln aus dem Arbeitsspeicher herausgelesen und in ein 2D-Array abgelegt. Parallel dazu wird die Differenz zwischen dem „Timer“-Wert und dem gemessenen Messzeitpunkt errechnet und als letzten Wert in das 2D-Array abgelegt. Dieses Array wird nun über das Element Messwerte an den entsprechenden Anschluss des VI übergeben.

S8_Herunterfahrsequenz

Diese sehr simple Sequenz beendet die Kommunikation und gibt den vorher benötigten Speicher wieder frei. Es sollte auch im Fehlerfall beim Beenden des Programms ausgeführt werden, da sonst nachfolgende Programme, die auf den Spider8 zugreifen, weitere Fehler erzeugen können. Wenn der Fehlercluster zum Steuern des Programms genutzt wird, werden die Treiberfunktionen innerhalb der Sequenz auch im Fehlerfall ausgeführt. Es besteht nur aus den Treiberfunktionen S8_ClosePort und S8_CloseDevice, die über den Fehlercluster verbunden sind.

S8_InitSequenz



Dieses VI führt alle Schritte aus, die zum Initialisieren des Spider8 notwendig sind. Dazu ist eine *.SP8-Datei mit den Geräten und Kanaleinstellungen notwendig, deren Pfad am Anschluss „Einstellungsdateipfad“ übergeben wird.

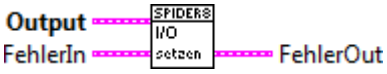
Diese Sequenz wird durch den Fehlercluster gesteuert. Dabei wird zuerst mit S8_InitAll die Verbindung geöffnet, danach mit S8_ReadSettings die Einstellungsdatei ausgelesen und mit S8_SaveDevSettings und S8_SaveChanSettings die Einstellungen für das Gerät und alle Kanäle übertragen.

Die Einstellungsdatei lässt sich entweder aus den sich aktuell im Arbeitsspeicher befindlichen Strukturvariablen mit der Funktion S8_WriteSettings.vi oder aus den aktuellen Einstellungen, die auf den Spider8 gespeichert sind, mit dem Programm Spider32 Setup erstellen. Letzteres ist der komfortablere Weg, da die Einstellungen beim Erstellen der Datei angezeigt werden.

S8_IO_lesen

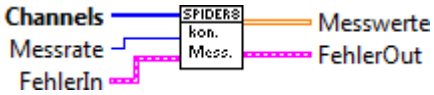
Dieses VI liest den Kanal 8 (I/O-Buchse) aus und wandelt die Signale in einen Cluster aus booleschen Werten um, der am Anschluss „I/O-Signale“ herausgegeben wird. Dazu wird zuerst mit der Funktion S8_MeasOneVal der Kanal ausgelesen und danach mit S8_GetChanSettings die Strukturvariable DigiIO ermittelt. Diese enthält einen numerischen Wert im 32Bit Long-Integer-Format. Dabei geben die ersten 16Bit die acht Inputs und die acht In/Outputs an. Diese Bits werden in boolesche Werte umgewandelt und in ein Cluster an den Anschluss I/O-Signale weitergegeben.

S8_IO_setzen



Dieses VI setzt die Ausgänge des Kanals 8 (I/O-Buchse) gemäß eines Eingangsclusters aus booleschen Werten, der am Anschluss „Output“ übergeben wird. Der Cluster und die darin enthaltenen booleschen Werten werden in einem numerischen Wert im 32Bit Long-Integer-Format umgewandelt. Dabei stehen die ersten acht Bit für die acht In/Outputs der I/O-Buchse des Spider8. Dieser numerische Wert wird nun zuerst in die Strukturvariablen geladen, was mit der Funktion S8_SetChanSettings realisiert wird, und danach mit der Funktion S8_SaveChanSettings an das Gerät übertragen.

S8_kon_Messung

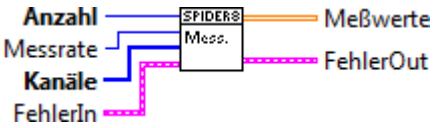


Dieses VI startet eine kontinuierliche Messung mit konstanter Messrate und ohne Begrenzung der Anzahl der Messwerte und ruft die Daten über Datenpakete von je 100 Werten ab, bis die Messung beendet wird. Aufgrund des Datenflusses wird dieses VI erst ausgeführt, wenn an allen Anschlüssen Daten vorliegen und die Daten an diesen Anschlüssen werden nach Start des SubVIs nicht wieder heraus gelesen. Deswegen kann die Schleife, die sich innerhalb des VI befindet, nicht über ein Signal an einem Anschluss des Programms beendet werden. Deswegen muss dieses VI, wenn es als SubVI verwendet wird, modifiziert werden indem z. B. eine sog. globale Variable eingefügt wird. Alternativ kann dieses VI auch als Schablone für eine kontinuierliche Messung dienen.

Die zu messenden Kanäle werden als 1D-Array am Anschluss „Channels“ und die Messrate über das vordefinierte Enum Element MeasRate.ctl am Anschluss „Messrate“ übergeben. Die Verwendung des Elements ist notwendig, da der Spider8 nur bestimmte Messraten unterstützt und so eine Eingabe von ungültigen Werten verhindert wird. Wenn keine Messrate angegeben wird, wird die aktuelle Einstellung des Spider8 benutzt. Die Messwerte werden als 2D-Array im 32Bit-Float-Format am Anschluss „Messwerte“ herausgegeben. Dabei sind die Spalten die Kanäle. In diesem VI ist kein Trigger definiert. Die Messung beginnt unmittelbar nach Aufruf des Programms.

Diese Sequenz wird durch den Fehlercluster gesteuert und stellt den grundsätzlichen Aufbau eines Messvorgangs dar. Zuerst werden die zu messenden Kanäle mit der Funktion S8_ACQSetup aktiviert. Danach wird eine kontinuierliche Messung mit S8_ACQStart gestartet. Die Funktion S8_ACQRead lädt die gemessenen Werte aus und gibt diese als ein 2D-Array an. Dazu wird die Funktion in einer Schleife bis zum Auslösen ihrer Abbruchbedingung ausgeführt. Es ist allerdings keine Abbruchbedingung definiert, damit ein Anwender der VI-Bibliothek dazu gezwungen ist, eine individuelle Bedingung zu setzen.

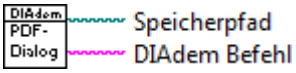
S8_Messung



Dieses VI ähnelt sehr stark S8_kon_Messung.vi. Der einzige Unterschied besteht darin, dass die Angabe einer Anzahl an Messwerten nötig ist. Diese werden dann als ein Datenpaket von S8_ACQRead herunter geladen, wodurch keine Schleife notwendig ist. Daher kann diese Sequenz ohne weitere Modifikation als SubVI verwendet werden.

Zusätzlich zu diesen vordefinierten Sequenzen enthält die Bibliothek noch zwei Hilfs-VIs, die die Programmierung erleichtern sollen. Diese sind DIAdem_Speicherpfad.vi und S8_Fehlerhandler.vi

DIAdem_Speicherpfad



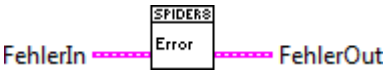
Um die aufgenommenen Messwerte automatisiert auszuwerten, kann das Programm DIAdem benutzt werden. Dazu stehen Express-VIs zur Verfügung, die für die meisten Standardfälle ausreichend sind. Zusätzlich bietet National Instruments eine Bibliothek aus VIs an, die es ermöglichen, jede Funktion von DIAdem aus LabVIEW heraus anzusprechen. Die Bibliothek wird „Diadem Connectivity Toolkit“ genannt.

Um nun ein erstelltes Protokoll im PDF-Format zu speichern, muss ein entsprechender Text-Befehl von LabVIEW an DIAdem übertragen werden. Der dafür nötige String wird von diesem VI erstellt.

Es öffnet einen Dateimonitor, in dem ein Dateipfad für eine PDF-Datei ausgewählt werden kann, und wandelt diesen Dateipfad in einen Befehl für Diadem zur Erstellung eines PDFs um. Der Befehl wird als String am Anschluss „DIAdem Befehl“ und der Dateipfad am Anschluss „Speicherpfad“ ausgegeben.

Ein Express-VI startet einen Dateidialog und gibt den dort angegebenen Pfad heraus. Dieser wird in einen String umgewandelt und in den Befehlsstring eingefügt.

S8_Fehlerhandler



Jede Treiberfunktion erstellt eine Fehlermeldung aus einem numerischen Wert im 32Bit Long-Integer-Format. Wenn kein Fehler vorliegt, wird eine 0 herausgegeben. Diese Fehlercodes können umständlich mittels Hilfedatei übersetzt werden, oder der Fehlercluster wird mit diesem VI verbunden. Es erstellt eine automatische Fehlermeldung mit dem übersetzten Fehlercode und dem Namen der Funktion, die den Fehler verursacht hat. Dazu sind alle in der Hilfedatei erwähnten Fehlercodes als Array aus String-Daten in diesem VI hinterlegt. Wenn ein Fehler vorliegt, wird das Array nach diesem Fehlercode durchsucht und der mit dem Code verknüpfte String als Dialogfeld angegeben. Vorher wird allerdings noch der String mit dem Namen der Treiberfunktion, bei der der Fehler aufgetreten ist, an den String angehängt. Wenn kein passender Eintrag im Array gefunden wird, wird die Meldung „unknown error“ angegeben.

7 Erstellen des Programms zur Nachlaufwegmessung

7.1 Generelle Sequenz einer Nachlaufwegmessung

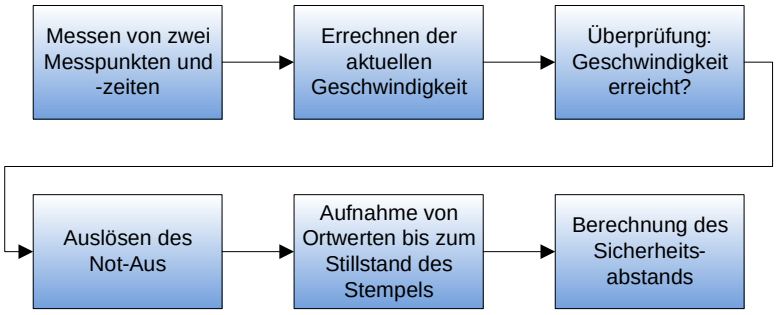


Abbildung 13: Schematische Darstellung einer Nachlaufwegmessung

Die generelle Sequenz einer Nachlaufmessung ist in Abbildung 13 dargestellt. Beim Messen des Nachlaufs einer Presse muss sichergestellt werden, dass der Not-Aus der Maschine erst bei maximaler Stempelgeschwindigkeit ausgelöst wird, um den für den Anwender schlechtesten Fall zu simulieren. Daher sollte die Geschwindigkeit kontinuierlich gemessen werden und erst nach Erreichen der maximalen Geschwindigkeit der Not-Aus betätigt werden. Der verwendete Sensor vom Typ „Waycon MAB-A-A-850-N“ ist allerdings ein Ortsensor, daher beginnt die Sequenz mit dem Messen von zwei Orten und der Berechnung der Geschwindigkeit zwischen diesen Orten. Diese Geschwindigkeit wird nun mit der maximalen Geschwindigkeit verglichen und bei einem positiven Ergebnis wird der Not-Aus ausgelöst. Bei einem negativen Ergebnis wird der nächste Ort aufgenommen und wieder eine Geschwindigkeit errechnet, die dann mit der maximalen Geschwindigkeit der Presse verglichen wird. Dies wird solange fortgesetzt, bis die maximale Geschwindigkeit erreicht ist.

Danach werden weiter Messwerte aufgenommen, bis der Stempel der Presse zum Stillstand gekommen ist. Diese Messwerte werden gemäß Kapitel 4 ausgewertet, um einen Sicherheitsabstand der Sicherheitseinrichtung zum Gefahrenbereich zu bestimmen.

7.2 Vereinfachter Programmablaufplan

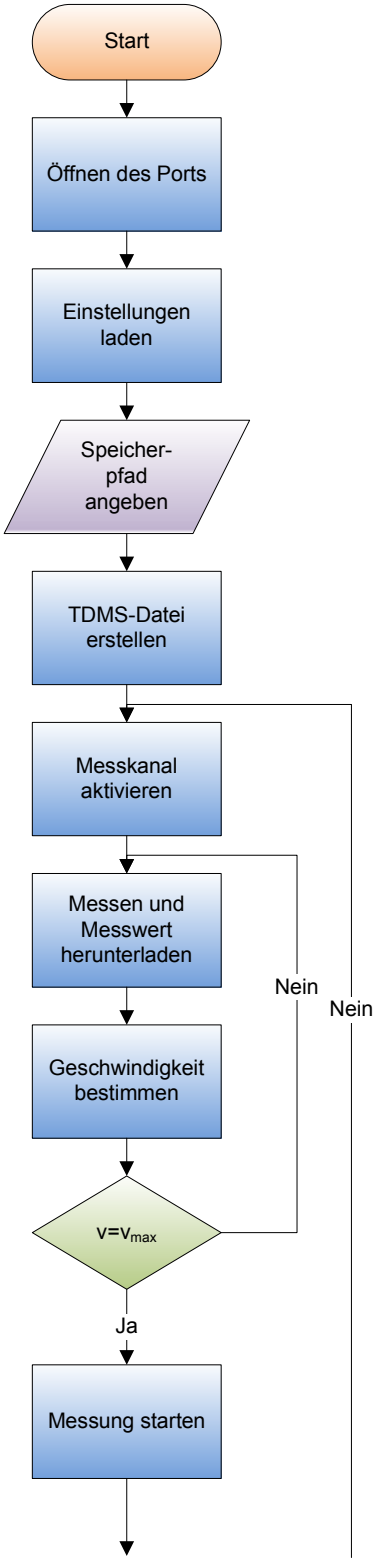


Abbildung 14: Programmablaufplan Teil 1

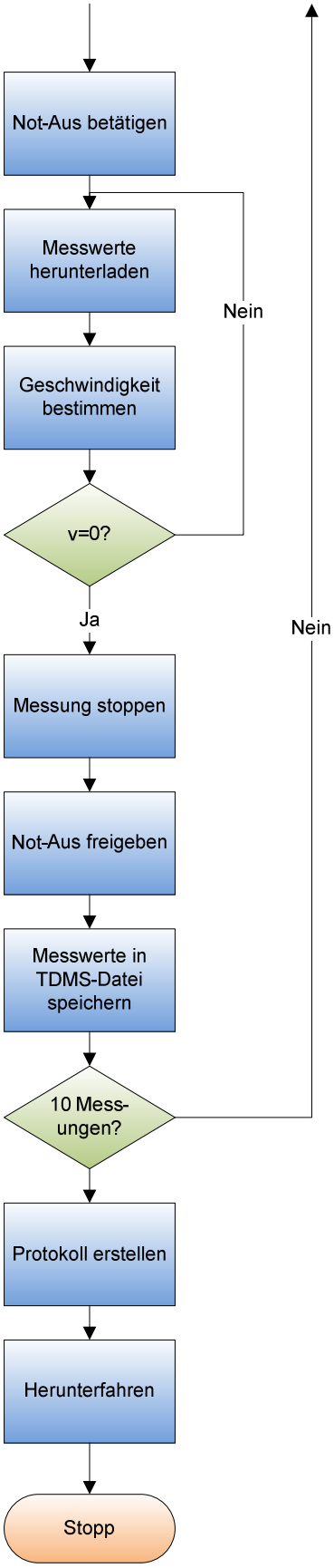


Abbildung 15: Programmablaufplan Teil 2

Der Programmablaufplan in Abbildung 14 und 15 zeigt einen vereinfachten Programmablauf, bei dem Optionsmenüs und Eingabemenüs sowie Unterprogramme und Fehlerhandling unberücksichtigt bleiben. Das eigentliche Programm besteht aus einem Zustandsautomaten, bei dem jede Operation einem Zustand entspricht. Dabei bestehen häufig Zustände aus lediglich einem SubVI. Wenn eine Treiberfunktion einen Fehler erzeugt, wird jedes Mal das SubVI S8_Fehlerhandler.vi ausgeführt, um eine entsprechende Fehlermeldung zu erzeugen, und der Zustand „Herunterfahren“ aufgerufen, womit das Programm beendet wird. Die Operation „Port öffnen“ heißt im Zustandsautomat „Programm initialisieren“. Dort wird die Treiberfunktion S8_InitAll.vi mit Konstanten für eine USB-Verbindung ausgeführt.

Der nächste Zustand wird „Einstellungen laden“ bezeichnet. Hier wird die config.ini ausgelesen, die die zuletzt abgespeicherten Programmeinstellungen enthält. Die geschieht über einen weiteren Zustandsautomat, bei dem jede Programmeinstellung einem Zustand entspricht. Dies ist in Abbildung 16 dargestellt. Hier wird die Text-Datei ausgelesen und der String in die gesuchten Einstellungen zerlegt. Diese werden dann in den benötigten Datentyp umgewandelt und am Ende der Schleife des Zustandsautomaten ausgegeben.

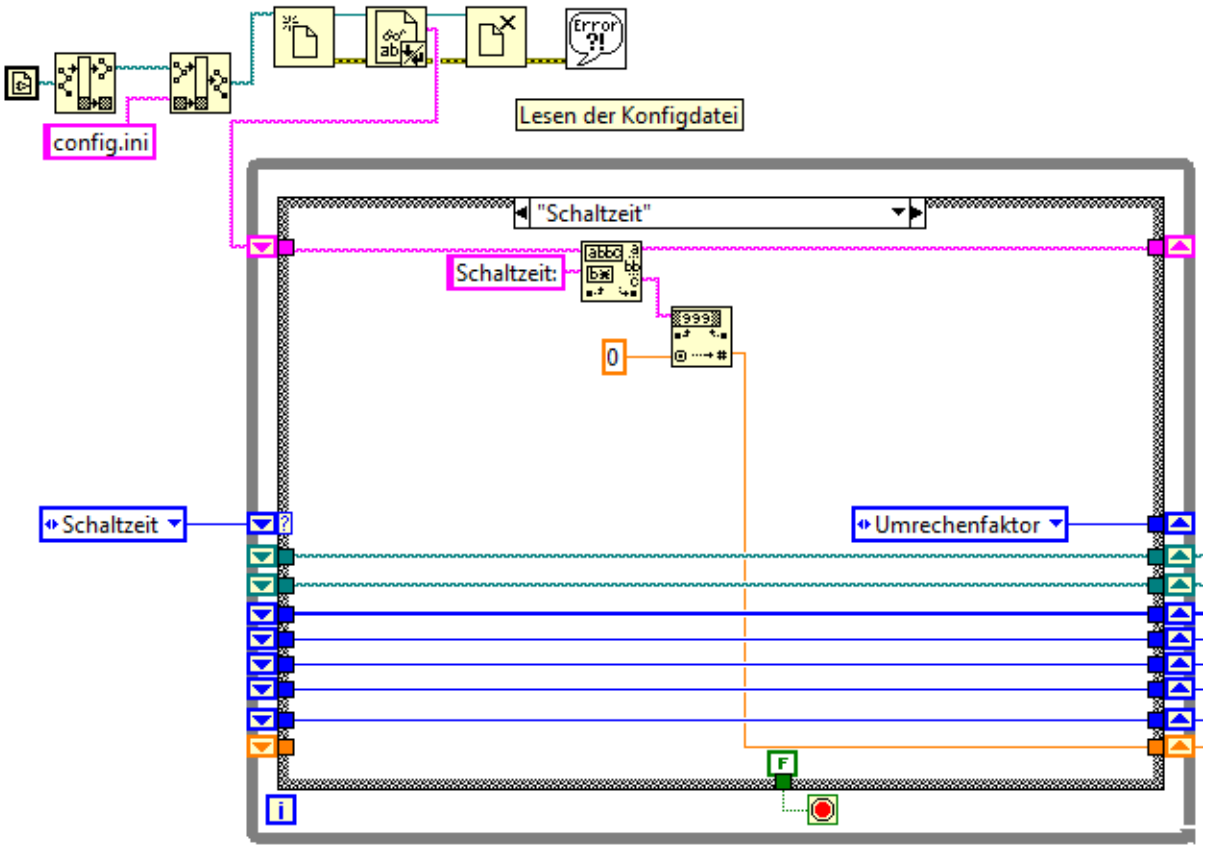


Abbildung 16: Auslesen der Kanaleinstellungen

Eine dieser Programmeinstellungen enthält den Pfad zur SP8-Datei, die die gewünschten Geräte- und Kanaleinstellungen enthält. Aus dieser Datei werden mit der Treiberfunktion S8_ReadSettings.vi die Strukturvariablen erzeugt und zuletzt werden die Strukturvariablen mit den Funktionen S8_SaveDevSettings.vi für Geräteeinstellungen und S8_SaveChanSettings.vi für Kanaleinstellungen an den Spider8 übertragen. Das Übertragen der Kanaleinstellungen muss allerdings für jeden Kanal einzeln erfolgen. Daher ist hier eine Schleife notwendig. Zusätzlich wird hier auch der Not-Aus der Maschine freigegeben. Das geschieht über eine vom Fehlercluster gesteuerte Sequenz, die in Abbildung 17 dargestellt ist.

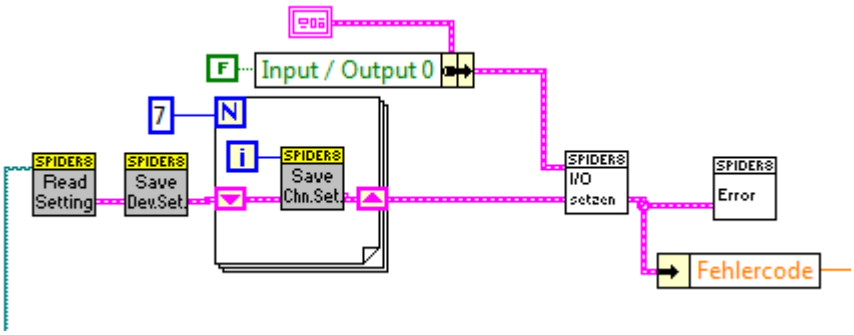


Abbildung 17: Übertragen der Einstellungen an den Spider8

Sobald diese Operationen durchgeführt worden sind, geht das Programm in den Zustand „Ruhemodus“. Dieser Zustand wird solange wiederholt ausgeführt bis eine der Optionen in der Bedienleiste des Programms ausgewählt wurde. Der Button „Optionen“ startet dabei das SubVI Optionsmenü.vi, das die oben erwähnte config.ini erstellt oder modifiziert. Es startet ein Eingabemenü, das mit den letzten Werten, also den Programmeinstellungen, gefüllt wird. Diese Werte können nun geändert werden. Beim Schließen des Menüs werden die Angaben abgespeichert und der Zustand „Einstellungen laden“ wird erneut ausgeführt, um die geladenen Einstellungen zu aktualisieren. Der Button „Spider32 Setup starten“ ruft einen Systembefehl auf, der die EXE-Datei von Spider32 Setup startet. Dies ist sowohl vom Hauptfenster als auch aus dem Optionsmenü möglich. Sowohl das Optionsmenü als auch das Ausführen von Spider32 Setup sind jeweils eigene Zustände im Zustandsautomaten und führen nach ihrer Ausführung zurück zum Zustand „Einstellungen laden“. Sie dienen der Konfiguration des Programms.

Die eigentliche Messung wird durch den Zustand „Speicherpfad bestimmen“ gestartet, welcher durch den Button „Nachlaufwegmessung starten“ aufgerufen wird. Hier wird ein ExpressVI aufgerufen, das einen Dateidialog startet, indem nun der Speicherort der TDMS-Datei eingegeben wird, in der die Messergebnisse abgelegt werden sollen. Darauf folgt der Zustand „TDMS-Datei erstellen“. Hier wird das SubVI Eingabemenü.vi gestartet, was für den Anwender ein Eingabemenü öffnet, in dem er für die Messung relevante Angaben macht. Zusätzlich wird die TDMS-Datei, die später fürs Speichern der

Daten benutzt wird, mit den Kanälen „Nachlaufweg“, „Nachlaufzeit“, „maxNachlaufzeit“ und „maxNachlaufweg“ erstellt. Diese Kanäle werden im späteren Programmverlauf für die Auswertung benötigt. Die getroffenen Angaben werden als Eigenschaften des Kanals „Nachlaufzeit“ in der TDMS-Datei gespeichert.

Die Operation Messkanal aktivieren wird im Zustand „Kanäle aktivieren“ durchgeführt. Dazu wird die Treiberfunktion S8_ACQSetup.vi aufgerufen. Zusätzlich wird hier ein Dialogfenster geöffnet, das den Anwender daran erinnert den Not-Aus an der Maschine freizugeben und den Stempel in Ausgangsposition zu fahren.

Im Zustand „Steuern: Werte in Array“ wird die Sequenz S8_Einzelmessung ausgeführt. Diese lädt den aktuellen Wert des Messkanals herunter und fügt ihn in ein Array ein. Das Programm speichert die letzten fünf Werte um eine Mittelwertsglättung durchzuführen.

Aus dem ersten und dem letzten gespeicherten Wert wird nun die durchschnittliche Geschwindigkeit gemäß folgender Formel errechnet.

$$v = \frac{s_2 - s_1}{t_2 - t_1} \tag{4}$$

Dabei steht s_2 für den letzten Ort, s_1 für den ältesten gespeicherten Ort, t_2 für den Zeitpunkt der aktuellsten Messung, t_1 für den Zeitpunkt der ältesten gespeicherten Messung und v für die Geschwindigkeit. Diese wird nun von V/ms auf mm/s umgerechnet und im Element „aktuelle Geschwindigkeit“ dargestellt. Auch die Überprüfung, ob die maximale Geschwindigkeit erreicht worden ist, findet hier statt. Je nachdem wird als nachfolgenden Zustand der Zustand „Steuern: Werte in Array“ oder „Messen: Messung starten“ gestartet. Sollte nur ein Wertepaar zur Geschwindigkeitsberechnung vorliegen, wird die Überprüfung der Geschwindigkeit ignoriert und immer der Zustand „Steuern: Werte in Array“ aufgerufen. Zusätzlich wird hier der Button „aktuelle Messung abbrechen“ ausgelesen und bei Betätigung der „Ruhemodus“ aufgerufen.

Der Zustand „Messen: Messung starten“ führt die Treiberfunktion „S8_ACQStart.vi“ aus. Dabei wird durch die Konstanten ein externer Trigger definiert, der die Messung bei entsprechendem Signal an der I/O-Buchse startet. Hier wird ein Signal an die I/O-Buchse übertragen, wodurch ein Relais den Not-Aus auslöst und ein weiteres baugleiches Relais ein Startsignal für den Trigger an die I/O-Buchse durchschaltet. Das Blockdiagramm ist in Abbildung 18 dargestellt.

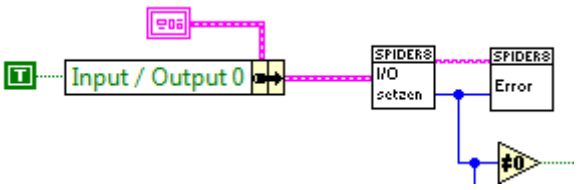


Abbildung 18: Blockdiagramm des Zustands "Not-Aus auslösen"

Die Messwerte werden als Datenpakete herunter geladen, wobei die Größe der Datenpakete im Optionsmenü des Programms festgelegt werden kann. Das Herunterladen geschieht im Zustand „Messen: Werte in Array“ über die Treiberfunktion „S8_ACQRead.vi“. Jedes Datenpaket wird dabei an ein Array aus den bisher aufgenommenen Werten angehängt. Um festzustellen, ob die Maschine zum Stillstand gekommen ist, wird die Differenz zwischen dem ersten Wert vom aktuellen und vom vorherigen Datenpaket errechnet. Sollte sie kleiner oder gleich 0 sein, geht das Programm zum Zustand „Messen: Messung stoppen“ über. Wenn die Differenz größer als 0 ist, oder nur ein Datenpaket bisher herunter geladen wurde, wird das nächste Datenpaket herunter geladen. Aus dieser Differenz und der aktuellen Messrate, wird eine durchschnittliche Geschwindigkeit ermittelt und im Element „aktuelle Geschwindigkeit“ auf dem Frontpanel angezeigt. Ansonsten wird die Geschwindigkeit nicht weiter verwendet. Auch hier wird überprüft, ob der Button „aktuelle Messung abbrechen“ betätigt wurde und ggf. wird die Messung beendet und das Programm wird in den „Ruhemodus“ gesetzt.

Da im Zustand „Messen: Messung starten“ eine kontinuierliche Messung gestartet wurde, muss diese über die Treiberfunktion „S8_ACQStopp.vi“ beendet werden. Die wird im Zustand „Messen: Messung stoppen“ durchgeführt. Des Weiteren muss der Graph aus den Messwerten über ein eindeutiges Maximum verfügen, damit während der automatischen Auswertung kein Fehler auftritt. Um dies zu gewährleisten, wird als letzter Messwert dem Messwerte-Array eine 0 angehängt. Dies wird im Zustand „Messen: 0 anhängen“ durchgeführt. Daraufhin wird im Zustand „Not-Aus freigeben“ für die nächste Messung wieder ein Signal an das Relais übertragen, wodurch der Not-Aus freigegeben wird.

Für den Fall, dass die Messung fehlerhaft verlaufen ist, wird das Array in den Datentyp Signal umgewandelt und im Graphenfenster als Weg-Zeit-Diagramm dargestellt. Dazu sind Daten für dt und ein Startzeitpunkt notwendig. Das dt wird aus der aktuellen Messrate errechnet und der Startzeitpunkt liegt konstant bei 0. Der Anwender kann nun anhand des Kraft-Weg-Diagramms beurteilen ob die Messung erfolgreich war oder nicht. Nach der Darstellung wird ein Eingabedialog geöffnet, über den er die Messung wiederholen, abbrechen oder zur Auswertung übergehen kann. Im Programm bestimmt der Dialog den nächsten Zustand des Programms. Eine fehlerhafte Messung wird wiederholt, was bedeutet, dass der nächste Zustand „Kanäle aktivieren“ ist. Bei korrekter

Messung wird der Zustand „Ergebnisse in TDMS-Datei“ aufgerufen. Zusätzlich kann der Anwender durch das Wählen der „Abbrechen“-Option das Programm an dieser Stelle beenden. In diesem Fall würde der Zustand „Herunterfahren“ ausgewählt.

Im Zustand „Ergebnise in TDMS-Datei“ wird nun je ein Kanal für die X- bzw. Y-Werte in der TDMS-Datei erzeugt. Die X-Werte ergeben sich aus der aktuellen Messrate und der Größe des Messwerte-Arrays. Die Y-Werte sind die Messwerte. Je nachdem ob in dem Optionsmenü ein Kurzprotokoll oder ein vollständiges Protokoll nach DIN EN ISO 13855 gewählt wurde, wird nun überprüft, ob eine oder zehn Messungen durchgeführt werden sollen und ob die gewünschte Anzahl an Messzyklen bereits durchgeführt wurde. Sollten alle Messungen durchgeführt worden sein, wird der Zustand „Diademskript“ aufgerufen. Ansonsten wird der nächste Messzyklus gestartet, der mit dem Zustand „Kanäle aktivieren“ beginnt.

Da nun alle Daten zur Bestimmung des Nachlaufs der Maschine in der TDMS-Datei gespeichert worden sind, kann die Auswertung durchgeführt werden. Diese wird nicht durch ein LabVIEW VI durchgeführt, sondern das Programm ruft DIAdem Funktionen auf. Dazu werden zuerst die TDMS-Datei als DIAdem-Referenz geöffnet, danach ein DIAdem-Script durchgeführt, die Ergebnisse in eine Protokoll-Vorlage geladen und diese als PDF-Datei abgespeichert. Schließlich wird die DIAdem-Referenz wieder geschlossen. Jede dieser Funktionen besteht aus einem eigenen VI, das im DIAdem Connectivity Toolkit von der National Instruments Homepage herunter geladen werden können.

Das DIAdem Script, führt eine Sequenz aus Berechnungen durch, die mit der Bestimmung der Maxima der verschiedenen Nachlaufzeit-Messungen beginnt. Dabei werden die Koordinaten von jedem Maximum in einem neuen Kanal gespeichert. Die X-Koordinate des Maximums entspricht hier der Nachlaufzeit. Diese gemessenen Nachlaufzeiten werden in dem Kanal „Nachlaufzeit“ gespeichert. Aus dem Y-Wert beim Starten der Messung und der Y-Koordinate des Maximums von jeder Messung wird der Nachlaufweg bestimmt und im Kanal „Nachlaufweg“ gespeichert. In den Kanälen „maxNachlaufzeit“ und „maxNachlaufweg“ wird nun die maximale Nachlaufzeit und der Nachlaufweg, der bei dieser Nachlaufzeitmessung auftrat, gespeichert. Danach werden der arithmetische Mittelwert und die Standardabweichung aus allen gemessenen Nachlaufzeiten errechnet. Je nachdem ob der gemessene Maximalwert oder der Mittelwert plus drei Standartabweichungen größer ist, wird der höhere Wert im Kanal „maxNachlaufzeit“ gespeichert. Aus diesem Wert wird nun der resultierende Sicherheitsabstand errechnet und als Eigenschaft dem Kanal Nachlaufzeit zugeordnet. Damit ist das Script abgearbeitet.

Das Programm zur Nachlaufwegmessung ruft nun eine Protokollvorlage auf, die von DIAdem automatisch ausgefüllt wird. Das Protokoll wird nun als PDF-Datei mit dem gleichen Name wie die

TDMS-Datei im selben Ordner gespeichert und mit dem Programm geöffnet, das in der Systemsteuerung des Computer dem PDF-Format zugeordnet wurde. Während das Protokoll geöffnet wird, geht das Programm in den Zustand „Herunterfahren“ über.

Hier wird die Sequenz „S8_Herunterfahrsequenz.vi“ ausgeführt, die den Port schließt und die Strukturvariablen löscht. Danach wird das Programm beendet.

7.3 Relais-Schaltung

Damit das Programm zur Nachlaufwegmessung den Not-Aus der Maschine auslösen kann, muss dieser mit der I/O-Buchse am In-/Output 0 verbunden werden. Dies geschieht über eine Relais-Schaltung, die sich in einem geschlossenen Gehäuse befindet, welches über einen 25poligen D-Sub-Stecker mit dem Spider8 und über einen vierpoligen Kabel-Stecker mit der Pressensteuerung verbunden wird. Hier wird der gesamte Not-Aus-Kreis, der mit einer Spannung von 220V betrieben wird, durch ein Relais unterbrochen, welches bei einer Spannung von 24V schaltet und den Kreislauf schließt.

Die I/O-Buchse enthält eine optionale Verstärkungsschaltung, mit der die Spannung eines „True“-Signals von 4,5V auf maximal 50V erhöht werden kann, indem eine weitere Spannungsquelle an Common (Pin 19) und an der Masse (Pin 9) angeschlossen wird[38]. Das Relais wird am Output 0 (Pin 13) und Common (Pin 19) angeschlossen. Bei einem „True“-Signal des Spider8 schaltet nun das Relais und der Not-Aus-Kreis wird freigegeben, sodass der Pressenstempel bewegt werden kann.

Damit der exakte Auslösezeitpunkt des Not-Aus bei der Auswertung berücksichtigt wird, wird im Programm ein externer Trigger definiert. Dieser startet die Messung erst, wenn der Pin 14 mit der Masse der I/O-Buchse verbunden wird. Dazu wird am Output 0 ein weiteres baugleiches Relais parallel geschaltet, das dieses Signal durchschaltet und dadurch die Messung möglichst gleichzeitig mit dem Auslösen des Not-Aus startet.

Die komplette Schaltung ist in Abbildung 19 dargestellt.

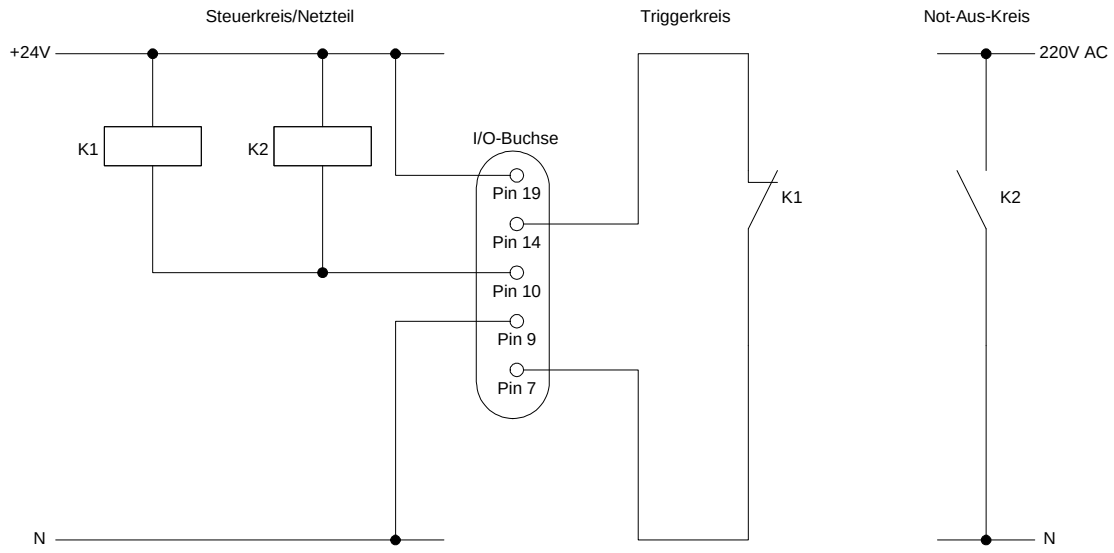


Abbildung 19: Schaltskizze der Relaisschaltung

[38] Vgl: Hottinger Baldwin Messtechnik:PC Messelektronik/Spider8 Spider8-30 Spider8-01, Darmstadt, Hottinger Baldwin Messtechnik, S.B-15

7.4 Durchführung einer Messung mit dem erstellten Programm zur Nachlaufwegmessung

7.4.1 Bestimmen der maximalen Geschwindigkeit der hydraulischen Presse

Die meisten Parameter, die für eine Nachlaufwegmessung relevant sind, ergeben sich aus der technischen Dokumentation der Geräte oder aus dem Versuchsaufbau und sind damit bekannt. Die technische Dokumentation der hydraulischen Presse enthält bei der Geschwindigkeit des Stempels allerdings eine für die Nachlaufwegmessung irreführende Angabe, denn hier wird eine vom Hersteller garantierte Geschwindigkeit von 500mm/s angegeben. Die Erfahrung deutet allerdings auf eine tatsächliche maximal Geschwindigkeit von über 700mm/s hin. Die DIN EN ISO 13855 schreibt für eine Nachlaufwegmessung vor, dass der für den Maschinenbediener schlimmste Fall simuliert werden soll, was hier bedeutet, dass die größtmögliche Geschwindigkeit als Auslösebedingung eingestellt werden muss. Da diese nicht bekannt ist, muss sie bestimmt werden. Dazu wird die Presse 10mal auf einem Stempelweg von 375mm maximal beschleunigt, was fast dem kompletten Stempelhub entspricht und das Weg-Zeit-Diagramm aufgenommen. Dies geschieht mit einer Messrate von 200Hz. Dies wird sowohl mit als auch ohne Werkzeug durchgeführt, sodass 20 Weg-Zeit-Diagramme vorliegen.

Aus diesen Diagrammen wird nun ein Geschwindigkeits-Zeit-Diagramm abgeleitet und die 10 Maxima bestimmt. Um nun aus dieser Stichprobe einen Geschwindigkeitswert abzuleiten, der möglichst hoch ist aber trotzdem zuverlässig von der Presse erreicht wird, wird aus diesen Ergebnissen das arithmetische Mittel gebildet und davon drei Standardabweichungen abgezogen. Unter der Annahme, dass die Messergebnisse normal verteilt sind, ergibt sich so ein Wert, der zu 99,85% erreicht wird. Zusätzlich kann aus den Weg-Zeit-Diagrammen auch der Weg herausgelesen werden, den die Presse mindestens benötigt, um die maximale Stempelgeschwindigkeit zu erreichen. Dazu werden die Zeitpunkte zwischen Stillstand der Presse und maximaler Geschwindigkeit aus dem Geschwindigkeits-Zeit-Diagramm herausgelesen und der Stempelweg zwischen diesen Zeitpunkten ermittelt.

7.4.2 Ausführen des Programms

Vor dem Starten des Programms sollte bereits die Relais-Schaltung mit der Maschine verbunden sein. Wenn dies der Fall ist, muss der Not-Aus-Kreis erst geschlossen werden, damit die Maschine freigegeben wird. Deswegen kann es sein, dass der Not-Aus nach Anschließen der Relais-Schaltung aber vor Programmstart bereits ausgelöst wird. Dies ist von der letzten Verwendung des Spider8 abhängig, da die letzen Kanaleinstellungen und damit die Signale der I/O-Buchse gespeichert werden. Beim Programmstart werden korrekte Geräte- und Kanaleinstellungen geladen und der Not-Aus-Kreis geschlossen, womit die Presse freigegeben wird.

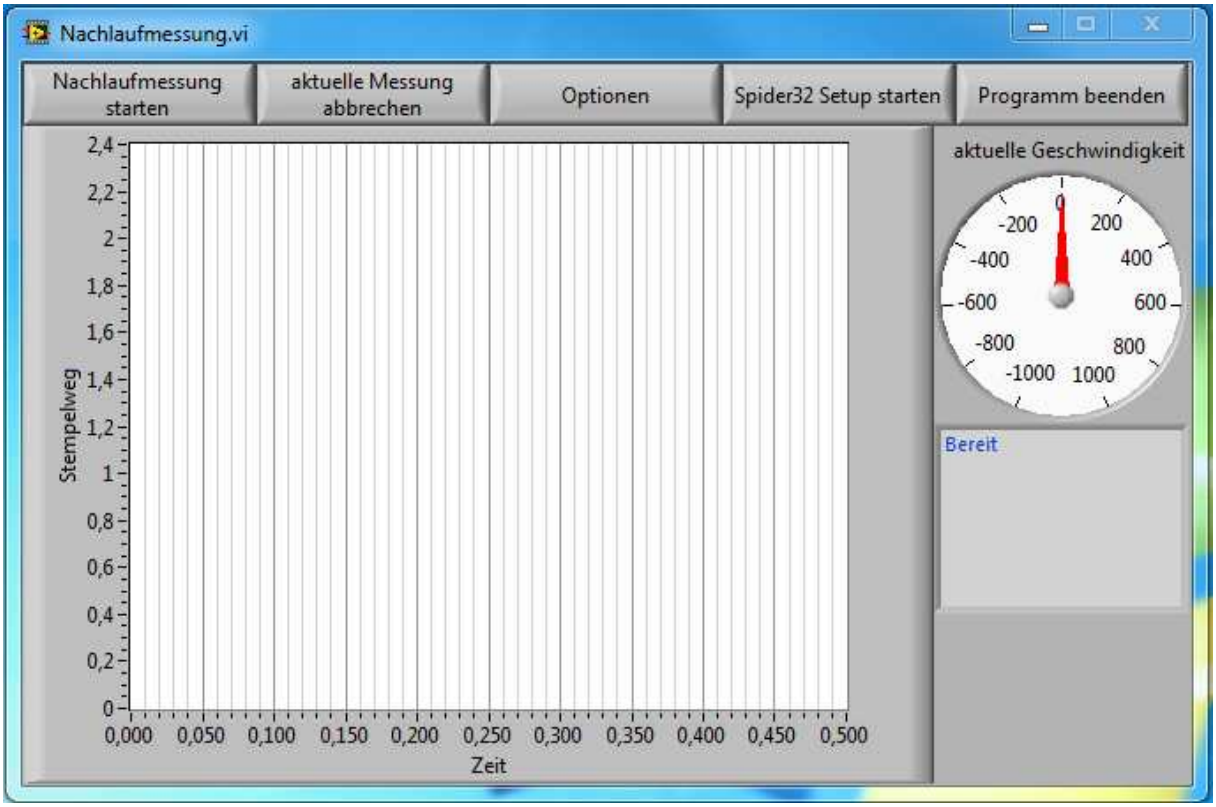


Abbildung 20: Benutzeroberfläche nach Start des Programms

Nach Starten des Programms öffnet sich die unter Abbildung 20 dargestellte Benutzeroberfläche. Diese enthält ein Diagrammfeld, in der das Weg-Zeit-Diagramm der aktuellen Messung dargestellt wird, eine Geschwindigkeitsanzeige, in der die aktuelle Stempelgeschwindigkeit in mm/s dargestellt wird, ein Textfenster in dem der aktuelle Bearbeitungsschritt angezeigt wird sowie eine Leiste mit Buttons zu Steuerung des Programms.

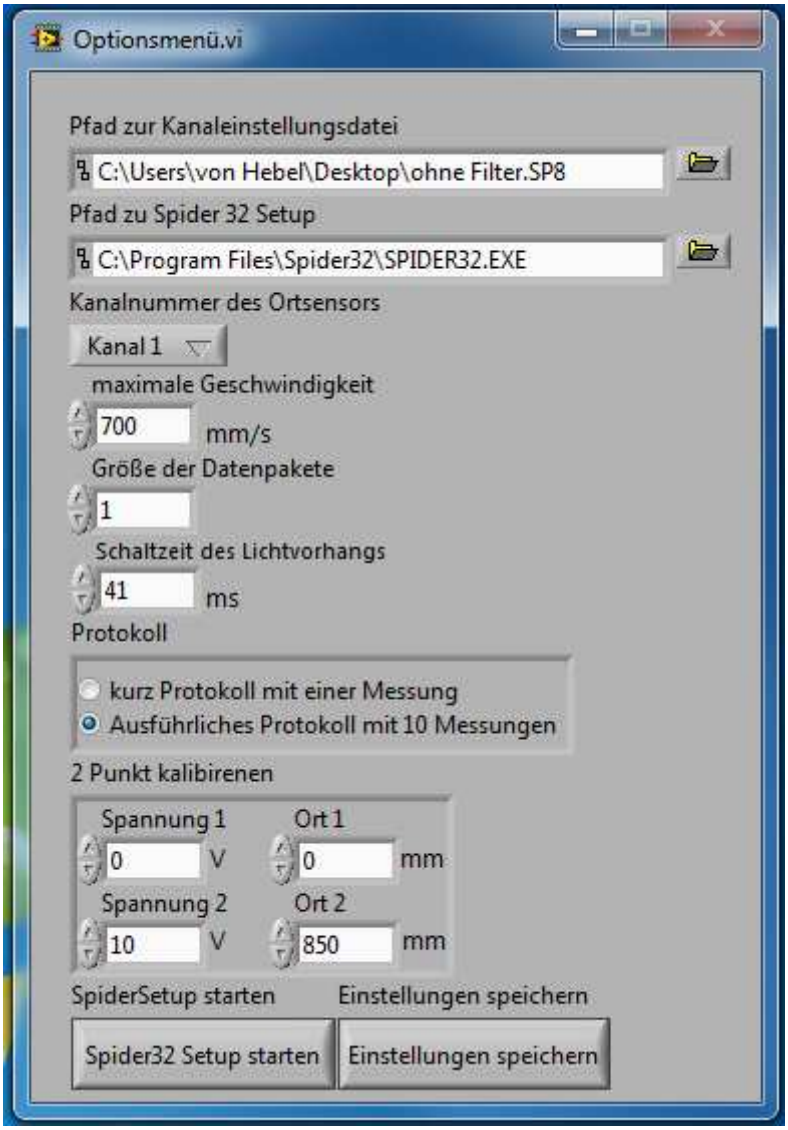


Abbildung 21: Das Optionsmenü

Der Button Optionen öffnet ein Optionsmenü in dem die Parameter der Messung festgelegt werden. Das Menü wird in Abbildung 21 dargestellt. Geräteeinstellungen und Kanaleinstellungen können hier allerdings nicht geändert werden. Dafür kann über die Buttons „Spider32 Setup starten“ das Programm Spider32 Setup aus dem Optionsmenü oder aus dem Hauptfenster heraus aufgerufen werden. Dazu muss bei „Pfad zu Spider32 Setup“ allerdings der korrekte Pfad zur Spider32.exe angegeben werden. Dort kann eine SP8-Datei mit Geräte- und Kanal-Einstellungen erstellt werden, aus der diese Einstellungen geladen werden können. Der Pfad zur SP8-Datei muss dafür unter „Pfad zur Kanaleinstellungsdatei“ angegeben werden.

Der Kanal, an dem der Ortsensor angeschlossen ist, kann unter Kanalnummer des Ortsensors ausgewählt werden. Unter „maximale Geschwindigkeit“ muss die maximale Geschwindigkeit der Presse angegeben werden. Bei dieser Geschwindigkeit wird der Not-Aus der Presse betätigt.

Der Speicher des Spider8 ist auf 20.000 Messwerte begrenzt. Bei hohen Messraten kann dieser Speicher vollaufen, was zu einem Abbruch der Messung führt. Um das zu vermeiden, kann die Größe der Datenpakete, mit der die Messwerte herunter geladen werden, erhöht werden. Die Größe muss unter „Größe der Datenpakete“ angegeben werden.

Bei der Berechnung des Sicherheitsabstands ist die Schaltzeit der Sicherheitseinrichtung zu berücksichtigen. Diese muss bekannt sein (z.B. über das technischen Datenblatt der Sicherheitseinrichtung) und wird in dem Eingabefeld „Schaltzeit des Lichtvorhangs“ eingegeben.

Das Programm ermöglicht die Erstellung von zwei unterschiedlichen Protokollarten: Ein Kurzprotokoll, bei dem nur eine Messung durchgeführt wird und bei dem nur aus dieser einen Messung ein Sicherheitsabstand berechnet wird, oder ein vollständigem Protokoll nach DIN EN ISO 13855 bestehend aus zehn Einzelmessungen. Die Art des Protokolls wird unter „Protokoll“ ausgewählt.

Letztlich wird hier auch eine Zweipunkt-Kalibrierung des Ortsensors durchgeführt. Dazu werden zwei Punkte der Sensor-Kennlinie unter „2 Punkt kalibrieren“ eingegeben.

Mit dem Button „Einstellungen speichern“ werden das Menü geschlossen und die Einstellungen gespeichert. Das schließt ein erneutes Laden der Geräte- und Kanaleinstellungen in die Strukturvariablen und auf den Spider8 mit ein.

Mit dem Button „Nachlaufwegmessung starten“ kann die Messung gestartet werden. Sollte das Programm den Not-Aus der Presse nicht auslösen, z. B. weil eine falsche Angabe zur maximalen-Geschwindigkeit eingegeben wurde, oder die Messung nicht beenden, kann die Messung über den Button „aktuelle Messung abbrechen“ beendet werden.

Nach dem Starten der Messung öffnet sich ein Datei-Dialog, bei dem ein Speicherort und ein Namen für die TDMS-Datei angegeben werden muss. Das Protokoll, das am Ende der Messung erstellt wird, ist eine PDF-Datei, die mit demselben Namen im selben Ordner erstellt wird.

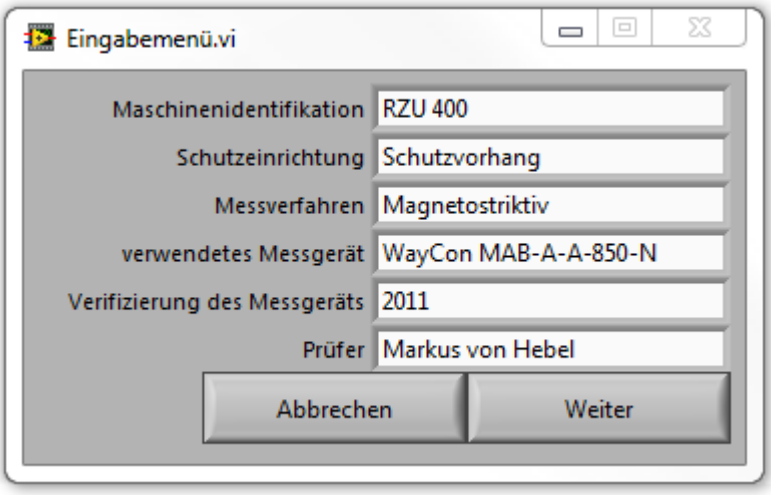


Abbildung 22: Eingabemenü für zusätzliche Angaben zum Protokoll

Danach öffnet sich das Eingabemenü, das in Abbildung 22 dargestellt wird. Hier werden Angaben zur Maschine, Schutzeinrichtung, Messverfahren, Messgerät und Prüfer gemacht. Diese Angaben werden sowohl in der TDMS-Datei als auch im Protokoll gespeichert.

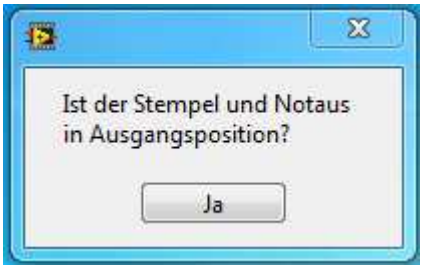


Abbildung 23: Warndialog vor Beginn der Messung

Nun beginnt die eigentliche Messung. Dazu muss der Stempel in Ausgangsposition sein und der Not-Aus freigegeben sein. Um dies sicherzustellen, öffnet sich ein Dialog-Fenster das in Abbildung 23 dargestellt ist und den Maschinenbediener daran erinnert.

Abbildung 24: Dialogfeld nach einer Messung

Nun kann die Bewegung des Stempels gestartet werden. Dabei wird kontinuierlich die Stempelgeschwindigkeit gemessen und bei Erreichen der Maximalgeschwindigkeit wird der Not-Aus ausgelöst und der Nachlauf aufgenommen. Unter „aktuelle Geschwindigkeit“ wird dabei die aktuelle Geschwindigkeit angezeigt. Nach der Messung wird das Kraft-Weg-Diagramm im Hauptfenster dargestellt und das Dialogfenster, das in Abbildung 24 dargestellt ist, öffnet sich.

Jetzt muss anhand des Kraft-Weg-Diagramms überprüft werden, ob die Messung erfolgreich war. Dabei sollte beachtet werden, dass das Programm als letzten Wert in jeder Messung den Wert 0 anhängt. Dies ist notwendig, um sicherzustellen, dass die Suche nach Maxima in der Auswertung in jedem Fall einen Wert ergibt. Wenn die Messung korrekt ausgeführt wurde, wird mit dem Button „Ja“ die Ermittlung des Nachlaufs fortgesetzt. Ein Drücken des Buttons „Nein“ ermöglicht die Wiederholung der Messung und ein Betätigen des Buttons „Abbrechen“ beendet die Messung ohne weitere Auswertung.

Je nachdem ob ein vollständiges Protokoll oder ein Kurzprotokoll erstellt werden soll, werden diese Schritte zehnmal wiederholt und danach zur Auswertung DIAdem gestartet. Von der eigentlichen Ausführung von DIAdem sieht der Anwender lediglich ein kleines Symbol in der Taskleiste. Trotzdem führt DIAdem alle Berechnungen zur Auswertung durch und erstellt automatisch ein Protokoll. Dies kann einige Momente dauern, was dem Anwender durch eine Textnachricht im Hauptfenster angezeigt wird. Diese ist in Abbildung 25 dargestellt.

Abbildung 25: Textnachricht bei Erstellung des Protokolls

Das Protokoll wird automatisch erstellt und gespeichert. Dabei enthält das vollständige Protokoll alle gemessenen Nachlaufzeiten und -wege, während beim Kurzprotokoll das Kraft-Weg-Diagramm dargestellt wird. Ein Beispiel für je ein Kurzprotokoll und ein vollständiges Protokoll ist im Anhang zu finden. Das Programm wird nach einer Protokollerstellung automatisch beendet.

8 Ermittlung von Grenzen der Bibliothek

8.1 Messen der Schaltzeit der I/O-Buchse

Eine technische Grenze, die bei der Benutzung der I/O-Buchse berücksichtigt werden sollte, ist die Schaltzeit, mit der die I/O-Buchse reagiert bzw. wie viel Zeit vom Senden des Signals am PC bis zum Schalten am Spider8 verget. Diese Zeit ist die Summe aus zwei einzelnen Komponenten, der reinen Schaltzeit des Geräts und der Übertragungszeit des Signals vom PC an den Spider8. Diese Komponenten lassen sich nicht einzeln messen, denn es nicht möglich den Empfangszeitpunkt des Signals am Spider8 exakt zu ermitteln. Zudem ist es schwierig, den genauen Sendezeitpunkt und den genauen Schaltzeitpunkt zu messen, da die erstgenannte Zeit im Programm gemessen werden muss, die zweitgenannte aber im Spider8. Hinzu kommt, dass die VIs, die in LabVIEW zur Verfügung stehen, die Zeit lediglich in Millisekunden aufnehmen können, was eine zu geringe Auslösung für diese Messaufgabe darstellt. Um eine höhere Auflösung zu erreichen, wird das Signal der I/O-Buchse mit dem Anschluss eines Messkanals des Spider8 verbunden und so kontinuierlich mit der maximalen Messrate des Spider8 von 9600Hz gemessen. Der genaue Startzeitpunkt des Schaltvorgangs wird durch ein ein „True“-Signal angegeben, welches vom Messprogramm so schnell wie möglich wieder in ein „False“-Signal umgeschaltet werden soll. Im Spannungs-Zeit-Diagramm entspricht die Gesamtschaltzeit der Dauer des „True“-Signals. Dabei entspricht 4,5V dem „True“-Signal und 0,03V dem „False“-Signal. Um eine höhere Aussagekraft zu erhalten, wird dieser Versuch zehnmal wiederholt.

8.2 Bestimmen der maximalen Messrate beim Steuern

Die nächste technische Grenze, die beim Erstellen von Messprogrammen eine Rolle spielt, ist die maximal mögliche Messrate. Wenn der Spider8 die Steuerung der Messrate selbst überwacht, gelten die Angaben der technischen Dokumentation, da dies losgelöst vom PC geschieht. Diese gibt 21 voreingestellte Messraten vor und die maximal einstellbare Messrate beträgt 9600Hz. Diese Werte gelten bei der Verwendung der Treiberfunktion S8_ACQStart.vi, die in den Sequenzen S8_Messung.vi und S8_kon_Messung.vi verwendet wurde. Sie sind allerdings ungeeignet, um mit den gemessenen Werten zu steuern, denn die Messwerte werden nach dem First-in-First-out-Prinzip vom Spider8 herunter geladen, also ist der zuerst gemessene Wert auch der erste Wert, der herunter geladen wird. Dieser Wert ist aber nicht zwangsläufig der aktuelle Wert und da auch kein Messzeitpunkt aufgenommen wird, ist nicht feststellbar, wie aktuell der gemessene Wert genau ist. Außerdem kann es vorkommen, dass bei hohen Messraten die Werte schneller erzeugt werden, als heruntergeladen. Die im Programm aufgenommenen Werte sind also in der Regel älter als die Realität und daher reagiert die Steuerung träger, je höher die Messrate ist.

Deswegen sollte bei Steuerungsaufgaben die Treiberfunktion S8_MeasOneVal.vi verwendet werden, was bei der Sequenz S8_Einzelmessung.vi der Fall ist. Hier übernimmt das Messprogramm die Steuerung der Messrate und nicht die Hardware, indem die LabVIEW-Funktionen „Timerwert“ und „bis zum nächsten vielfachen von ms warten“ verwendet werden. Mit der „Timerwert“-Funktion, wird die Zeit in ms angegeben, die nach Starten des Programms vergangen ist, und durch die Funktion „bis zum nächsten vielfachen von ms warten“ unterbricht das Programm bis der Timerwert ein Vielfaches der angegebenen Zeit beträgt. Hier sind also vorwiegend das Programm, die Prozessorleistung und die Übertragungsgeschwindigkeit zum Spider8 für die Messfrequenz verantwortlich. Um die in diesem Fall maximal mögliche Messrate zu bestimmen, wird eine Messung mit 150 Messwerten je Frequenzen durchgeführt, wobei die Frequenzen 100Hz, 200Hz und 1000Hz im Messprogramm eingestellt werden. Dabei ist 1000Hz die höchste einstellbare Frequenz, denn die Timer-Funktionen in LabVIEW arbeiten in Millisekunden. Das Programm misst also mit dieser Einstellung so schnell wie möglich. Danach werden die angegebenen Werte mit den gemessenen Werten verglichen. Dies wird bei eingestellter Gerätemessrate von 100Hz, 1200Hz und 9600Hz wiederholt, um die Ergebnisse auf einen Einfluss der Geräteeinstellungen hin zu überprüfen.

9 Vorstellung einer einfachen Messsequenz mit der Bibliothek

Mit der nun vorliegenden Bibliothek sind mit wenigen Handgriffen und Programmierkenntnissen sowie grundlegenden Kenntnissen der Messtechnik Messprogramme erstellbar. Um dies zu verdeutlichen wird hier ein simples Beispiel eines Messprogramms und sein Aufbau detailliert erläutert. Das Schreiberbeispiel genannte Programm befindet sich ebenfalls in der VI-Bibliothek und kann von dort aus geladen und bei Bedarf modifiziert werden.

Generell wird die Sequenz durch den Fehlercluster gesteuert, der sich durch alle verwendeten Knoten zieht und im Blockdiagramm violett dargestellt ist. Es arbeitet also von links nach rechts die Knoten ab. Am linken Rand des Blockdiagramms werden alle benötigten Randbedingungen aufgelistet. Dabei handelt es sich um die Anzahl der Messwerte, die Kanäle, die ausgelesen werden sollen, der Pfad zu einer Einstellungsdatei mit Geräte und Kanaleinstellungen und einer Cluster-Konstanten, die die nötigen Angaben für eine Verbindung über USB enthält. Messwerteanzahl, Kanäle und Einstellungsdateipfad werden dabei über ein Bedienelement im Frontpanel eingegeben.

Mit der Einstellungsdatei, die z. B. Messrate, Sensortyp sowie Filtereinstellungen enthält, und den Verbindungsangaben öffnet S8_InitSequenz den Port und erstellt passende Strukturvariablen, die danach an das Gerät übertragen werden. S8_ACQSetup aktiviert alle im Kanäle-Array angegebenen Kanäle zum Messen. S8_Messung startet eine Messung mit der angegebenen Anzahl von Messwerten und lädt diese herunter. Danach wird die Messung automatisch beendet. Diese Messergebnisse werden in der Einheit mit der der Spider8 das Signal des Sensors empfängt in einem 2D-Array ausgegeben und danach in einem Signalverlaufsdiagramm angezeigt. S8_Herunterfahrsequenz beendet die Verbindung und löscht die Strukturvariablen aus dem Arbeitsspeicher. Am Ende ist noch S8_Fehlerhandler ergänzt, das aus den Daten des Fehlercluster eine Fehlermeldung erstellt, wenn irgendwo ein Fehler aufgetreten ist. Danach ist jeder Datenfluss an einer Datensenke angekommen und alle Knoten wurden abgearbeitet, wodurch das Programm beendet wird. Das Frontpanal ist in Abbildung 26 und das Blockdiagramm in Abbildung 27 dargestellt.

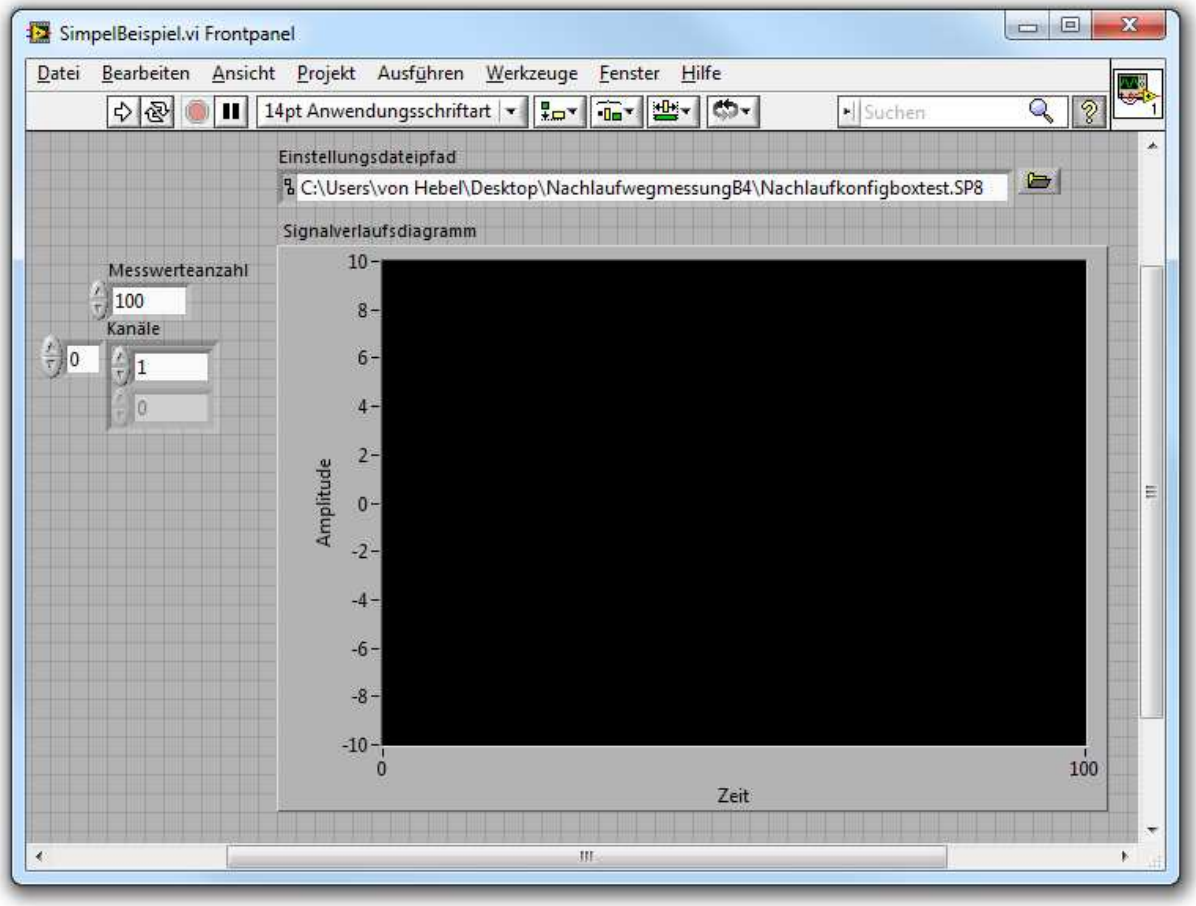


Abbildung 26: Frontpanel des vorgestellten Beispiels

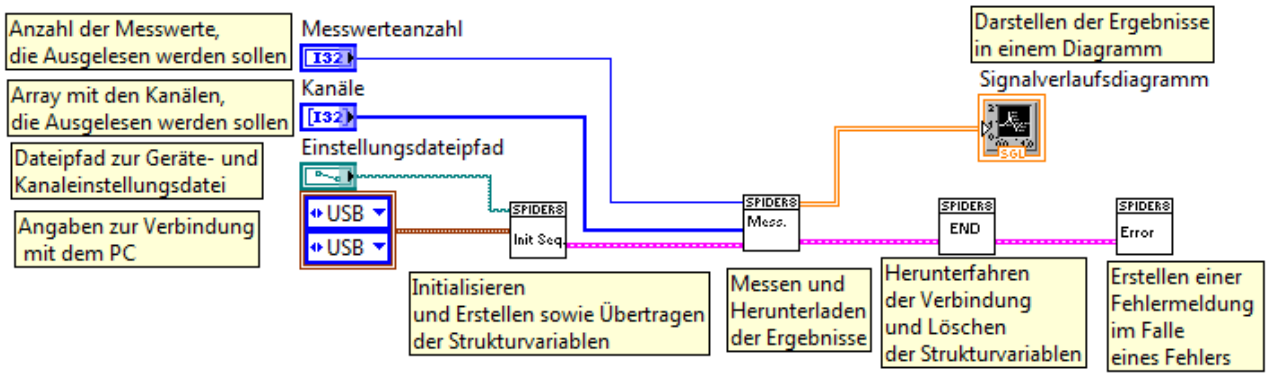


Abbildung 27: Blockdiagramm des vorgestellten Beispiels

10 Darstellung der Grenzen des Verfahrens

10.1 Abhängigkeit der Genauigkeit einer Geschwindigkeitsmessung von der Messrate

Ein gängiges Verfahren zur Ermittlung einer Geschwindigkeit eines Objekts ist die Errechnung der Geschwindigkeit aus zwei Ort-Messungen und den dazugehörigen Zeitunterschied gemäß Gleichung (5).

$$v = \frac{ds}{dt} \tag{5}$$

Dabei ist v gleich der Geschwindigkeit, ds gleich dem Weg zwischen zwei Orten und dt gleich der Zeitunterschied. Bei kontinuierlichen Geschwindigkeitsbestimmungen wird der Zeitunterschied durch die Messrate bestimmt. Je größer die Messrate ist, desto kleiner wird der Zeitabstand zwischen zwei Messungen und damit auch der Divisor der Gleichung. Dies führt bei digitalen Messungen zu einer steigenden Ungenauigkeit der Geschwindigkeitsmessung bei steigender Messrate, denn Bitsprünge bei der sind Messfehler, die beim umwandeln des analogen Signals in ein digitales Signal entstehen. Wenn der tatsächliche zu messende Wert aufgrund der Auflösung nicht genau bestimmt werden kann, wird entweder auf- oder abgerundet. Ein konstantes analoges Signal kann dann im digitalen Signal zwischen zwei immer gleichen Werten zufällig hin und her springen. Der maximale Rundungsfehler ist hier die Auflösung des A-D-Wandlers. Dieser konstante Wert wird bei steigender Messrate durch einen immer kleiner werdenden Divisor geteilt und führt zu einem immer größer werdenden Unterschied zwischen errechneter Geschwindigkeit und der realen Geschwindigkeit.

Der Spider8 verfügt über eine Auflösung von ±25.000Digits beim eingestellten Messbereich. Bei einem Sensor, der ein Spannungssignal an den Messkanal des Spider8 sendet, wie es beim Waycon MAB-A-A-850-N der Fall ist, ist ein Messbereich von 10V vorgegeben. Da Gleichung (6) gilt beträgt ein Bitsprung 0,4mV.

$$Bitsprung = \frac{Messbereich}{Auflösung} = \frac{10V}{25.000} = 0{,}4mV \tag{6}$$

Gleichung (5) kann, um den durch einen Bitsprung auftetenden maximalen Fehler im Geschwindigkeits-Zeit-Diagramm zu berechen, modifiziert werden, indem für ds der Betrag eines Bitsprungs eingesetzt wird und für dt die Messrate. Daraus ergibt sich Gleichung (7)

$$\max.Fehler = \frac{Bitsprung}{(1 \div Messrate)}$$

In Abbildung 28 kann der Fehler in Abhängigkeit zur Messrate für alle am Spider8 einstellbaren Messraten abgelesen werden und es wird deutlich, dass der maximale Fehler linear von der Messrate abhängt.

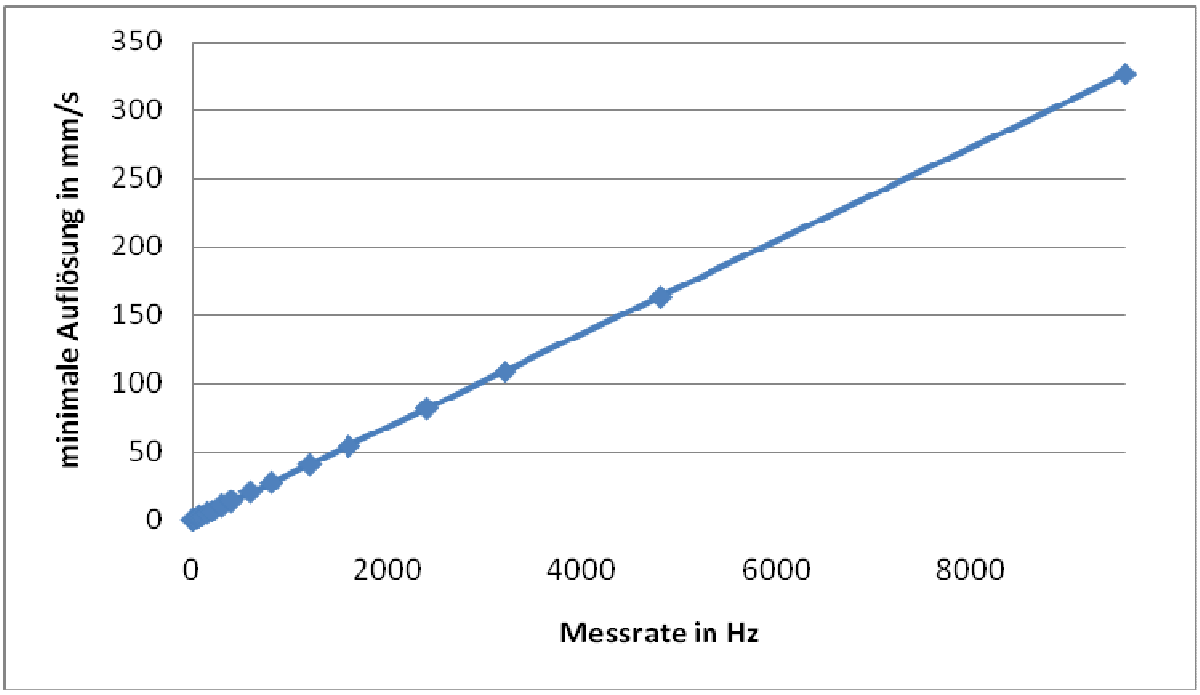


Abbildung 28: Abhängigkeit der Auflösung von der Messrate

10.2 Auswertung der Schaltzeitmessung

Da es mit dem Spider8 nicht möglich ist, gänzlich ohne Filter Messwerte aufzunehmen, wird hier mit einem möglichst sensiblen Besselfilter gearbeitet und es wird als Zeitpunkt für den Beginn des Schaltvorgangs der erste ansteigende Wert und für das Ende des Schaltvorgangs der erste fallende Wert im Spannungs-Zeit-Diagramm gewählt.

Die gemessenen Schaltzeiten sind der Abbildung 29 zu entnehmen.

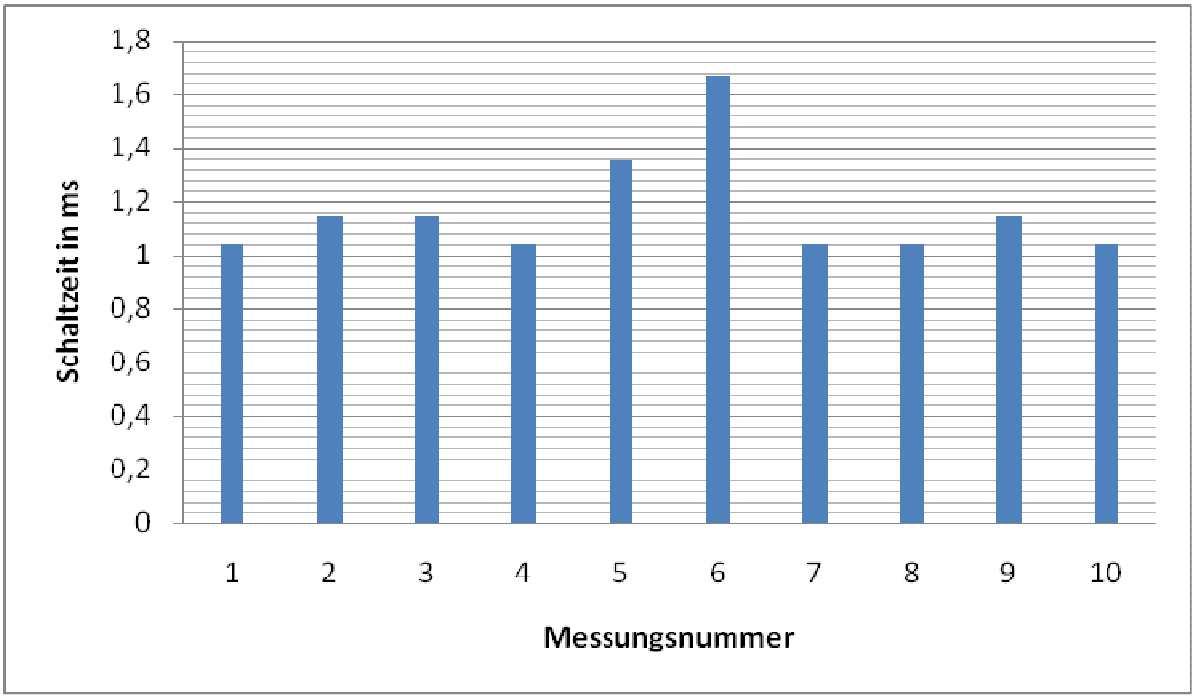


Abbildung 29: Ergebnisse der Schaltzeitmessung

Die Ergebnisse der Auswertung inklusive Minimum, Maximum, Mittelwert und Standardabweichung sind in Tabelle 3 aufgelistet.

Tabelle 3: Auswertung der Schaltzeitmessung

	Gesamtschaltzeit in ms
Minimum	1,042
Maximum	1,667
Mittelwert	1,167
Standardabweichung	0,201

Die Geräte-Schaltzeit ist laut Handbuch von der aktuellen Messrate abhängig und beträgt 1/Messrate[39]. Das ergibt in diesem Fall einen Wert von ca. 0.104ms. Nimmt man nun eine Normalverteilung für die Schaltzeiten an, liegt die Verzögerung, die sich durch die Verwendung eines Steuerprogramms zum Schalten bzw. Auslesen der I/O-Buchse mit einer Wahrscheinlichkeit von 99.85% unter dem Wert, der sich gemäß Formel (7) errechnen lässt. Da die Messergebnisse mit den VIs der Bibliothek und dem vorliegenden PC gemessen wurden, gelten die hier bestimmten Verzögerungen für die Verwendung der VI-Bibliothek.

$$t_{schalt} = t_{mittel} - t_{Gerät} + 3 \cdot \sigma \qquad (7)$$

Dabei entspricht t_{schalt} der zusätzlichen Verzögerung durch das Programm, t_{mittel} der Schaltzeit des Geräts und σ der Standardabweichung. Durch Einsetzen der Werte in (7) ergibt sich für t_{schalt}:

$$t_{schalt} = 1{,}167ms - 0{,}104ms + 3 \cdot 0{,}201ms = 1{,}666ms \qquad (8)$$

10.3 Maximale Messrate beim Steuern

Zusätzlich zu dem unter 9.2 beschriebenen Versuch wurde ein Programm erstellt, das nur der Überprüfung der Timerfunktionen dient. Hier wird eine Schleife mit der Funktion „bis zum nächsten Vielfachen von ms warten“ ausgeführt und nach jedem Wartevorgang der Timerwert ausgelesen. Dabei zeigt sich, dass die Timerwert-Funktionen um ±1ms schwanken. Diese Ungenauigkeit liegt innerhalb von LabVIEW und lässt sich nicht beheben. Daher werden Schwankungen um 1ms bei der Auswertung nicht berücksichtigt.

Die eigentliche Fragestellung ist, wie lange das Herunterladen, Ablegen in ein Array und das Verarbeiten der Messwerte benötigt. Dies ist sehr stark vom Verarbeiten der Messwerte abhängig, daher ist es sehr schwierig eine allgemein gültige Aussage zu treffen. Der durchgeführte Versuch

[39] Vgl: Hottinger Baldwin Messtechnik:PC Messelektronik/Spider8 Spider8-30 Spider8-01, Darmstadt, Hottinger Baldwin Messtechnik, S. B-13

verzichtet auf einen Verarbeitungsteil, sodass hier hauptsächlich die Kommunikationsgeschwindigkeit und das Array-Handling einen Einfluss ausüben. In Abbildung 30 wird die Erfolgsrate beim Einhalten der verschiedenen Messraten bei unterschiedlichen Geräteeinstellungen dargestellt.

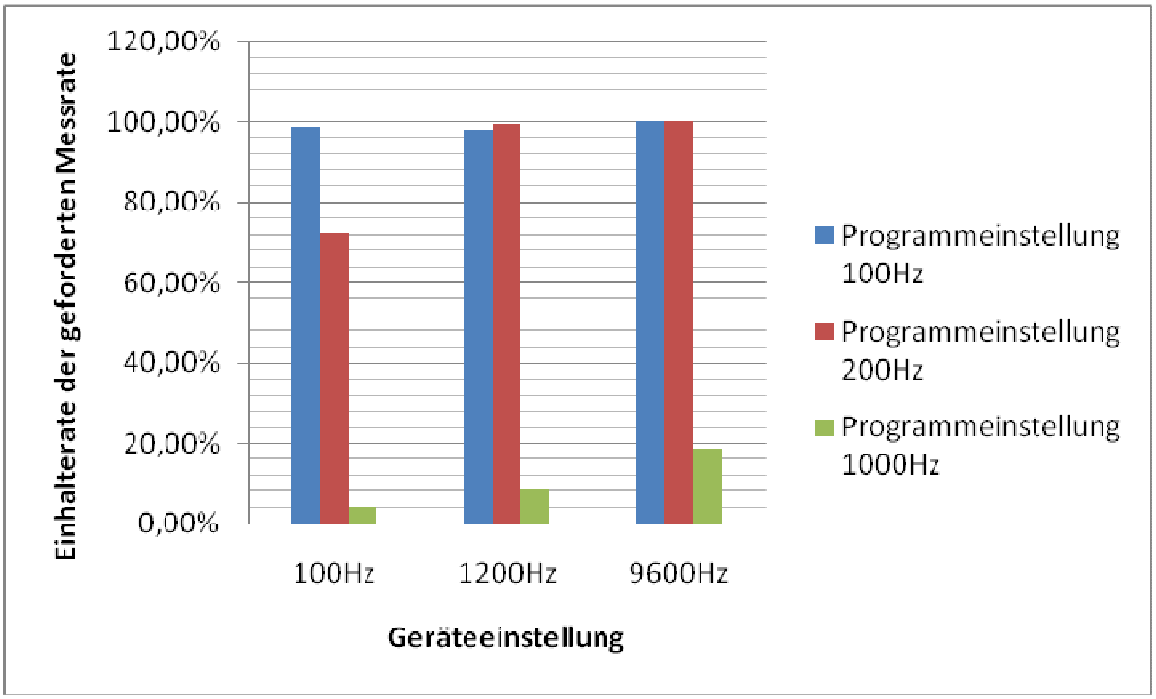


Abbildung 30: Erfolgsraten beim Einhalten der geforderten Messrate

Hier zeigt sich, dass bei eingestellten 100Hz am Spider8 das Programm auch nur 100Hz realisieren kann. 200Hz wird nur in ca.72 % der Fälle und 1000Hz nur in ca. 4% der Fälle realisiert.

Mit eingestellten Geräteeinstellungen von 1200Hz, ist auch eine Messrate von 200Hz vom Programm erzeugbar. Die 1000Hz werden aber auch hier nur zu 9% erreicht. Es bestätigt sich also die Vermutung, dass die Geräteeinstellung der Messrate einen Einfluss auf die maximal erreichbare Messrate hat, obwohl das Programm und nicht das Gerät die Steuerung der Messrate übernimmt. Da zusätzlich auch die Schaltzeiten der I/O-Buchse direkt von der eingestellten Messrate abhängen, kann man vermuten, dass das gesamte Gerät zyklisch mit einer von der Geräteeinstellung bestimmten Periodendauer arbeitet.

9600Hz entspricht der maximal am Gerät einstellbaren Messrate. Auch hier können die Messraten 100Hz und 200Hz mit einer ausreichenden Sicherheit gewährleistet werden. Die 1000Hz sind allerdings nur mit einer Einhalterate von ca. 19% erreichbar. Bei der Betrachtung der einzelnen Messwerte, fällt auf, dass 97.3% der Messwerte mit einem dt von 5ms oder weniger gemessen wurden, was 200Hz entspricht, während nur 78,5% der Messwerte über ein dt von 4ms oder weniger verfügen. Da in diesem Fall der bestmögliche Fall generiert wurde, entspricht die maximal

erreichbare Messrate bei der Verwendung der Treiberfunktion S8_MeasOneVal also 200Hz. Es zeigt sich allerdings auch, dass diese Funktion zum Messen von reinen Messaufgaben ungeeignet ist, da selbst bei optimalen Vorraussetzungen die Messrate nicht absolut konstant ist und sie zudem deutlich geringer ist, als die maximale Gerätemessrate.

11 Bewertung der Ergebnisse des Programms zur Nachlaufwegmessung

11.1 Ergebnisse der Geschwindigkeitsmessung

Die Ergebnise der Geschwindigkeitsmessung können der Abbildung 31 entnommen werden

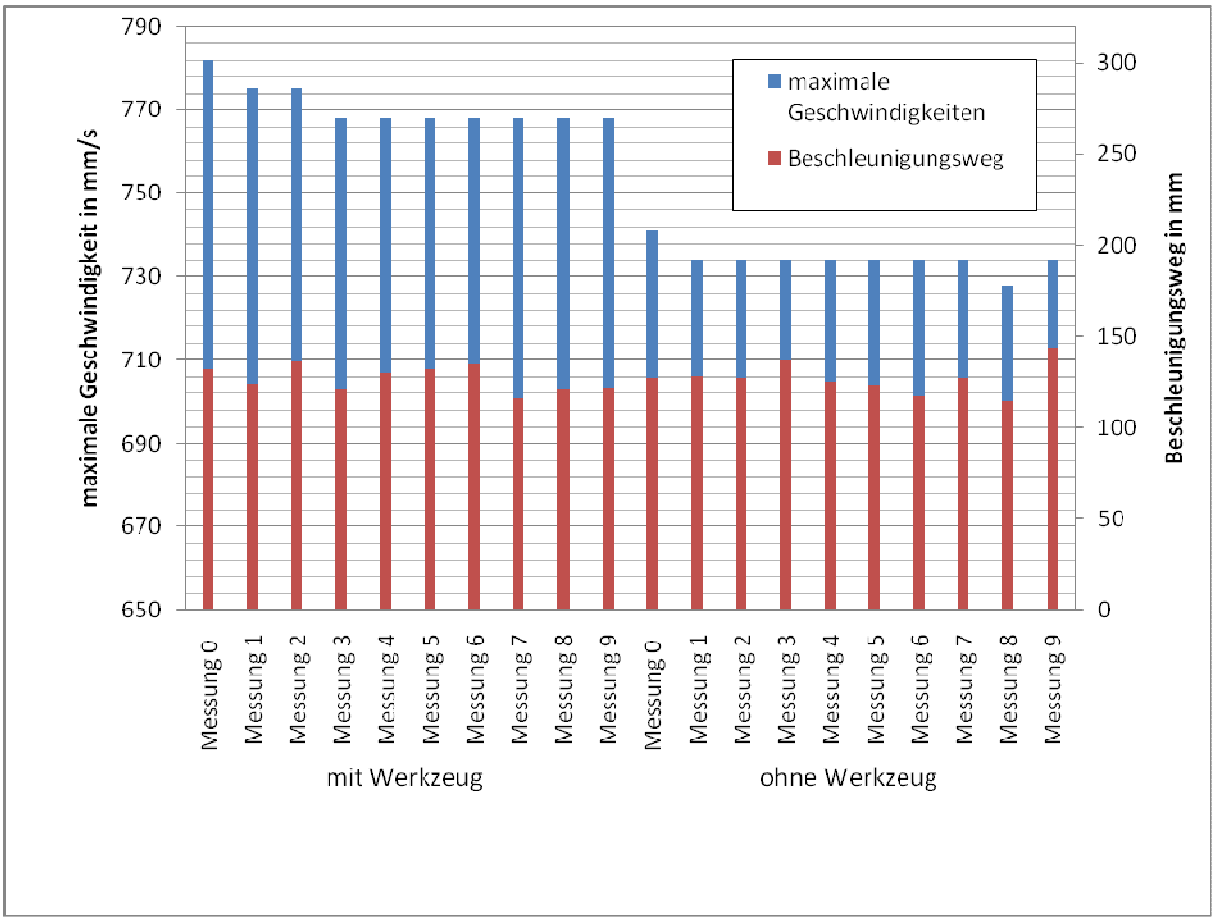


Abbildung 31: Ergebnisse der Geschwindigkeitsmessung

Es fällt auf, dass die Presse, wenn sie mit einem Werkzeug bestückt ist, über eine höhere Maximalgeschwindigkeit verfügt. Dies liegt vermutlich an der Wirkung der zusätzlichen Masse des Werkzeugs auf den hydraulischen Antrieb. Generell sollten Nachlauf-Messungen mit einem Werkzeug durchgeführt werden, da die höhere Masse zu einem längeren Nachlauf führt und damit der für den Maschinenbediener ungünstigeren Fall simuliert wird.

Um nun eine Geschwindigkeit für die Nachlaufwegmessung mit eingebautem Werkzeug zu bestimmen, werden vom arithmetischen Mittel drei Standardabweichungen abgezogen, was hier den Wert 756,13mm/s ergibt. Die Untersuchungen unter 10.1 zeigen jedoch, dass die Genauigkeit von diesem Wert nicht erreichbar ist, da der Steuerungsteil des Programms zur Nachlaufwegmessung die Geschwindigkeit mit 200Hz bestimmt. Aufgrund dieser Messfehler wird der Wert auf 750mm/s abgerundet. Dies ist nun die empfohlene Programmeinstellung für die vorliegende Presse vom Typ RZU 400.

Der Beschleunigungsweg scheint allerdings unbeeinflusst vom Werkzeug zu sein. Dieser liegt in beiden Fällen im Mittel bei ca. 127mm. Um auch hier eine Empfehlung für den minimal benötigten Verfahrweg zu geben, werden analog zu der Geschwindigkeitsauswertung zum Mittelwert drei Standardabweichungen addiert, sodass sich der Wert ca. 152mm ergibt. Dazu kommt noch der eigentliche Nachlaufweg, der unter Punkt 11.5 bestimmt wird und ca. 105mm beträgt. Zusätzlich sollte noch eine gewisse Sicherheit vorhanden sein, sodass bei einer Nachlaufmessung mit Werkzeug ein Weg von mindesten 300mm vorhanden sein sollte.

11.2 Wahl der benötigten Messrate

Die ausgewählte Messrate gibt die Auflösung bei der Aufnahme des Weg-Zeit-Diagramms an. Für den Steuerungsteil des Programms zur Nachlaufwegmessung spielt sie nur eine untergeordnete Rolle, da mit der Sequenz S8_Einzelmessung gearbeitet wird und daher die Ergebnisse von Punkt 10.3 gelten. Das bedeutet, solange eine Geräteeinstellung von über 200Hz gewählt wurde, wird es im Programm zu keinen zusätzlichen Ungenauigkeiten in der Geschwindigkeitsmessung kommen. Allerdings beträgt die Auflösung im Kraft-Weg-Diagramm dann lediglich 5ms, was zu ungenau ist. Maximal einstellbar am Gerät sind 9600Hz, was einer maximalen Auflösung von ca. 0,104ms entspricht. Diese Genauigkeit wird zwar vom Messverstärker erreicht, aber die einzelnen gemessenen Nachlaufzeiten streuen zu stark, als dass eine Angabe dieser Genauigkeit sinnvoll wäre. Es wird lediglich auf 1ms genau der Wert vom Programm errechnet, sodass 1000Hz vollkommen ausreichend sind. Diese Einstellung ist am Spider8 nicht wählbar, sodass der nächst höhere Wert gewählt wird, der 1200Hz beträgt.

11.3 Vorstellen des bestehenden Systems zur Nachlaufwegmessung

Das vorliegende System zur Nachlaufwegmessung besteht aus zwei wesentlichen Komponenten: einer Relaisschaltung, die den Not-Aus auslöst, und einer Messvorgabe für das Programm Catman Easy. Catman Easy ist ein Messprogramm, das viele Mess- und Auswerte-Optionen bietet, von Hottinger Baldwin Messtechnik erstellt worden ist und vollständig mit dem Messverstärker vom Typ Spider8 kompatibel ist. Allerdings ist dieses Programm zum Steuern mittels des Spider8 ungeeignet,

da aus diesem Programm heraus die I/O-Buchse nicht ansprechbar ist und die Messung nur über externe oder interne Trigger gesteuert werden kann.

Die Relaisschaltung beinhaltet ein Zeitrelais, das 0,25s nach Erhalt eines Startsignals schaltet und dadurch den Not-Aus-Kreis unterbricht. Dieses Unterbrechungssignal wird als Spannungssignal an einem Messkanal des Spider8 aufgenommen und dient als interner Trigger zum Start der Messung.

Die Messvorgaben enthalten eine Definition der Messungsdauer sowie einer Trigger-Definition und die benötigten Geräteeinstellungen. Dabei wird die Messung bei 4800Hz und einem Bessel-Filter mit einer Filterfrequenz von 600Hz durchgeführt. Catman Easy erzeugt nach einem Messdurchgang automatisch ein Weg-Zeit-Diagramm und ein Spannungs-Zeitdiagramm. Dort werden nun manuell der Haltepunkt des Stempel und der Auslösezeitpunkt des Not-Aus sowie die Stempelposition beim Auslösen herausgelesen und daraus die Nachlaufzeit sowie der Nachlaufweg bestimmt. Laut Anleitung zur Nachlaufmessung, die der Vorgabendatei beiliegt, wird diese Messung 10mal wiederholt und danach aus dem zweithöchsten Wert für die Nachlaufzeit und einer angenommenen Eindringgeschwindigkeit von 1,6m/s ein Sicherheitsabstand errechnet.

11.4 Vergleich des bestehenden Systems und des im Rahmen dieser Arbeit erstellten System.

Beim Vergleich der Ermittlung des Weg-Zeit-Diagramms fällt bereits auf, dass das vorhandene System nicht nach DIN EN ISO 13855 arbeitet, denn das Auslösen des Not-Aus wird im vorherigen System zeitlich gesteuert und ist nicht abhängig von der tatsächlichen Stempelgeschwindigkeit. Daher kann bei diesem System nicht gewährleistet werden, dass tatsächlich die Maximalgeschwindigkeit erreicht wurde. Die tatsächlichen Auslösegeschwindigkeiten einer vollständigen Nachlaufzeit-Messung mit Werkzeug unter Verwendung des bestehenden Systems sind in Abbildung 32 dargestellt.

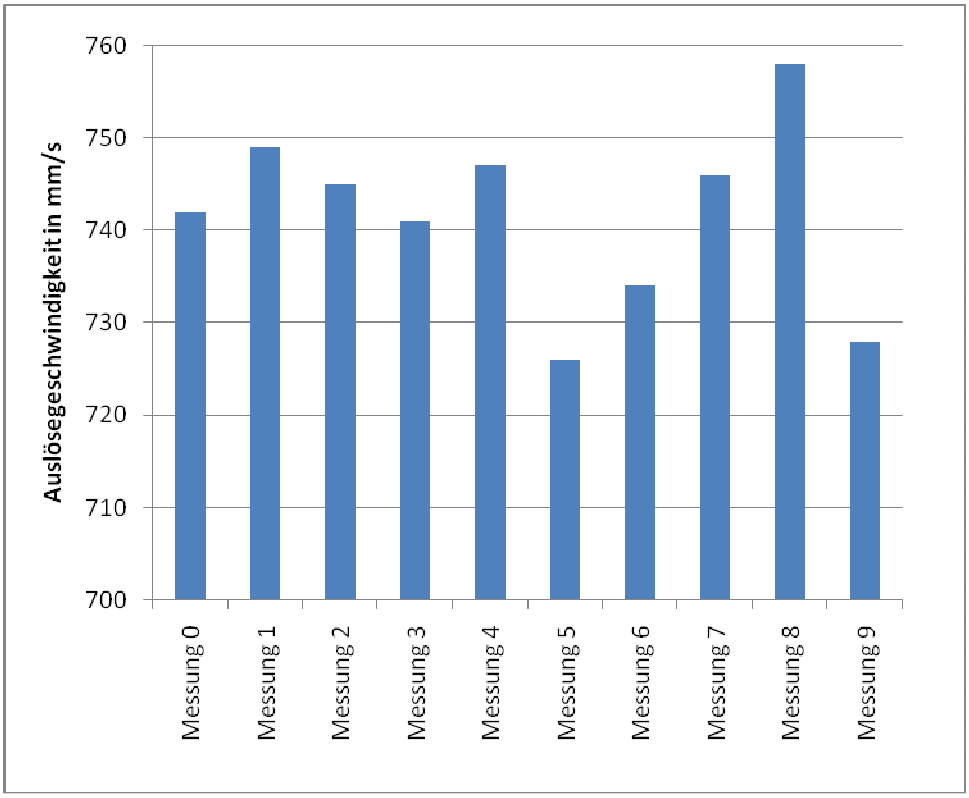


Abbildung 32: Auslösegeschwindigkeit bei Messung mit bestehenden System und eingebautem Werkzeug

Diese Ergebnisse entsprechen den Geschwindigkeiten der unter Punkt 7.4.1 gemessenen Weg-Zeit-Diagramme bei einer Beschleunigungszeit von 0,25s.

Die VIs der Bibliothek können im Gegensatz zu Catman Easy auf die I/O-Buchse zugreifen. Zudem können die Ergebnisse direkt nach dem Herunterladen weiterverarbeitet werden, um daraus z.B. eine Steuerung zu realisieren. Daher kann das in LabVIEW erstelle Programm kontinuierlich die Geschwindigkeit messen und erst bei Erreichen der gewünschten Haltegeschwindigkeit den Not-Aus über ein Signal der I/O-Buchse betätigen. Diese Haltegeschwindigkeit ist im neuen System einstellbar, sodass es auch bei anderen Pressen anwendbar ist. Im bestehenden System müsste die Einstellung

am Zeit-Relais geändert werden, was allerdings beim verwendeten Relais nicht sehr präzise möglich ist.

Das bestehende System liefert lediglich Weg-Zeit-Diagramme. Jede Auswertung ist manuell durchzuführen und die Ergebnisse für eine Nachlauf-Messung sind einzeln außerhalb von Catman Easy zu speichern. Auch die Errechnung des Sicherheitsabstands ist außerhalb von der Messsoftware vorzunehmen. Zudem entspricht die Auswerte-Anleitung nicht der DIN EN ISO 13855. Bei der Verwendung von LabVIEW kann eine automatisierte Auswertung mittels DIAdem umgesetzt werden, was im Falle des neuen Systems gemacht worden ist. Alle nötigen Rechenschritte werden hier von DIAdem durchgeführt und die relevanten Ergebnisse in einem Protokoll gemäß einer Vorlagedatei gespeichert. Diese Auswertung entspricht der DIN EN ISO 13855 und benötigt keine manuell durchgeführten Rechenschritte. Das spart Zeit und ist für den Anwender deutlich komfortabler. In LabVIEW kann das TDMS-Dateiformat verwendet werden, was auch die Speicherung von zusätzlichen Daten, wie z.B. Versuchsparametern oder Annahmen, möglich macht. Diese Option wird im neuen Programm zur Nachlaufwegmessung an mehreren Stellen genutzt, wie z.B. während der Aufnahme der Weg-Zeit-Verläufe und bei der Auswertung. So wird z. B. der Prüfer als eine Eigenschaft in der TDMS-Datei gespeichert.

Es ist bereits festgestellt worden, dass die Parameter des bestehenden Systems nicht optimal sind. Die Messrate von 4800Hz ist sehr viel genauer als notwendig und die Auslösegeschwindigkeit wird nicht kontrolliert und liegt bei ca. 740mm/s, was zudem ein wenig zu gering ist. Wie unter 11.2 erläutert reicht eine Messrate von 1200Hz für die angestrebte Genauigkeit aus. Zudem wurde unter 11.1 als empfohlene Auslösegeschwindigkeit 750mm/s ermittelt. Damit die Ergebnisse beider Systeme trotzdem verglichen werden können, wird das mittels LabVIEW erstellte Programm auf diese nicht optimalen Parameter eingestellt.

Die einzelnen Nachlaufwege und -zeiten können der Abbildung 33 entnommen werden. Der resultierende Sicherheitsabstand nach der Auswertung kann der Tabelle 4 entnommen werden. Beide Messungen sind dabei mit eingebautem Werkzeug durchgeführt worden.

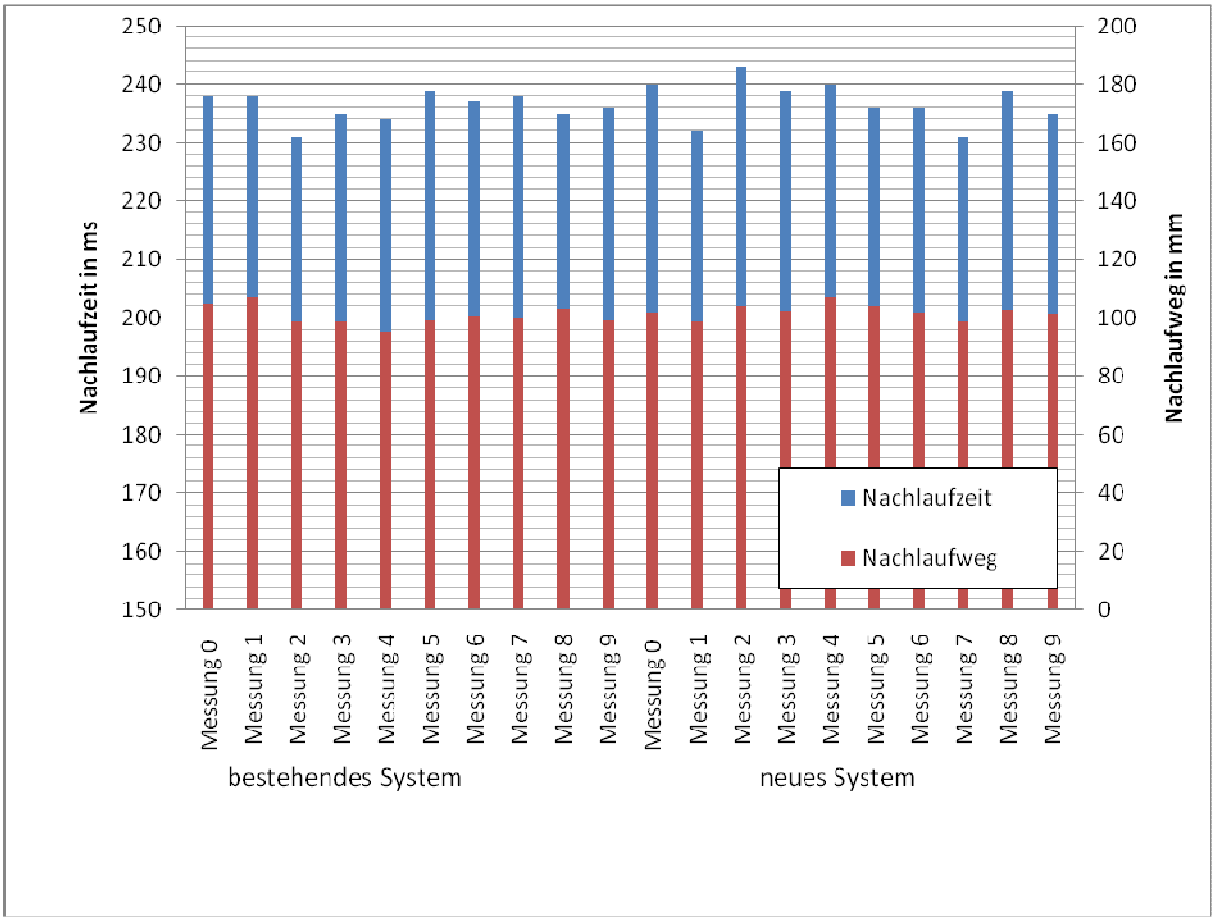


Abbildung 33: Die gemessenen Nachlaufzeiten und -wege mit dem bestehenden und dem neuen System

Tabelle 4: Ergebnisse einer Nachlaufwegmessung je System

	bestehendes System		neues System	
	Nachlaufzeit in ms	Nachlaufweg in mm	Nachlaufzeit in ms	Nachlaufweg In mm
Maximum	239	106,9	243	107,17
Mittelwert	236,1	100,68	237,1	102,31
Standardabweichung	2,42	3,46	3,78	2,55
zweithöchster Wert	238	104,9	240	104,1
Sicherheitsabstand in mm	446,4		500	

Die Ergebnisse der Messung sowohl der Nachlaufzeiten als auch der Nachlaufwege sind im Rahmen der Streuung durch die Presse identisch. Also kann angenommen werden, dass das mit LabVIEW erstellte Programm funktioniert. Nach der Auswertung, die auch die Schaltzeit des Lichtvorhangs mit

41ms berücksichtigt, unterscheidet sich allerdings der benötigte Sicherheitsabstand. Das liegt in der Auswertung nach DIN EN ISO 13855 begründet, die als Rechengrundlage die maximale oder eine höhere Nachlaufzeit verwendet, während die Anleitung des bestehenden Systems nur den zweithöchsten Wert als Ausgangswert berücksichtigt.

Der Vergleich bestätigt also, dass das mit LabVIEW erstellte System dem bestehenden System überlegen ist, da es komfortabler in der Bedienung ist, eine Auswertung nach DIN EN ISO 13855 durchführt und bei gleichen Einstellungen zu gleichen Einzelergebnissen führt. Zudem ist es auch für die Nutzung bei anderen Pressen verwendbar, da alle wichtigen Parameter komfortabel über ein Optionsmenü veränderbar sind.

11.5 Vorstellen der Ergebnisse einer Nachlaufwegmessung mit optimierten Parametern

Die unter 11.4 ausgewerteten Nachlaufzeiten und -wege sind nicht mit den unter Punkt 11.1 und 11.2 ermittelten optimalen Parametern durchgeführt worden. Daher können diese Ergebnisse nur dem Vergleich zwischen bestehenden System und neuem System dienen. Um nun eine verlässliche Aussage zum Nachlauf der vorliegenden Presse zu machen, wurde noch je eine Messung mit den empfohlenen Parametern mit und ohne Werkzeug durchgeführt. Hierbei kann zusätzlich beurteilt werden, ob die unter Punkt 11.1 bemerkten Unterschiede in der maximalen Geschwindigkeit sich auf die Ergebnisse der Messung auswirken. Die Ergebnisse sind der Abbildung 14 und der Tabelle 5 zu entnehmen.

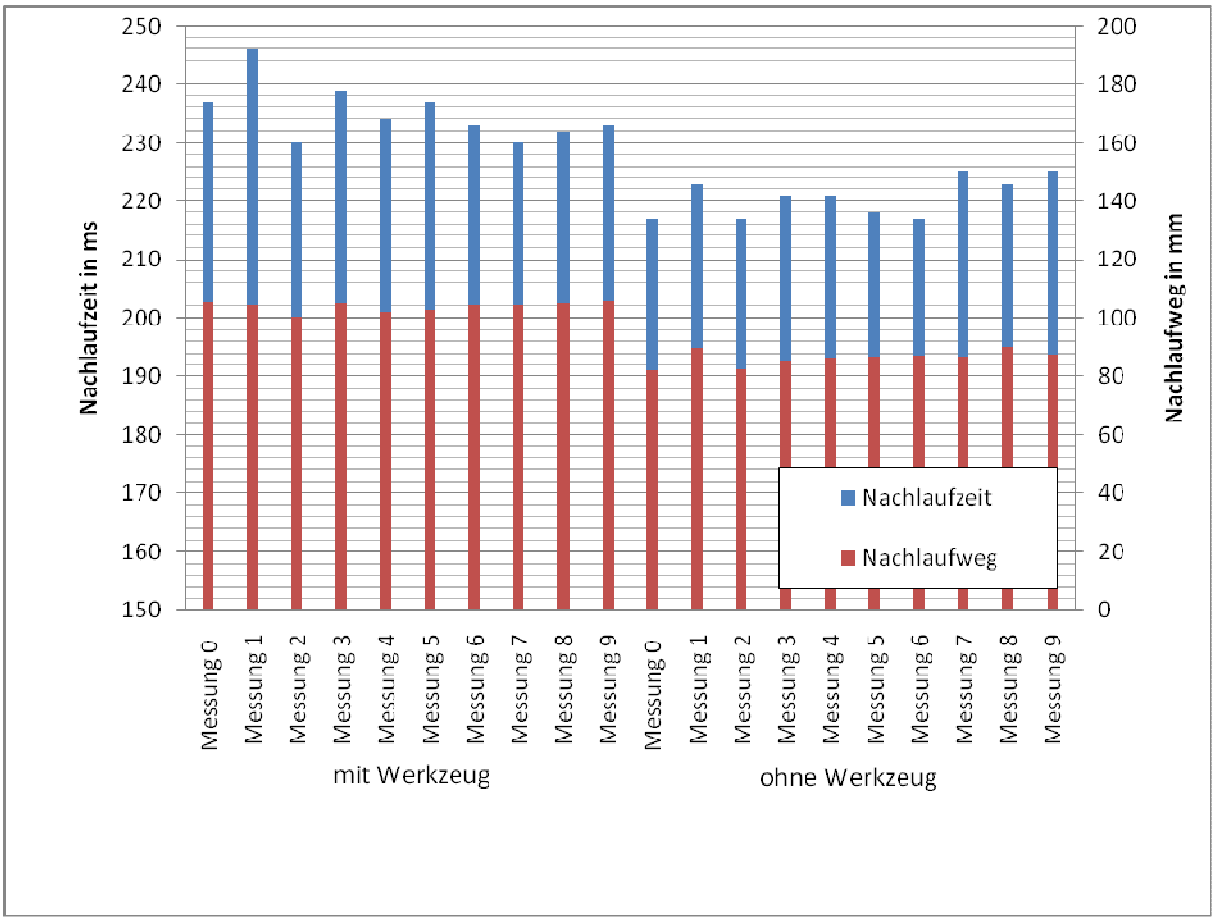


Abbildung 34: Nachlauf-Ergebnisse mit und ohne Werkzeug

Tabelle 5: Ergebnisse der Nachlaufwegmessung mit empfohlenen Parametern

	mit Werkzeug		ohne Werkzeug	
	Nachlaufzeit in ms	Nachlaufweg in mm	Nachlaufzeit in ms	Nachlaufweg in mm
Maximum	246	105,88	225	90,34
Mittelwert	235,1	104,012	220,7	86,387
Standardabweichung	4,86	1,74	3,27	2,62
Sicherheitsabstand in mm	500		500	

Bei der Messung ohne Werkzeug fällt auf, dass sowohl Nachlaufzeit als auch Nachlaufweg etwas kürzer ausfallen. Dies liegt an der deutlich geringeren Masse, die abgebremst werden muss, und an der geringeren Maximalgeschwindigkeit der Presse und entspricht den Erwartungen. Daher kann das Programm auch ohne Werkzeug genutzt werden. Die Auswertung nach DIN EN ISO 13855 ergibt allerdings in beiden Fällen ein Sicherheitsabstand des Lichtvorhangs von 500mm.

Quellenverzeichnis

Georgi, Wolfgang/Metin, Ergun: Einführung in LabVIEW, 3. Auflage, München, Carl Hanser Verlag, 2007

Hottinger Baldwin Messtechnik:PC Messelektronik/Spider8 Spider8-30 Spider8-01, Darmstadt, Hottinger Baldwin Messtechnik

Hottinger Baldwin Messtechnik: Spider8 DLL Help, Hottinger Baldwin Messtechnik, 2004

Hottinger Baldwin Mess- und Systemtechnik: Spider8 Setup Hilfe, Version 3.00 041099, Hottinger Baldwin Mess- und Systemtechnik, 1999

National Instruments Corporation: DIAdem Schnupperkurs/Effizient von Daten zu Ergebnissen

National Instruments Corporation: LabVIEW Basic I: Einführung/Kurshandbuch, Version 8.6, National Instruments Corporation, 2008

National Instruments Corporation: LabVIEW-Hilfe, National Instruments Corporation, 2010

Norm BGR 500/Kapitel 2.3, 2007, Carl Heymanns Verlag

Norm DIN EN ISO 13855:2010-10

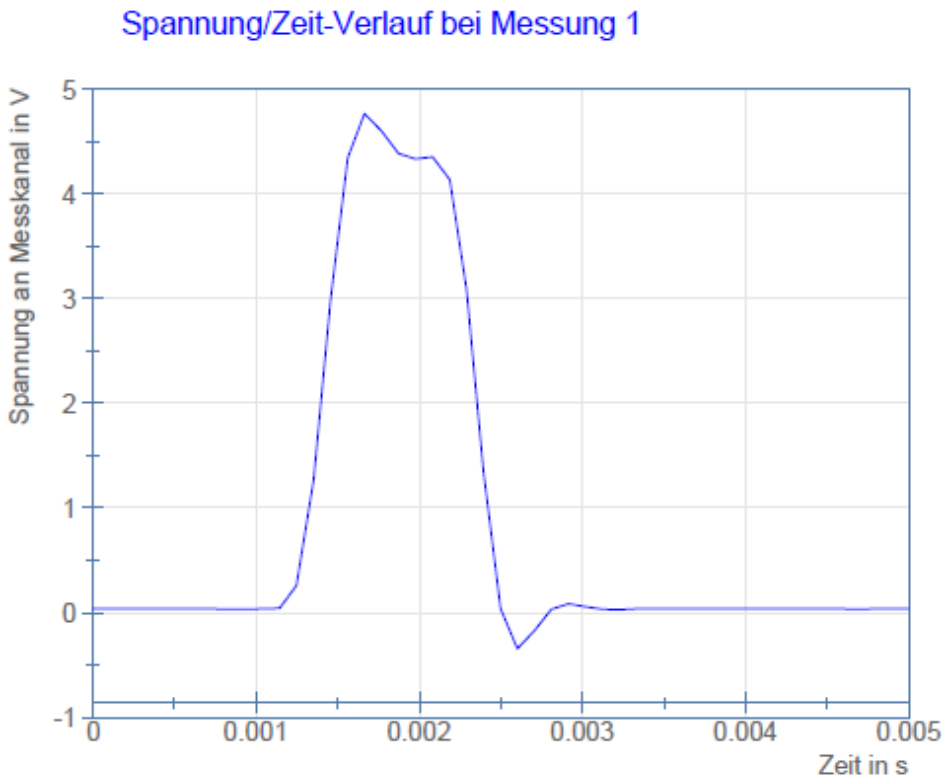
Norm VBG 7n5.2, 2005, Carl Heymanns Verlag

Norm ZH1/281, 2006, Carl Heymanns Verlag

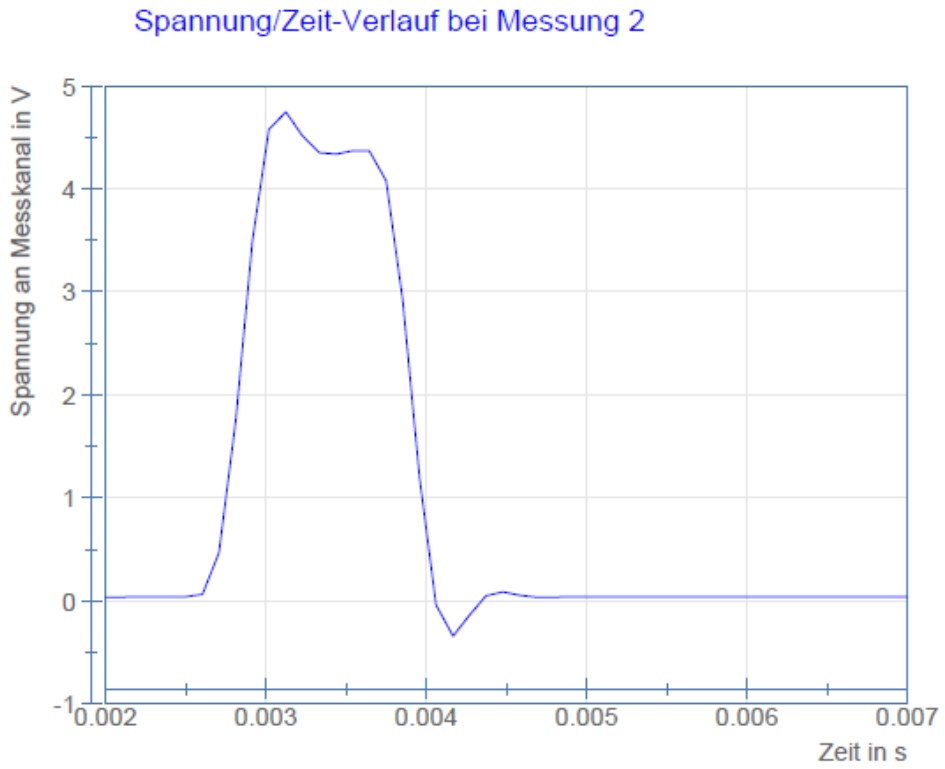
WayCon Positionsmesstechnik GmbH: MAGNETOSTRIKTIV/Magnetostriktiver Wegaufnehmer Serie MAB, www.waycon.de/fileadmin/pdf/Magnetostriktive_Geber_MAB.pdf, 01.02.2011

Anhang

Zu 10.2

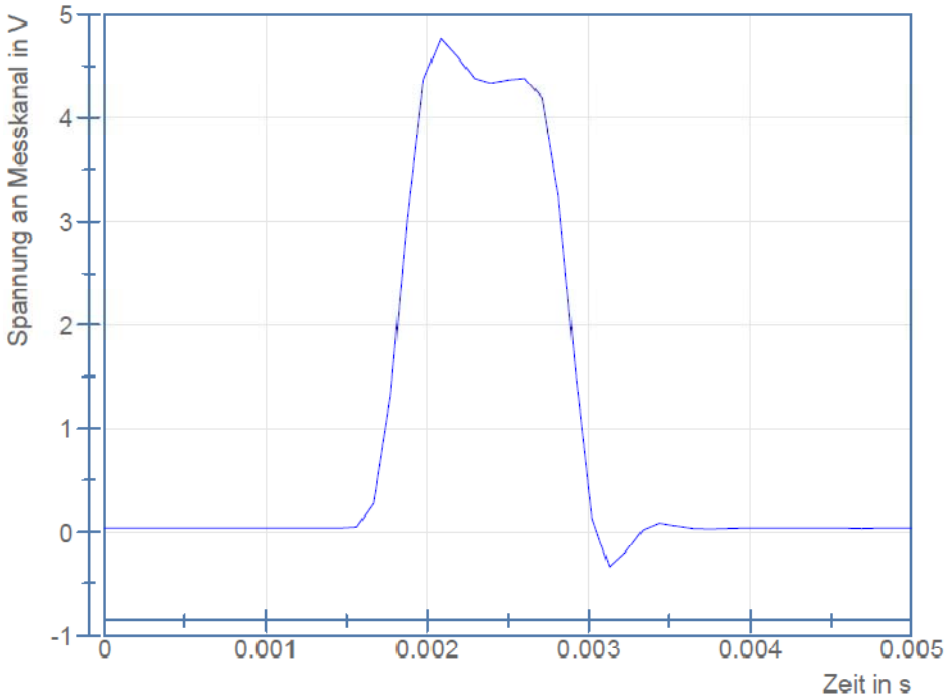


Zeitpunkt mit Beginn des Signals: 1,146ms
Zeitpunkt mit Ende des Signals: 2,084ms
Schaltzeit: 0.938ms



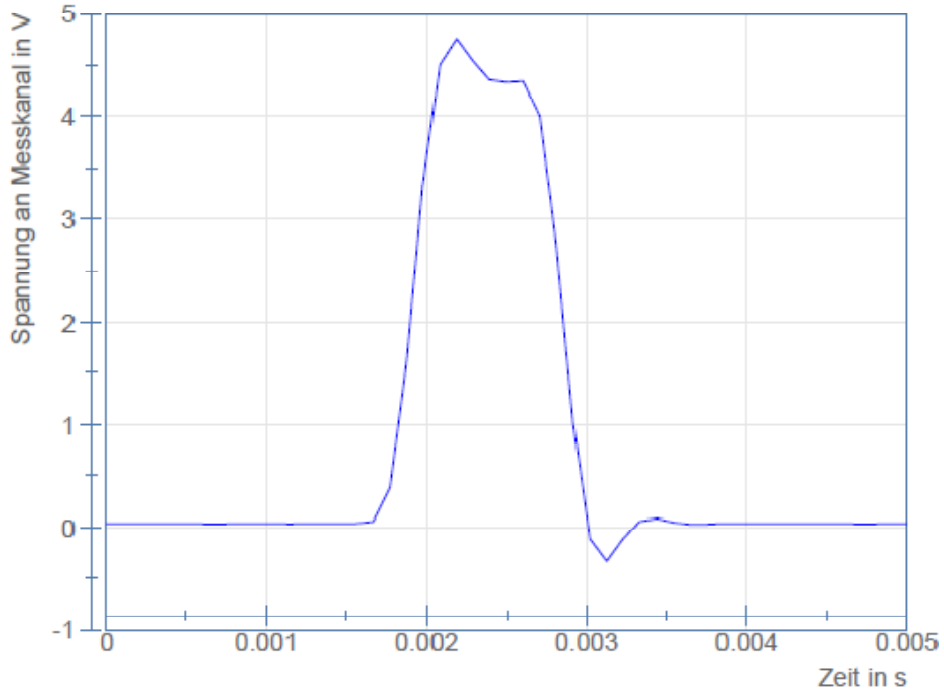
Zeitpunkt mit Beginn des Signals: 2.604ms
Zeitpunkt mit Ende des Signals: 3.646ms
Schaltzeit: 1.042ms

Spannung/Zeit-Verlauf bei Messung 3



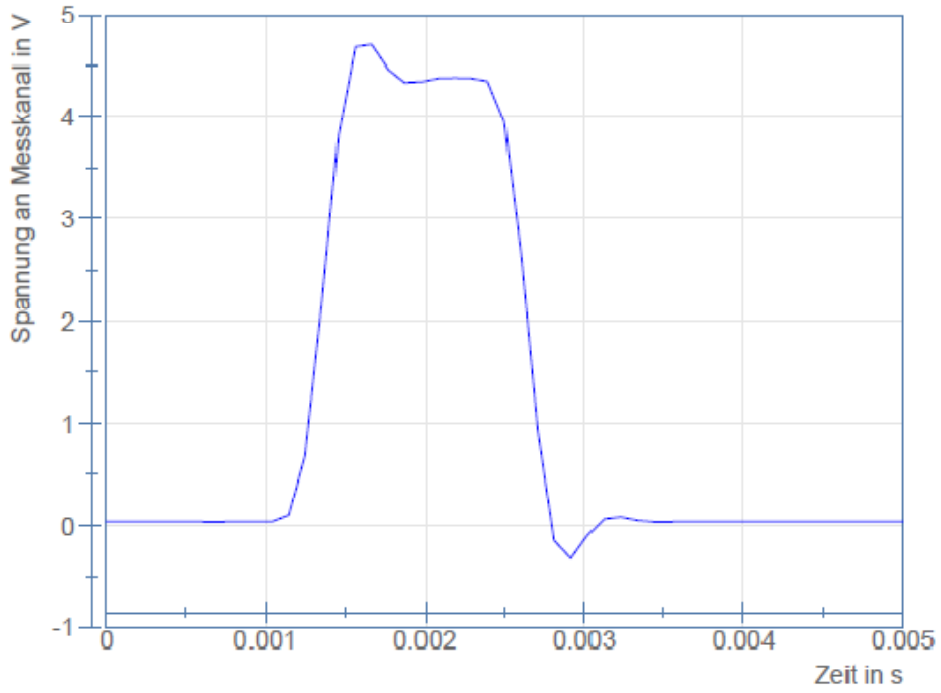
Zeitpunkt mit Beginn des Signals: 1.667ms
Zeitpunkt mit Ende des Signals: 2.709ms
Schaltzeit: 1.042ms

Spannung/Zeit-Verlauf bei Messung 4



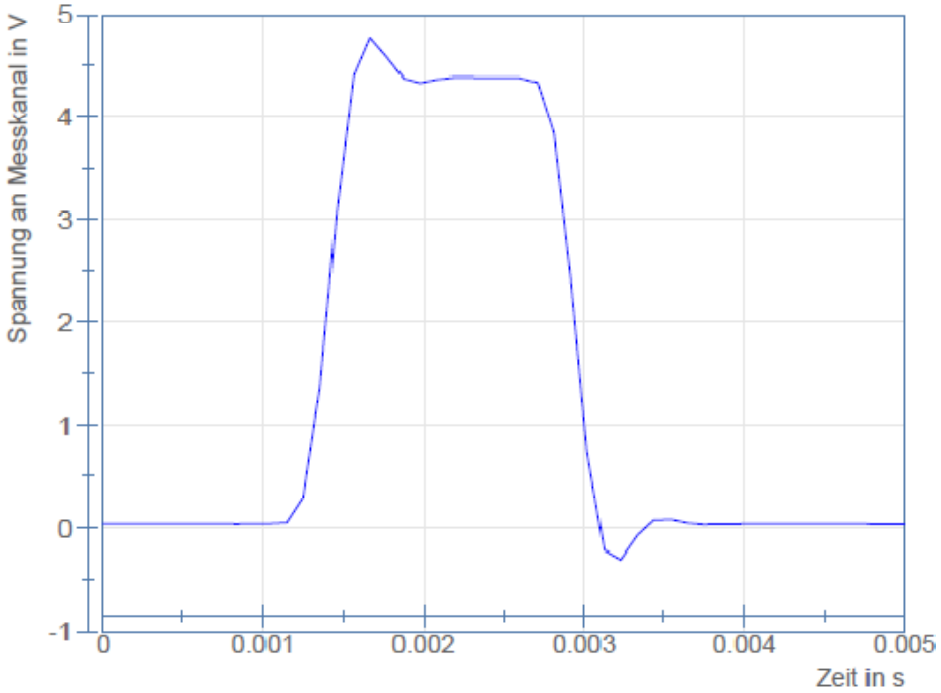
Zeitpunkt mit Beginn des Signals: 1.667ms
Zeitpunkt mit Ende des Signals: 2.605ms
Schaltzeit: 0.938ms

Spannung/Zeit-Verlauf bei Messung 5



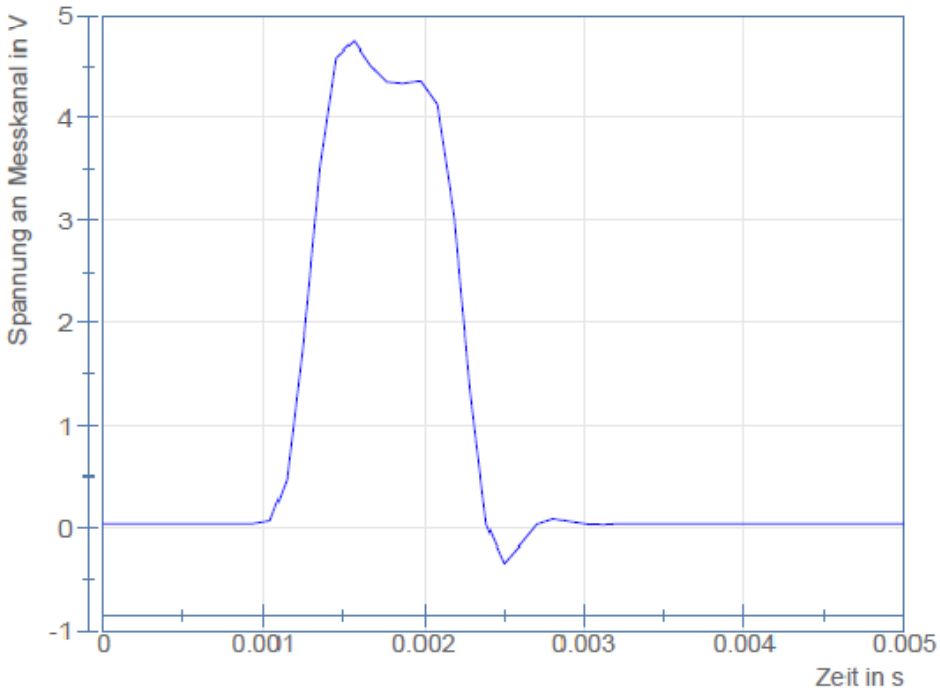
Zeitpunkt mit Beginn des Signals: 1.146ms
Zeitpunkt mit Ende des Signals: 2.396ms
Schaltzeit: 1.250ms

Spannung/Zeit-Verlauf bei Messung 6



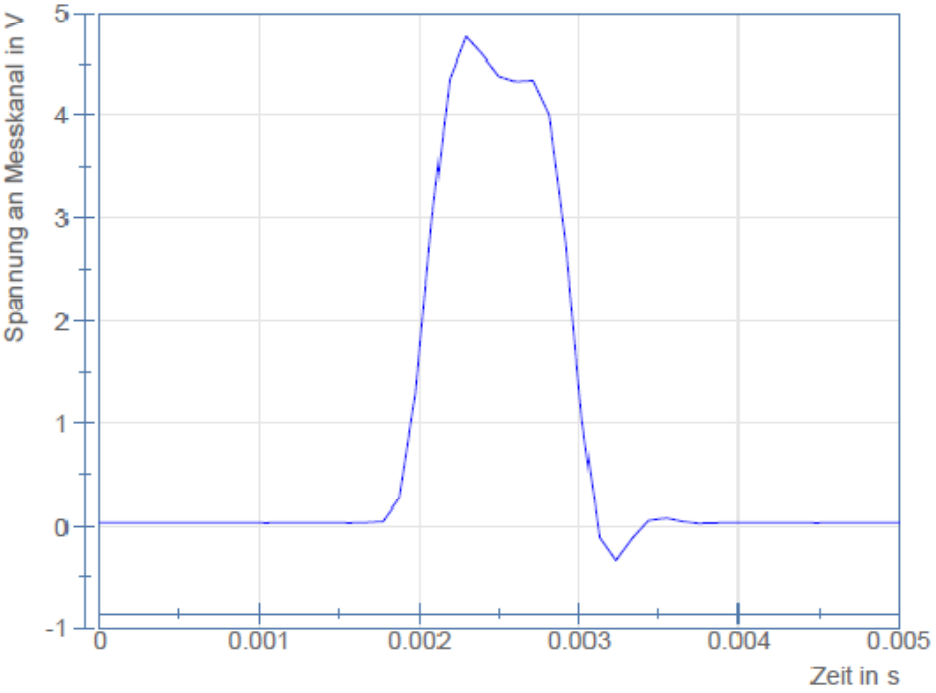
Zeitpunkt mit Beginn des Signals: 1.146ms
Zeitpunkt mit Ende des Signals: 2.709ms
Schaltzeit: 1.563ms

Spannung/Zeit-Verlauf bei Messung 7



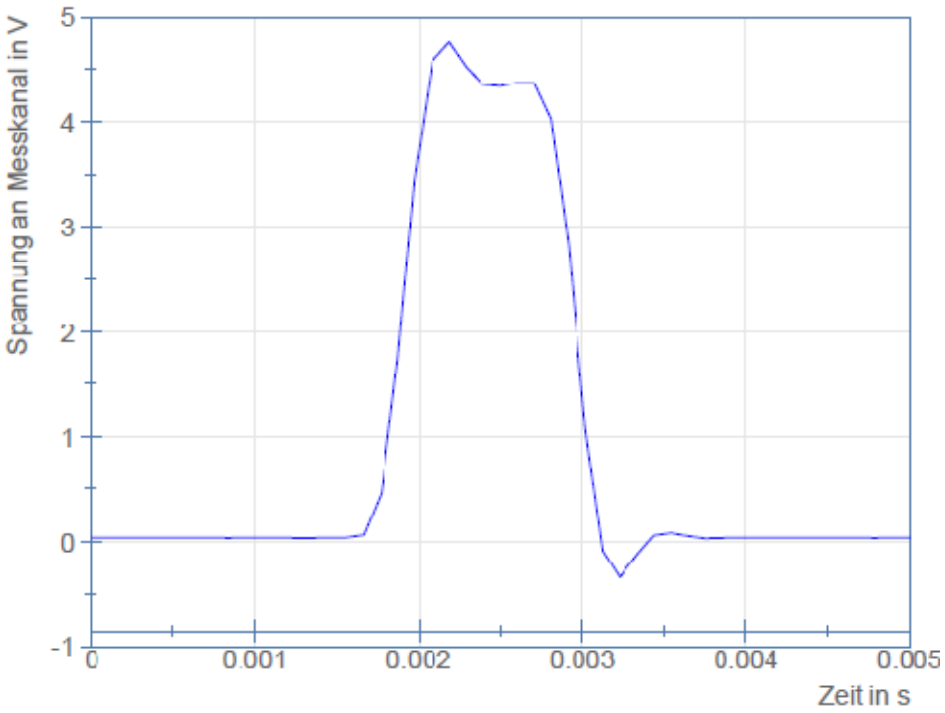
Zeitpunkt mit Beginn des Signals: 1.042ms
Zeitpunkt mit Ende des Signals: 1.980ms
Schaltzeit: 0.938ms

Spannung/Zeit-Verlauf bei Messung 8



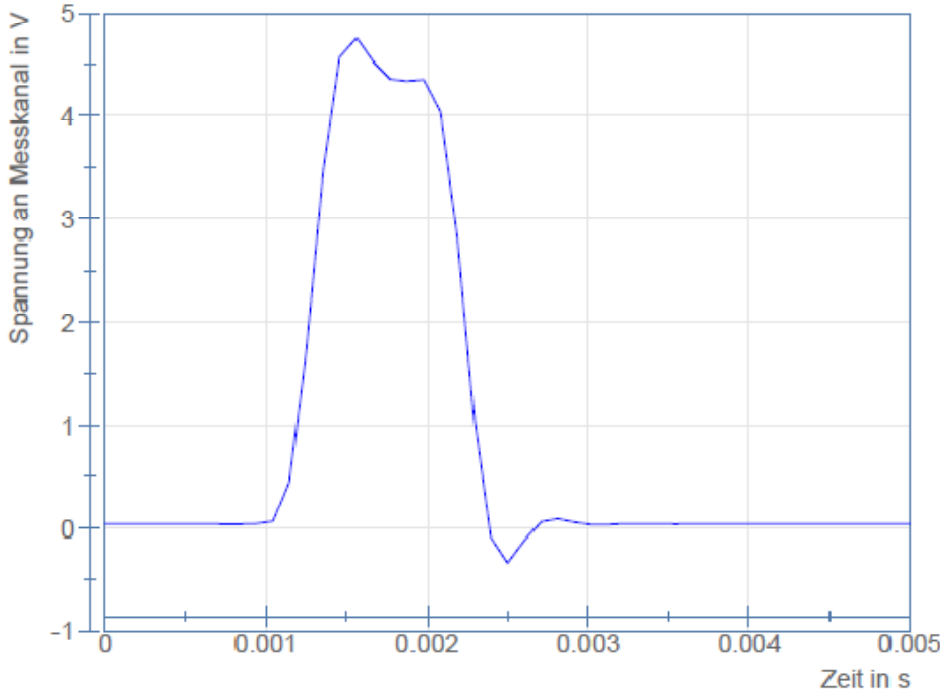
Zeitpunkt mit Beginn des Signals: 1.771ms
Zeitpunkt mit Ende des Signals: 2.709ms
Schaltzeit: 0.938ms

Spannung/Zeit-Verlauf bei Messung 9



Zeitpunk: mit Beginn des Signals: 1,667ms
Zeitpunk: mit Ende des Signals: 2,709ms
Schaltzeit: 1.042ms

Spannung/Zeit-Verlauf bei Messung 10



Zeitpunkt mit Beginn des Signals: 1.042ms
Zeitpunkt mit Ende des Signals: 1.980ms
Schaltzeit: 0.938ms

Zu 10.3

Mittelwert:	10,01ms		Programmmessrate:		100Hz
Maximum:	15,00ms		Gerätemessrate:		100Hz
Minimum:	6,00ms		erwartete Zeitdifferenz:		10ms
Ausreißer:	2		Einhalterate:		98,66%
Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz
10ms		510ms	9ms	1010ms	9ms
20ms	10ms	520ms	10ms	1020ms	10ms
30ms	10ms	530ms	10ms	1030ms	10ms
40ms	10ms	540ms	10ms	1040ms	10ms
50ms	10ms	550ms	10ms	1050ms	10ms
60ms	10ms	560ms	10ms	1060ms	10ms
70ms	10ms	570ms	10ms	1070ms	10ms
80ms	10ms	580ms	10ms	1080ms	10ms
90ms	10ms	591ms	11ms	1091ms	11ms
100ms	10ms	600ms	9ms	1100ms	9ms
110ms	10ms	610ms	10ms	1110ms	10ms
120ms	10ms	620ms	10ms	1120ms	10ms
130ms	10ms	630ms	10ms	1131ms	11ms
140ms	10ms	640ms	10ms	1140ms	9ms
150ms	10ms	650ms	10ms	1150ms	10ms
160ms	10ms	660ms	10ms	1160ms	10ms
171ms	11ms	671ms	11ms	1171ms	11ms
180ms	9ms	680ms	9ms	1180ms	9ms
190ms	10ms	690ms	10ms	1190ms	10ms
200ms	10ms	700ms	10ms	1200ms	10ms
211ms	11ms	711ms	11ms	1211ms	11ms
220ms	9ms	720ms	9ms	1220ms	9ms
230ms	10ms	730ms	10ms	1230ms	10ms
240ms	10ms	740ms	10ms	1240ms	10ms
251ms	11ms	751ms	11ms	1251ms	11ms
260ms	9ms	760ms	9ms	1260ms	9ms
270ms	10ms	770ms	10ms	1270ms	10ms
280ms	10ms	780ms	10ms	1280ms	10ms
290ms	10ms	790ms	10ms	1290ms	10ms
300ms	10ms	800ms	10ms	1300ms	10ms
310ms	10ms	810ms	10ms	1310ms	10ms
320ms	10ms	820ms	10ms	1320ms	10ms
330ms	10ms	830ms	10ms	1330ms	10ms
340ms	10ms	840ms	10ms	1341ms	11ms
350ms	10ms	850ms	10ms	1350ms	9ms
360ms	10ms	860ms	10ms	1360ms	10ms
370ms	10ms	870ms	10ms	1370ms	10ms
381ms	11ms	880ms	10ms	1381ms	11ms
390ms	9ms	890ms	10ms	1390ms	9ms
400ms	10ms	900ms	10ms	1400ms	10ms
410ms	10ms	910ms	10ms	1410ms	10ms
421ms	11ms	921ms	11ms	1421ms	11ms
430ms	9ms	930ms	9ms	1430ms	9ms
440ms	10ms	940ms	10ms	1440ms	10ms
450ms	10ms	950ms	10ms	1450ms	10ms
465ms	15ms	961ms	11ms	1461ms	11ms
471ms	6ms	970ms	9ms	1470ms	9ms
481ms	10ms	980ms	10ms	1480ms	10ms
490ms	9ms	990ms	10ms	1490ms	10ms
501ms	11ms	1001ms	11ms	1501ms	11ms

Mittelwert:	10,13ms		Programmmessrate:	100Hz
Maximum:	30,00ms		Gerätemessrate:	1200Hz
Minimum:	6,00ms		erwartete Zeitdifferenz:	10ms
Ausreißer:	3		Einhalterate:	97,99%

Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz
2ms		521ms	10ms	1021ms	10ms
12ms	10ms	531ms	10ms	1031ms	10ms
21ms	9ms	542ms	11ms	1042ms	11ms
31ms	10ms	551ms	9ms	1051ms	9ms
42ms	11ms	561ms	10ms	1061ms	10ms
51ms	9ms	571ms	10ms	1071ms	10ms
61ms	10ms	582ms	11ms	1082ms	11ms
71ms	10ms	591ms	9ms	1091ms	9ms
82ms	11ms	602ms	11ms	1101ms	10ms
91ms	9ms	611ms	9ms	1111ms	10ms
101ms	10ms	622ms	11ms	1122ms	11ms
111ms	10ms	631ms	9ms	1131ms	9ms
122ms	11ms	641ms	10ms	1141ms	10ms
131ms	9ms	651ms	10ms	1151ms	10ms
141ms	10ms	662ms	11ms	1162ms	11ms
151ms	10ms	671ms	9ms	1171ms	9ms
162ms	11ms	681ms	10ms	1182ms	11ms
171ms	9ms	691ms	10ms	1191ms	9ms
181ms	10ms	702ms	11ms	1202ms	11ms
191ms	10ms	711ms	9ms	1211ms	9ms
201ms	10ms	721ms	10ms	1221ms	10ms
211ms	10ms	731ms	10ms	1231ms	10ms
221ms	10ms	741ms	10ms	1241ms	10ms
231ms	10ms	751ms	10ms	1251ms	10ms
241ms	10ms	761ms	10ms	1261ms	10ms
251ms	10ms	771ms	10ms	1271ms	10ms
261ms	10ms	781ms	10ms	1281ms	10ms
271ms	10ms	792ms	11ms	1292ms	11ms
301ms	30ms	802ms	10ms	1301ms	9ms
311ms	10ms	811ms	9ms	1311ms	10ms
321ms	10ms	821ms	10ms	1321ms	10ms
332ms	11ms	835ms	14ms	1332ms	11ms
341ms	9ms	841ms	6ms	1341ms	9ms
351ms	10ms	851ms	10ms	1351ms	10ms
361ms	10ms	861ms	10ms	1361ms	10ms
372ms	11ms	872ms	11ms	1372ms	11ms
381ms	9ms	881ms	9ms	1381ms	9ms
391ms	10ms	891ms	10ms	1391ms	10ms
402ms	11ms	901ms	10ms	1401ms	10ms
412ms	10ms	912ms	11ms	1412ms	11ms
421ms	9ms	921ms	9ms	1421ms	9ms
431ms	10ms	931ms	10ms	1431ms	10ms
441ms	10ms	941ms	10ms	1441ms	10ms
452ms	11ms	952ms	11ms	1452ms	11ms
461ms	9ms	961ms	9ms	1461ms	9ms
471ms	10ms	971ms	10ms	1471ms	10ms
481ms	10ms	981ms	10ms	1481ms	10ms
491ms	10ms	991ms	10ms	1491ms	10ms
501ms	10ms	1001ms	10ms	1502ms	11ms
511ms	10ms	1011ms	10ms	1511ms	9ms

Mittelwert:	9,99ms		Programmmessrate:	100Hz
Maximum:	11,00ms		Gerätemessrate:	9600Hz
Minimum:	9,00ms		erwartete Zeitdifferenz:	10ms
Ausreißer:	0		Einhalterate:	100,00%

Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz
2ms		501ms	10ms	1001ms	10ms
12ms	10ms	511ms	10ms	1011ms	10ms
22ms	10ms	522ms	11ms	1022ms	11ms
31ms	9ms	531ms	9ms	1031ms	9ms
41ms	10ms	541ms	10ms	1041ms	10ms
51ms	10ms	551ms	10ms	1051ms	10ms
61ms	10ms	561ms	10ms	1061ms	10ms
71ms	10ms	572ms	11ms	1071ms	10ms
81ms	10ms	581ms	9ms	1081ms	10ms
91ms	10ms	591ms	10ms	1091ms	10ms
101ms	10ms	601ms	10ms	1101ms	10ms
112ms	11ms	612ms	11ms	1112ms	11ms
121ms	9ms	621ms	9ms	1121ms	9ms
131ms	10ms	631ms	10ms	1131ms	10ms
141ms	10ms	641ms	10ms	1141ms	10ms
152ms	11ms	652ms	11ms	1152ms	11ms
161ms	9ms	661ms	9ms	1161ms	9ms
171ms	10ms	671ms	10ms	1171ms	10ms
181ms	10ms	681ms	10ms	1181ms	10ms
191ms	10ms	692ms	11ms	1192ms	11ms
201ms	10ms	701ms	9ms	1201ms	9ms
211ms	10ms	711ms	10ms	1211ms	10ms
221ms	10ms	721ms	10ms	1221ms	10ms
232ms	11ms	732ms	11ms	1232ms	11ms
241ms	9ms	741ms	9ms	1241ms	9ms
251ms	10ms	751ms	10ms	1251ms	10ms
261ms	10ms	761ms	10ms	1261ms	10ms
272ms	11ms	772ms	11ms	1272ms	11ms
281ms	9ms	781ms	9ms	1281ms	9ms
291ms	10ms	791ms	10ms	1291ms	10ms
301ms	10ms	801ms	10ms	1301ms	10ms
311ms	10ms	811ms	10ms	1311ms	10ms
322ms	11ms	821ms	10ms	1321ms	10ms
331ms	9ms	831ms	10ms	1331ms	10ms
341ms	10ms	841ms	10ms	1341ms	10ms
351ms	10ms	851ms	10ms	1351ms	10ms
362ms	11ms	862ms	11ms	1362ms	11ms
371ms	9ms	871ms	9ms	1371ms	9ms
381ms	10ms	881ms	10ms	1381ms	10ms
391ms	10ms	891ms	10ms	1391ms	10ms
402ms	11ms	902ms	11ms	1402ms	11ms
411ms	9ms	911ms	9ms	1411ms	9ms
421ms	10ms	921ms	10ms	1421ms	10ms
431ms	10ms	931ms	10ms	1431ms	10ms
442ms	11ms	942ms	11ms	1442ms	11ms
451ms	9ms	951ms	9ms	1452ms	10ms
461ms	10ms	961ms	10ms	1461ms	9ms
471ms	10ms	971ms	10ms	1471ms	10ms
482ms	11ms	982ms	11ms	1482ms	11ms
491ms	9ms	991ms	9ms	1491ms	9ms

Mittelwert:	6,34ms		Programmmessrate:	200Hz
Maximum:	10,00ms		Gerätemessrate:	100Hz
Minimum:	4,00ms		erwartete Zeitdifferenz:	5ms
Ausreißer:	41		Einhalterate:	72,48%

Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz
2ms		308ms	6ms	627ms	4ms
9ms	7ms	318ms	10ms	637ms	10ms
17ms	8ms	327ms	9ms	642ms	5ms
22ms	5ms	332ms	5ms	647ms	5ms
27ms	5ms	337ms	5ms	652ms	5ms
37ms	10ms	343ms	6ms	662ms	10ms
43ms	6ms	352ms	9ms	667ms	5ms
47ms	4ms	357ms	5ms	673ms	6ms
52ms	5ms	362ms	5ms	677ms	4ms
62ms	10ms	367ms	5ms	687ms	10ms
67ms	5ms	377ms	10ms	693ms	6ms
73ms	6ms	382ms	5ms	698ms	5ms
77ms	4ms	387ms	5ms	708ms	10ms
83ms	6ms	392ms	5ms	712ms	4ms
93ms	10ms	402ms	10ms	718ms	6ms
97ms	4ms	407ms	5ms	723ms	5ms
102ms	5ms	413ms	6ms	728ms	5ms
107ms	5ms	417ms	4ms	737ms	9ms
117ms	10ms	423ms	6ms	742ms	5ms
123ms	6ms	428ms	5ms	748ms	6ms
128ms	5ms	433ms	5ms	752ms	4ms
133ms	5ms	438ms	5ms	757ms	5ms
142ms	9ms	447ms	9ms	767ms	10ms
147ms	5ms	457ms	10ms	772ms	5ms
153ms	6ms	462ms	5ms	782ms	10ms
158ms	5ms	468ms	6ms	792ms	10ms
167ms	9ms	477ms	9ms	797ms	5ms
172ms	5ms	483ms	6ms	802ms	5ms
177ms	5ms	487ms	4ms	807ms	5ms
182ms	5ms	492ms	5ms	817ms	10ms
192ms	10ms	498ms	6ms	823ms	6ms
197ms	5ms	508ms	10ms	828ms	5ms
203ms	6ms	512ms	4ms	838ms	10ms
207ms	4ms	518ms	6ms	843ms	5ms
213ms	6ms	527ms	9ms	848ms	5ms
218ms	5ms	537ms	10ms	853ms	5ms
223ms	5ms	542ms	5ms	863ms	10ms
228ms	5ms	547ms	5ms	872ms	9ms
232ms	4ms	552ms	5ms	877ms	5ms
237ms	5ms	562ms	10ms	882ms	5ms
247ms	10ms	567ms	5ms	888ms	6ms
253ms	6ms	572ms	5ms	897ms	9ms
257ms	4ms	577ms	5ms	903ms	6ms
262ms	5ms	587ms	10ms	907ms	4ms
272ms	10ms	592ms	5ms	912ms	5ms
277ms	5ms	597ms	5ms	922ms	10ms
283ms	6ms	607ms	10ms	927ms	5ms
288ms	5ms	612ms	5ms	932ms	5ms
297ms	9ms	617ms	5ms	937ms	5ms
302ms	5ms	623ms	6ms	947ms	10ms

Mittelwert:	5,07ms		Programmmessrate:	200Hz
Maximum:	16,00ms		Gerätemessrate:	1200Hz
Minimum:	4,00ms		erwartete Zeitdifferenz:	5ms
Ausreißer:	1		Einhalterate:	99,33%

Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz
4ms		254ms	4ms	504ms	5ms
10ms	6ms	259ms	5ms	509ms	5ms
15ms	5ms	264ms	5ms	514ms	5ms
19ms	4ms	269ms	5ms	519ms	5ms
25ms	6ms	274ms	5ms	524ms	5ms
29ms	4ms	279ms	5ms	529ms	5ms
34ms	5ms	284ms	5ms	534ms	5ms
39ms	5ms	289ms	5ms	539ms	5ms
44ms	5ms	294ms	5ms	544ms	5ms
49ms	5ms	299ms	5ms	549ms	5ms
54ms	5ms	304ms	5ms	554ms	5ms
59ms	5ms	309ms	5ms	559ms	5ms
64ms	5ms	314ms	5ms	564ms	5ms
70ms	6ms	320ms	6ms	570ms	6ms
74ms	4ms	324ms	4ms	574ms	4ms
79ms	5ms	329ms	5ms	579ms	5ms
84ms	5ms	334ms	5ms	584ms	5ms
89ms	5ms	339ms	5ms	589ms	5ms
94ms	5ms	344ms	5ms	594ms	5ms
99ms	5ms	349ms	5ms	610ms	16ms
104ms	5ms	354ms	5ms	614ms	4ms
110ms	6ms	359ms	5ms	619ms	5ms
114ms	4ms	364ms	5ms	624ms	5ms
119ms	5ms	370ms	6ms	629ms	5ms
124ms	5ms	374ms	4ms	634ms	5ms
129ms	5ms	379ms	5ms	639ms	5ms
134ms	5ms	384ms	5ms	644ms	5ms
139ms	5ms	389ms	5ms	649ms	5ms
144ms	5ms	394ms	5ms	655ms	6ms
149ms	5ms	399ms	5ms	659ms	4ms
154ms	5ms	404ms	5ms	664ms	5ms
159ms	5ms	409ms	5ms	669ms	5ms
164ms	5ms	414ms	5ms	674ms	5ms
169ms	5ms	419ms	5ms	679ms	5ms
174ms	5ms	424ms	5ms	684ms	5ms
179ms	5ms	429ms	5ms	689ms	5ms
184ms	5ms	434ms	5ms	695ms	6ms
189ms	5ms	439ms	5ms	699ms	4ms
195ms	6ms	445ms	6ms	704ms	5ms
200ms	5ms	449ms	4ms	709ms	5ms
204ms	4ms	454ms	5ms	714ms	5ms
210ms	6ms	459ms	5ms	719ms	5ms
214ms	4ms	464ms	5ms	724ms	5ms
219ms	5ms	469ms	5ms	729ms	5ms
224ms	5ms	474ms	5ms	735ms	6ms
229ms	5ms	479ms	5ms	739ms	4ms
235ms	6ms	485ms	6ms	744ms	5ms
240ms	5ms	489ms	4ms	749ms	5ms
244ms	4ms	494ms	5ms	754ms	5ms
250ms	6ms	499ms	5ms	759ms	5ms

Mittelwert:	5,00ms		Programmmessrate:		200Hz
Maximum:	6,00ms		Gerätemessrate:		9600Hz
Minimum:	4,00ms		erwartete Zeitdifferenz:		5ms
Ausreißer:	0		Einhalterate:		100,00%
Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz
3ms		253ms	4ms	503ms	5ms
9ms	6ms	258ms	5ms	508ms	5ms
13ms	4ms	263ms	5ms	513ms	5ms
18ms	5ms	268ms	5ms	518ms	5ms
24ms	6ms	274ms	6ms	524ms	6ms
28ms	4ms	278ms	4ms	528ms	4ms
33ms	5ms	283ms	5ms	533ms	5ms
38ms	5ms	288ms	5ms	538ms	5ms
43ms	5ms	293ms	5ms	543ms	5ms
48ms	5ms	298ms	5ms	548ms	5ms
53ms	5ms	303ms	5ms	553ms	5ms
58ms	5ms	308ms	5ms	558ms	5ms
64ms	6ms	314ms	6ms	564ms	6ms
68ms	4ms	319ms	5ms	568ms	4ms
73ms	5ms	323ms	4ms	573ms	5ms
78ms	5ms	329ms	6ms	578ms	5ms
83ms	5ms	333ms	4ms	583ms	5ms
88ms	5ms	338ms	5ms	588ms	5ms
93ms	5ms	343ms	5ms	593ms	5ms
98ms	5ms	348ms	5ms	598ms	5ms
103ms	5ms	353ms	5ms	603ms	5ms
108ms	5ms	358ms	5ms	609ms	6ms
113ms	5ms	363ms	5ms	614ms	5ms
118ms	5ms	368ms	5ms	618ms	4ms
123ms	5ms	373ms	5ms	624ms	6ms
128ms	5ms	378ms	5ms	628ms	4ms
133ms	5ms	383ms	5ms	633ms	5ms
138ms	5ms	388ms	5ms	638ms	5ms
143ms	5ms	393ms	5ms	643ms	5ms
149ms	6ms	399ms	6ms	649ms	6ms
153ms	4ms	404ms	5ms	654ms	5ms
158ms	5ms	408ms	4ms	658ms	4ms
163ms	5ms	414ms	6ms	664ms	6ms
168ms	5ms	418ms	4ms	668ms	4ms
173ms	5ms	423ms	5ms	673ms	5ms
178ms	5ms	428ms	5ms	678ms	5ms
183ms	5ms	433ms	5ms	683ms	5ms
189ms	6ms	439ms	6ms	689ms	6ms
194ms	5ms	444ms	5ms	694ms	5ms
198ms	4ms	448ms	4ms	698ms	4ms
204ms	6ms	454ms	6ms	704ms	6ms
208ms	4ms	458ms	4ms	708ms	4ms
213ms	5ms	463ms	5ms	713ms	5ms
218ms	5ms	468ms	5ms	718ms	5ms
223ms	5ms	473ms	5ms	723ms	5ms
228ms	5ms	478ms	5ms	728ms	5ms
233ms	5ms	483ms	5ms	733ms	5ms
238ms	5ms	488ms	5ms	738ms	5ms
243ms	5ms	493ms	5ms	743ms	5ms
249ms	6ms	498ms	5ms	748ms	5ms

Mittelwert:	4,60ms		Programmmessrate:	1000Hz	
Maximum:	23,00ms		Gerätemessrate:	100Hz	
Minimum:	2,00ms		erwartete Zeitdifferenz:	1ms	
Ausreißer:	143		Einhalterate:	4,03%	
Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz
2ms		260ms	23ms	481ms	5ms
15ms	13ms	270ms	10ms	485ms	4ms
20ms	5ms	274ms	4ms	489ms	4ms
22ms	2ms	280ms	6ms	494ms	5ms
29ms	7ms	282ms	2ms	497ms	3ms
36ms	7ms	288ms	6ms	502ms	5ms
42ms	6ms	292ms	4ms	507ms	5ms
46ms	4ms	296ms	4ms	510ms	3ms
54ms	8ms	300ms	4ms	515ms	5ms
57ms	3ms	305ms	5ms	520ms	5ms
62ms	5ms	308ms	3ms	522ms	2ms
66ms	4ms	314ms	6ms	527ms	5ms
70ms	4ms	317ms	3ms	532ms	5ms
74ms	4ms	321ms	4ms	535ms	3ms
78ms	4ms	326ms	5ms	539ms	4ms
82ms	4ms	330ms	4ms	544ms	5ms
87ms	5ms	334ms	4ms	547ms	3ms
91ms	4ms	338ms	4ms	552ms	5ms
95ms	4ms	343ms	5ms	557ms	5ms
99ms	4ms	346ms	3ms	561ms	4ms
105ms	6ms	352ms	6ms	564ms	3ms
112ms	7ms	355ms	3ms	568ms	4ms
117ms	5ms	359ms	4ms	573ms	5ms
124ms	7ms	361ms	2ms	576ms	3ms
128ms	4ms	368ms	7ms	582ms	6ms
131ms	3ms	372ms	4ms	586ms	4ms
137ms	6ms	376ms	4ms	590ms	4ms
142ms	5ms	381ms	5ms	595ms	5ms
146ms	4ms	384ms	3ms	599ms	4ms
149ms	3ms	388ms	4ms	603ms	4ms
154ms	5ms	394ms	6ms	607ms	4ms
158ms	4ms	396ms	2ms	610ms	3ms
162ms	4ms	401ms	5ms	615ms	5ms
167ms	5ms	406ms	5ms	620ms	5ms
171ms	4ms	409ms	3ms	623ms	3ms
175ms	4ms	414ms	5ms	627ms	4ms
180ms	5ms	418ms	4ms	633ms	6ms
184ms	4ms	422ms	4ms	636ms	3ms
188ms	4ms	426ms	4ms	640ms	4ms
192ms	4ms	432ms	6ms	646ms	6ms
195ms	3ms	438ms	6ms	652ms	6ms
200ms	5ms	443ms	5ms	657ms	5ms
205ms	5ms	447ms	4ms	661ms	4ms
208ms	3ms	451ms	4ms	664ms	3ms
213ms	5ms	456ms	5ms	666ms	2ms
218ms	5ms	460ms	4ms	670ms	4ms
226ms	8ms	464ms	4ms	674ms	4ms
230ms	4ms	468ms	4ms	677ms	3ms
233ms	3ms	473ms	5ms	683ms	6ms
237ms	4ms	476ms	3ms	687ms	4ms

Mittelwert:	3,69ms		Programmmessrate:	1000Hz
Maximum:	8,00ms		Gerätemessrate:	1200Hz
Minimum:	1,00ms		erwartete Zeitdifferenz:	1ms
Ausreißer:	136		Einhalterate:	8,72%

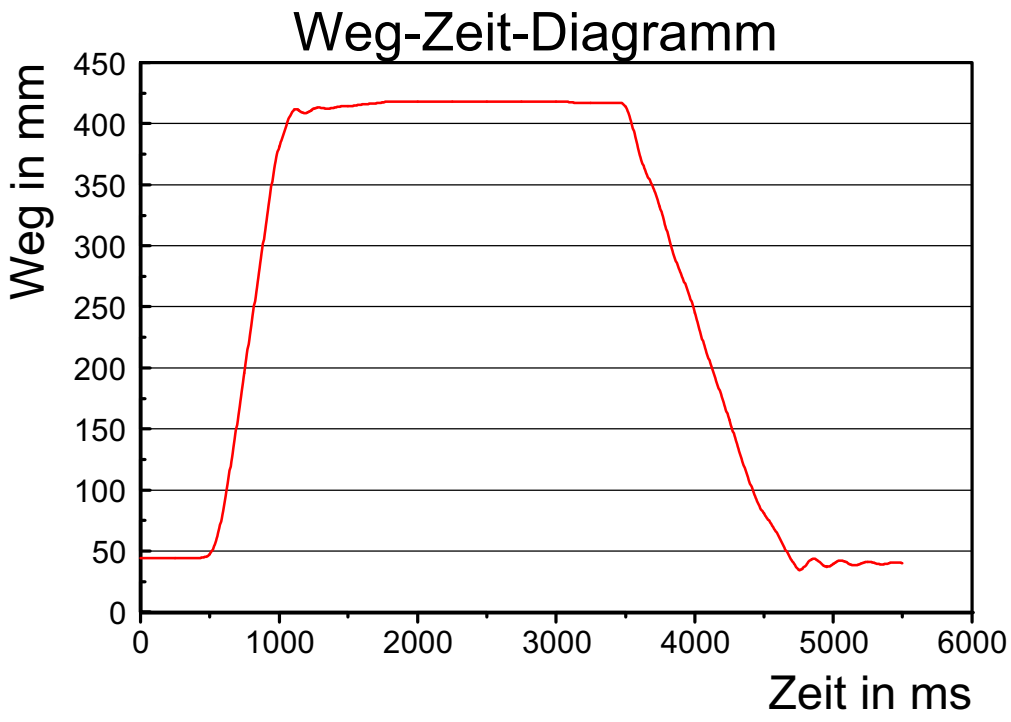
Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz
2ms		191ms	6ms	372ms	3ms
10ms	8ms	194ms	3ms	377ms	5ms
17ms	7ms	197ms	3ms	380ms	3ms
20ms	3ms	202ms	5ms	385ms	5ms
24ms	4ms	205ms	3ms	390ms	5ms
28ms	4ms	208ms	3ms	394ms	4ms
30ms	2ms	211ms	3ms	397ms	3ms
33ms	3ms	214ms	3ms	399ms	2ms
36ms	3ms	218ms	4ms	404ms	5ms
40ms	4ms	223ms	5ms	407ms	3ms
43ms	3ms	226ms	3ms	411ms	4ms
47ms	4ms	229ms	3ms	412ms	1ms
52ms	5ms	231ms	2ms	417ms	5ms
55ms	3ms	236ms	5ms	421ms	4ms
58ms	3ms	240ms	4ms	424ms	3ms
60ms	2ms	243ms	3ms	429ms	5ms
65ms	5ms	247ms	4ms	432ms	3ms
68ms	3ms	251ms	4ms	435ms	3ms
71ms	3ms	254ms	3ms	441ms	6ms
77ms	6ms	259ms	5ms	445ms	4ms
80ms	3ms	263ms	4ms	448ms	3ms
83ms	3ms	266ms	3ms	452ms	4ms
88ms	5ms	268ms	2ms	456ms	4ms
92ms	4ms	273ms	5ms	460ms	4ms
95ms	3ms	276ms	3ms	462ms	2ms
100ms	5ms	280ms	4ms	467ms	5ms
104ms	4ms	282ms	2ms	470ms	3ms
107ms	3ms	286ms	4ms	474ms	4ms
111ms	4ms	290ms	4ms	479ms	5ms
115ms	4ms	293ms	3ms	482ms	3ms
118ms	3ms	298ms	5ms	484ms	2ms
121ms	3ms	302ms	4ms	487ms	3ms
123ms	2ms	305ms	3ms	492ms	5ms
127ms	4ms	310ms	5ms	495ms	3ms
130ms	3ms	313ms	3ms	498ms	3ms
134ms	4ms	316ms	3ms	501ms	3ms
139ms	5ms	322ms	6ms	504ms	3ms
143ms	4ms	326ms	4ms	507ms	3ms
146ms	3ms	328ms	2ms	510ms	3ms
151ms	5ms	333ms	5ms	514ms	4ms
155ms	4ms	336ms	3ms	517ms	3ms
158ms	3ms	339ms	3ms	520ms	3ms
162ms	4ms	343ms	4ms	525ms	5ms
166ms	4ms	346ms	3ms	528ms	3ms
170ms	4ms	349ms	3ms	532ms	4ms
172ms	2ms	354ms	5ms	536ms	4ms
176ms	4ms	358ms	4ms	540ms	4ms
179ms	3ms	361ms	3ms	543ms	3ms
183ms	4ms	366ms	5ms	548ms	5ms
185ms	2ms	369ms	3ms	552ms	4ms

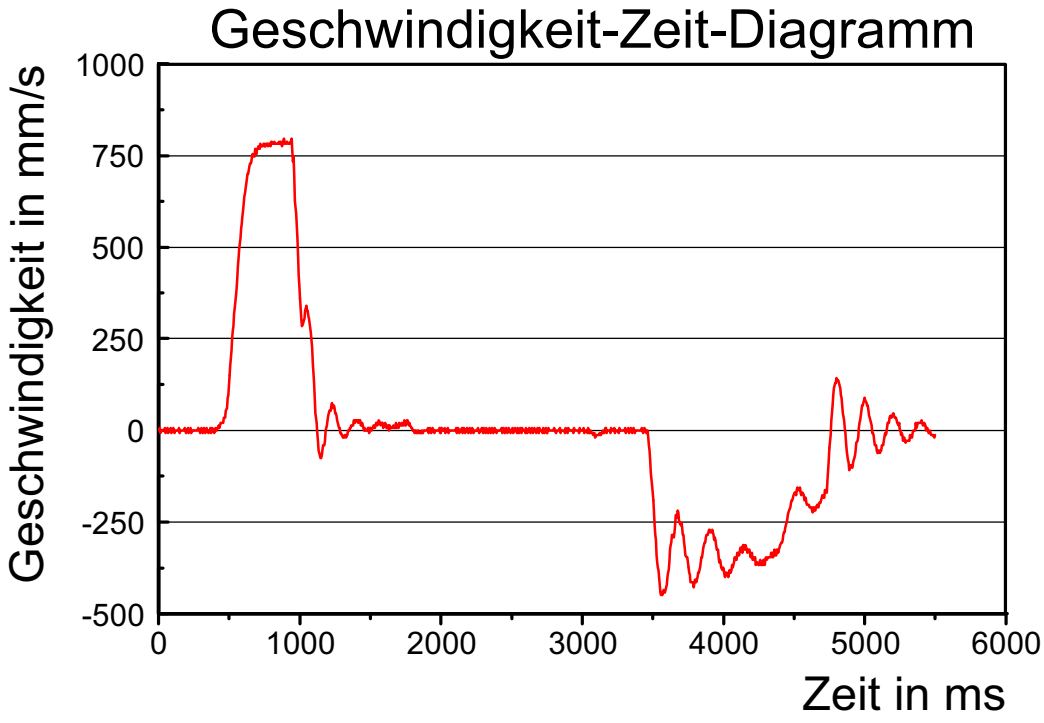
Mittelwert:	3,42ms		Programmmessrate:	1000Hz
Maximum:	7,00ms		Gerätemessrate:	9600Hz
Minimum:	2,00ms		erwartete Zeitdifferenz:	1ms
Ausreißer:	121		Einhalterate:	18,79%

Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz	Timerwert	Zeitdifferenz
2ms		180ms	2ms	351ms	4ms
9ms	7ms	185ms	5ms	354ms	3ms
16ms	7ms	188ms	3ms	356ms	2ms
19ms	3ms	191ms	3ms	360ms	4ms
23ms	4ms	194ms	3ms	363ms	3ms
25ms	2ms	199ms	5ms	366ms	3ms
29ms	4ms	202ms	3ms	369ms	3ms
32ms	3ms	205ms	3ms	374ms	5ms
35ms	3ms	207ms	2ms	377ms	3ms
40ms	5ms	212ms	5ms	380ms	3ms
43ms	3ms	215ms	3ms	382ms	2ms
47ms	4ms	218ms	3ms	387ms	5ms
49ms	2ms	220ms	2ms	390ms	3ms
54ms	5ms	226ms	6ms	393ms	3ms
58ms	4ms	229ms	3ms	395ms	2ms
61ms	3ms	232ms	3ms	400ms	5ms
63ms	2ms	236ms	4ms	402ms	2ms
67ms	4ms	239ms	3ms	406ms	4ms
71ms	4ms	242ms	3ms	411ms	5ms
74ms	3ms	247ms	5ms	414ms	3ms
79ms	5ms	250ms	3ms	417ms	3ms
83ms	4ms	253ms	3ms	419ms	2ms
87ms	4ms	255ms	2ms	424ms	5ms
91ms	4ms	260ms	5ms	427ms	3ms
95ms	4ms	263ms	3ms	430ms	3ms
98ms	3ms	266ms	3ms	432ms	2ms
100ms	2ms	271ms	5ms	437ms	5ms
106ms	6ms	274ms	3ms	440ms	3ms
109ms	3ms	276ms	2ms	442ms	2ms
111ms	2ms	279ms	3ms	445ms	3ms
114ms	3ms	284ms	5ms	450ms	5ms
119ms	5ms	287ms	3ms	453ms	3ms
122ms	3ms	290ms	3ms	456ms	3ms
125ms	3ms	295ms	5ms	458ms	2ms
127ms	2ms	298ms	3ms	463ms	5ms
132ms	5ms	301ms	3ms	466ms	3ms
135ms	3ms	305ms	4ms	470ms	4ms
138ms	3ms	308ms	3ms	472ms	2ms
140ms	2ms	311ms	3ms	477ms	5ms
145ms	5ms	314ms	3ms	480ms	3ms
148ms	3ms	316ms	2ms	483ms	3ms
151ms	3ms	320ms	4ms	486ms	3ms
153ms	2ms	323ms	3ms	491ms	5ms
158ms	5ms	326ms	3ms	494ms	3ms
161ms	3ms	328ms	2ms	497ms	3ms
164ms	3ms	333ms	5ms	499ms	2ms
166ms	2ms	336ms	3ms	504ms	5ms
171ms	5ms	340ms	4ms	507ms	3ms
175ms	4ms	342ms	2ms	510ms	3ms
178ms	3ms	347ms	5ms	512ms	2ms

zu 11.1

Messung 0 mit Werkzeug

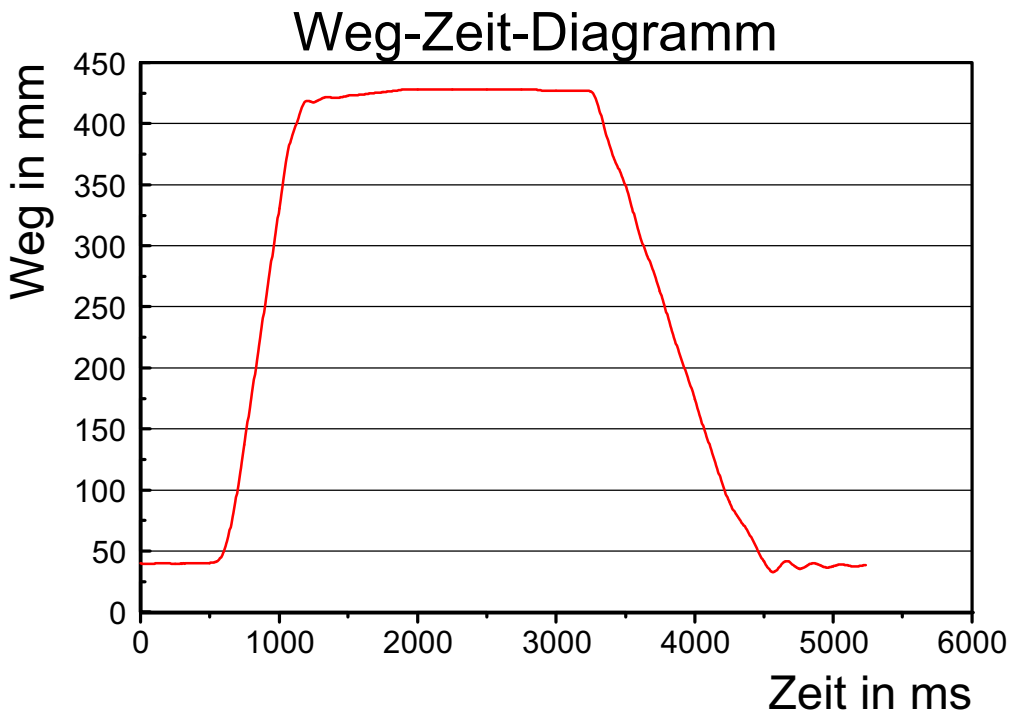


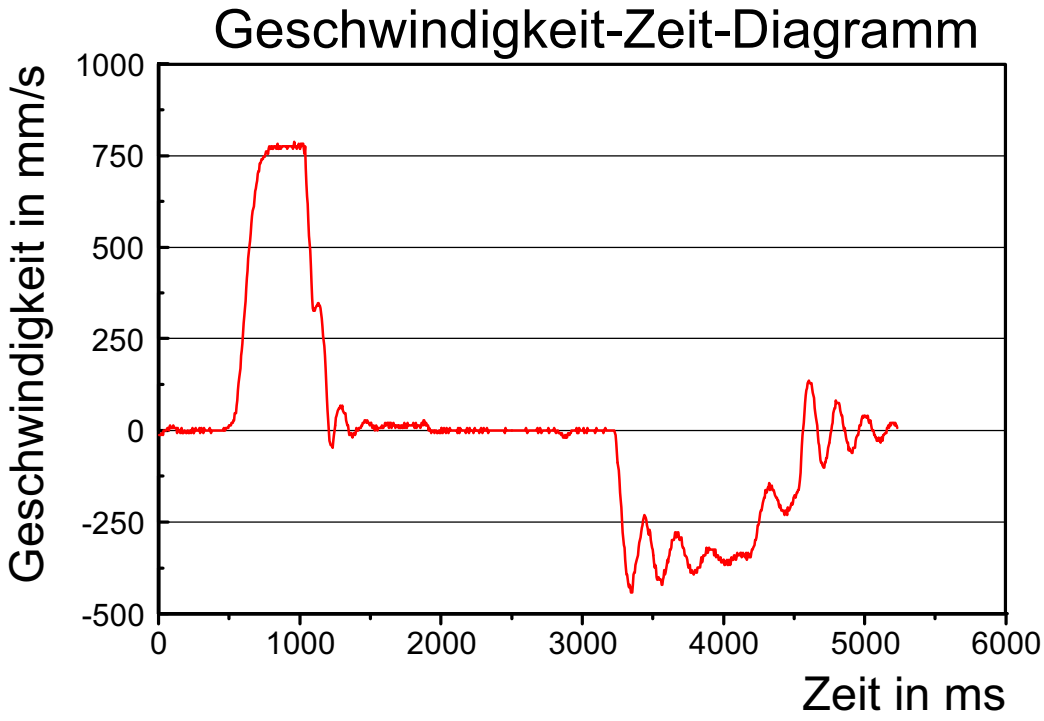


Maximalgeschwindigkeit: 782mm/s

Beschleunigungsweg: 132.09mm

Messung 1 mit Werkzeug

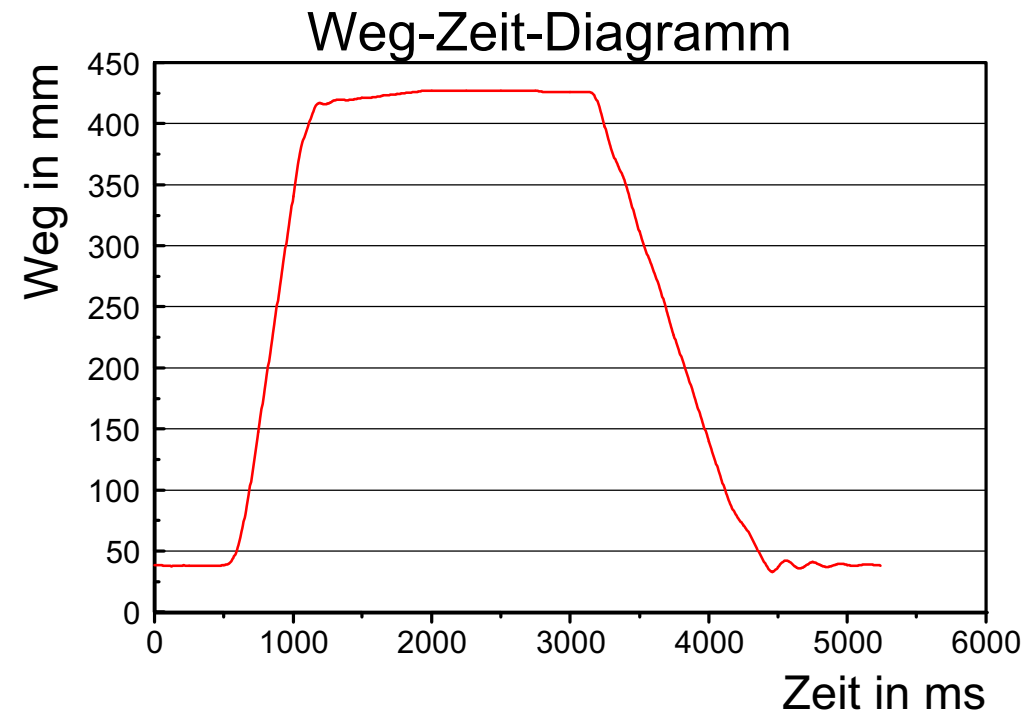


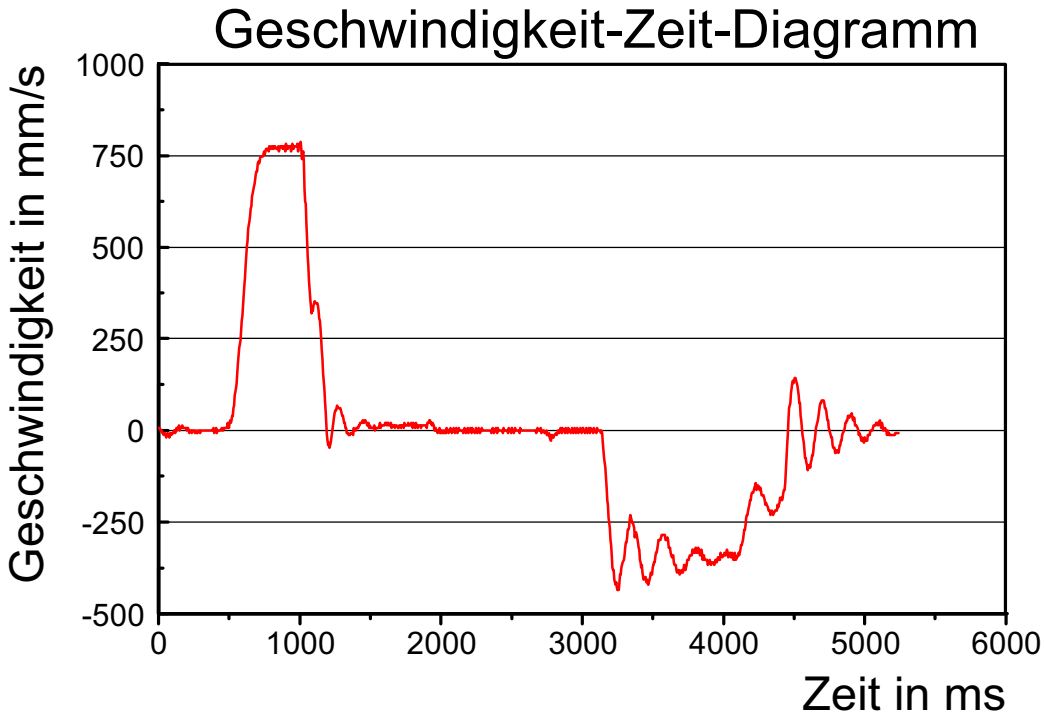


Maximalgeschwindigkeit: 775mm/s

Beschleunigungsweg: 123.73mm

Messung 2 mit Werkzeug

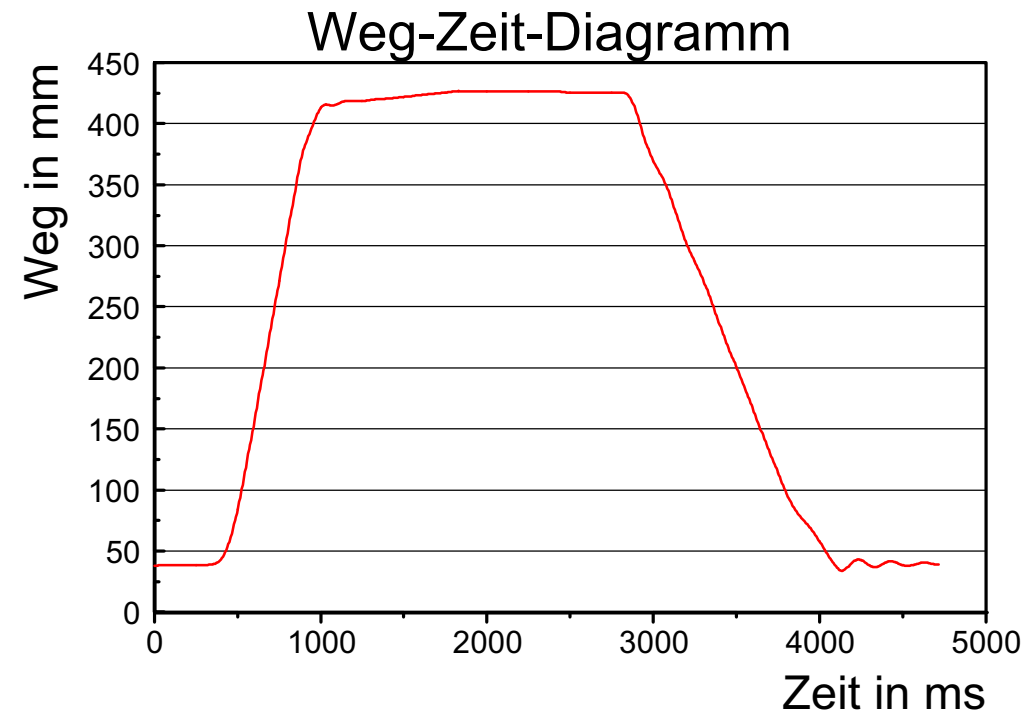


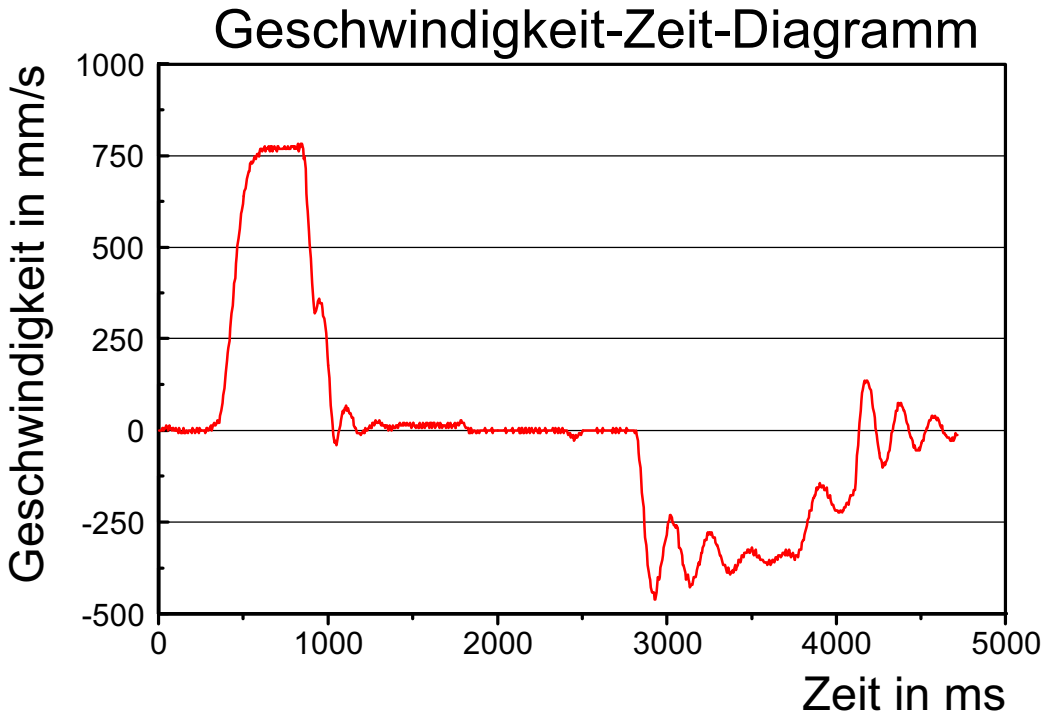


Maximalgeschwindigkeit: 775mm/s

Beschleunigungsweg: 136.13mm

Messung 3 mit Werkzeug

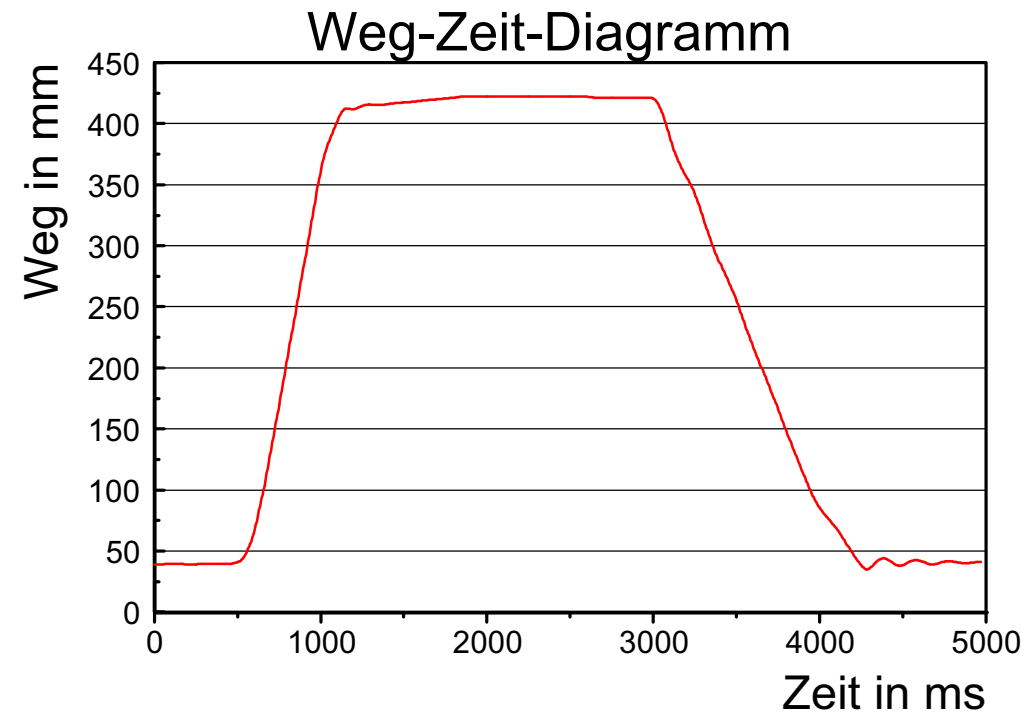


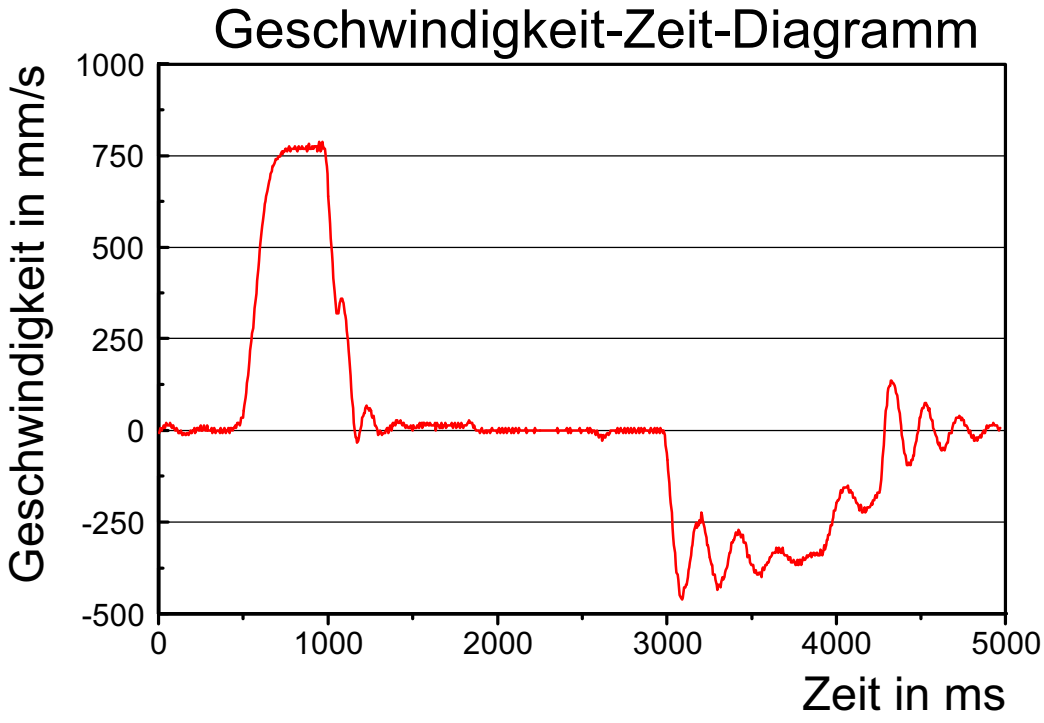


Maximalgeschwindigkeit: 768mm/s

Beschleunigungsweg: 121.00mm

Messung 4 mit Werkzeug

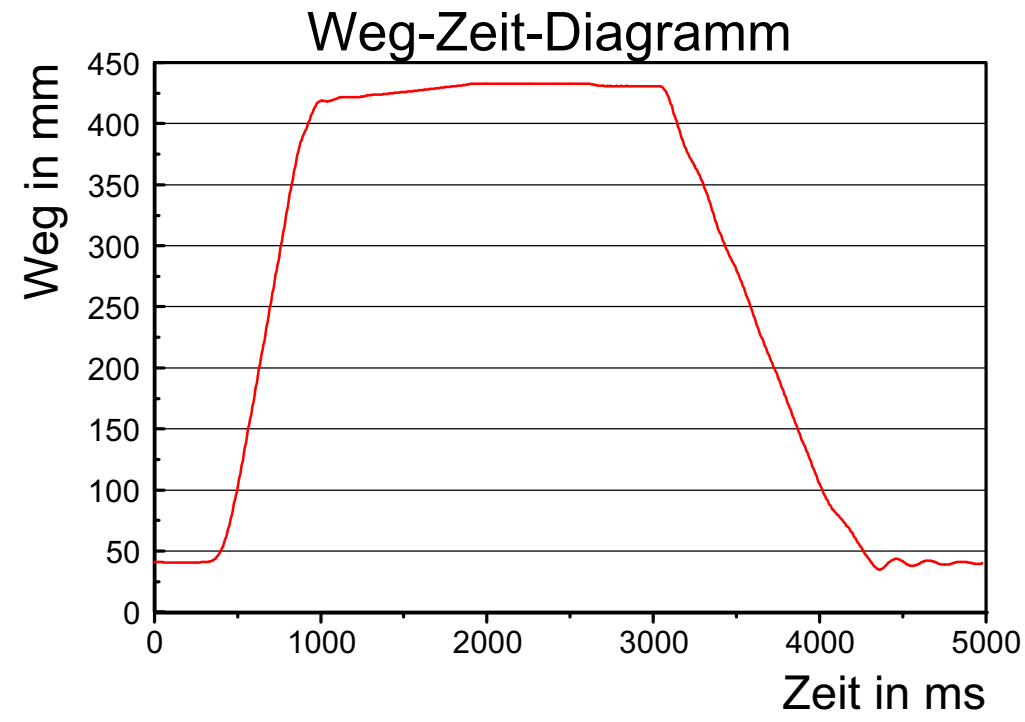


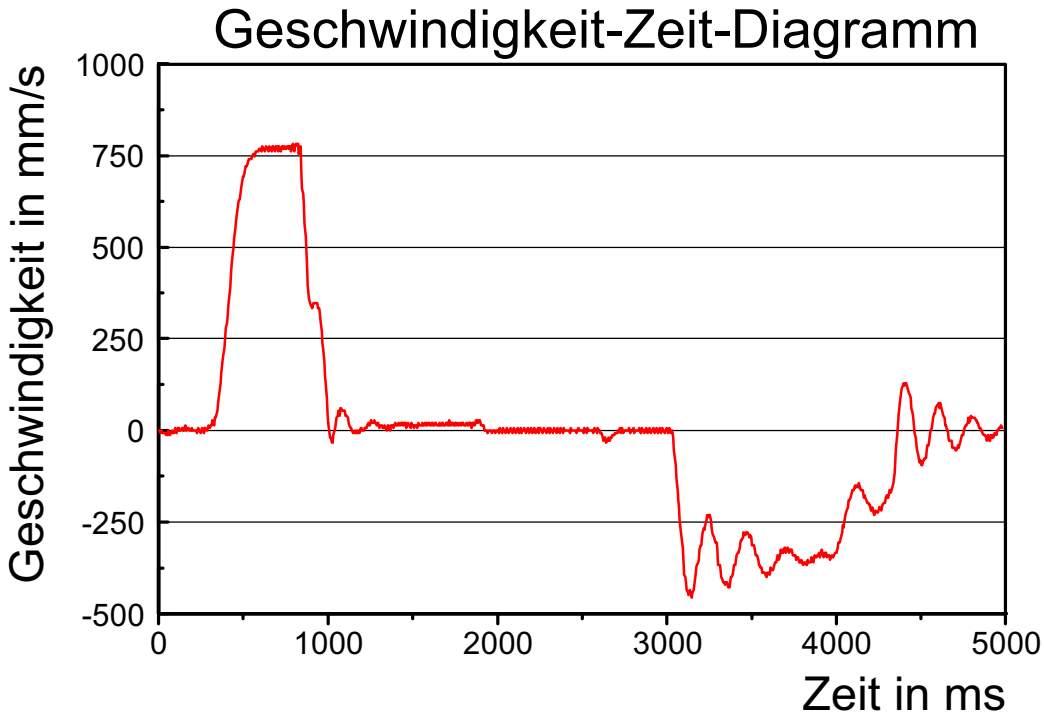


Maximalgeschwindigkeit: 768mm/s

Beschleunigungsweg: 130.32mm

Messung 5 mit Werkzeug

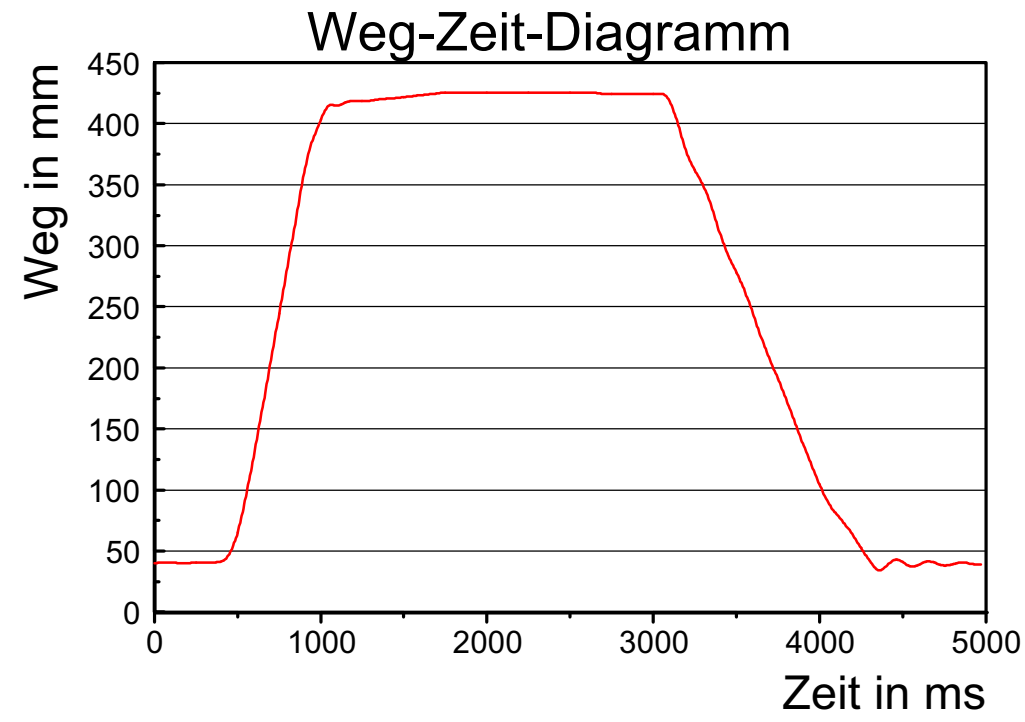


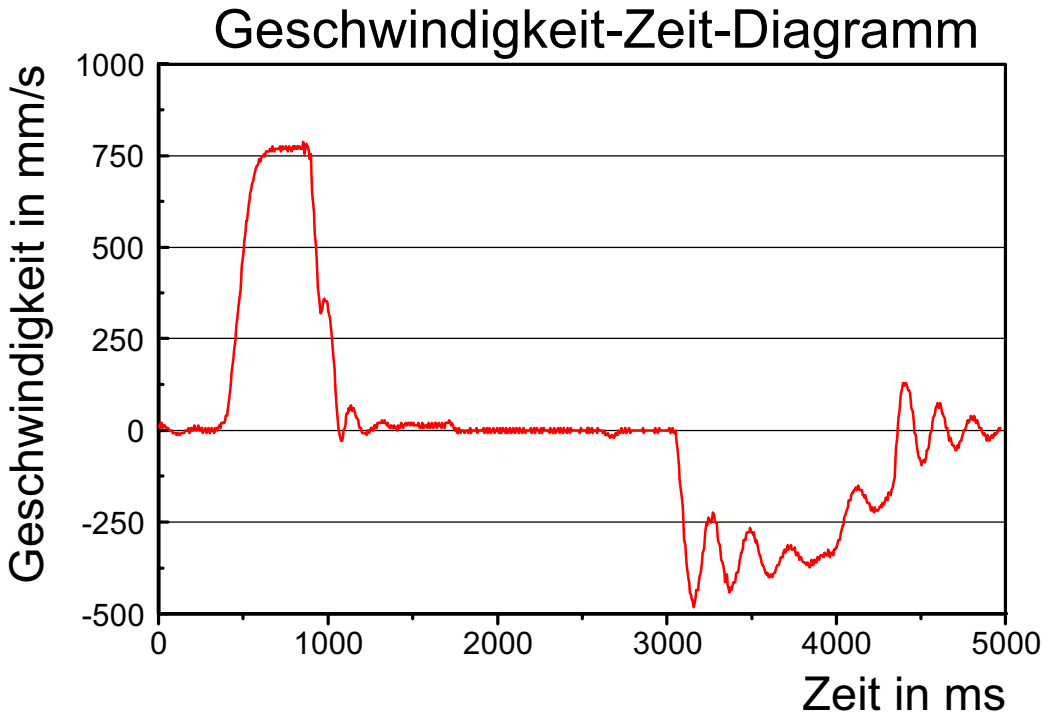


Maximalgeschwindigkeit: 768mm/s

Beschleunigungsweg: 132.23mm

Messung 6 mit Werkzeug

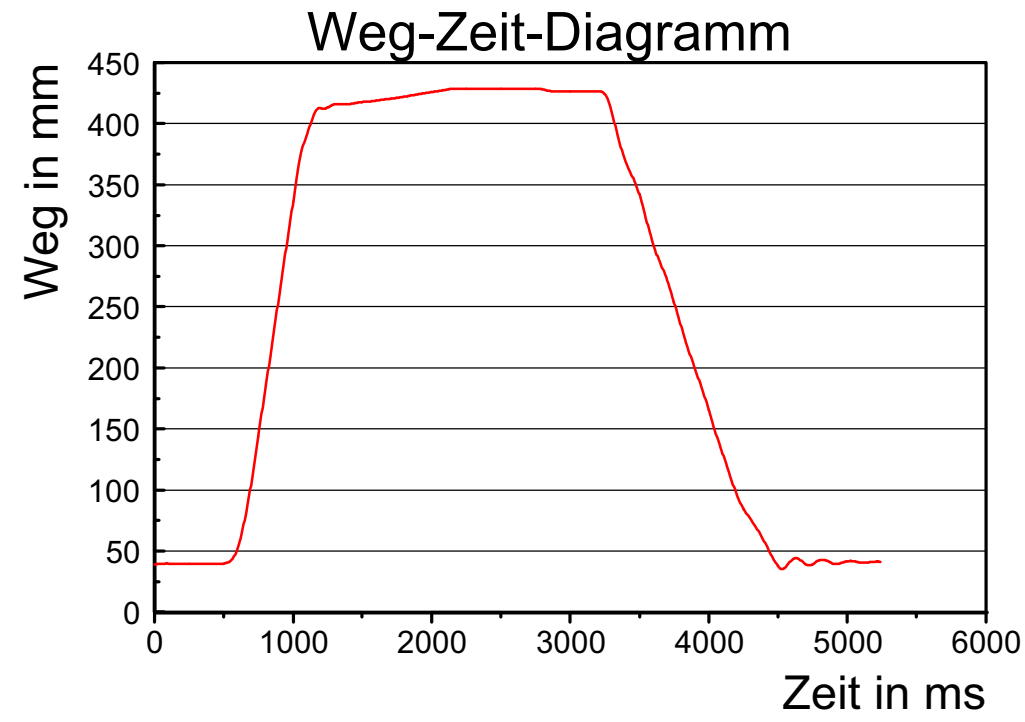


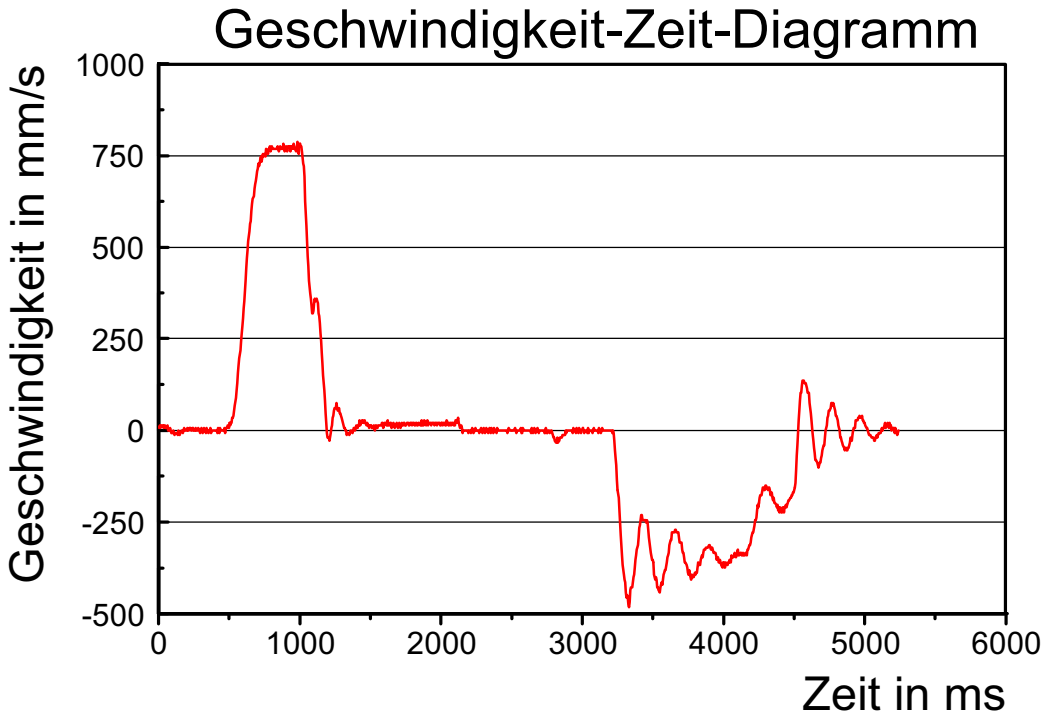


Maximalgeschwindigkeit: 768mm/s

Beschleunigungsweg: 134.94mm

Messung 7 mit Werkzeug

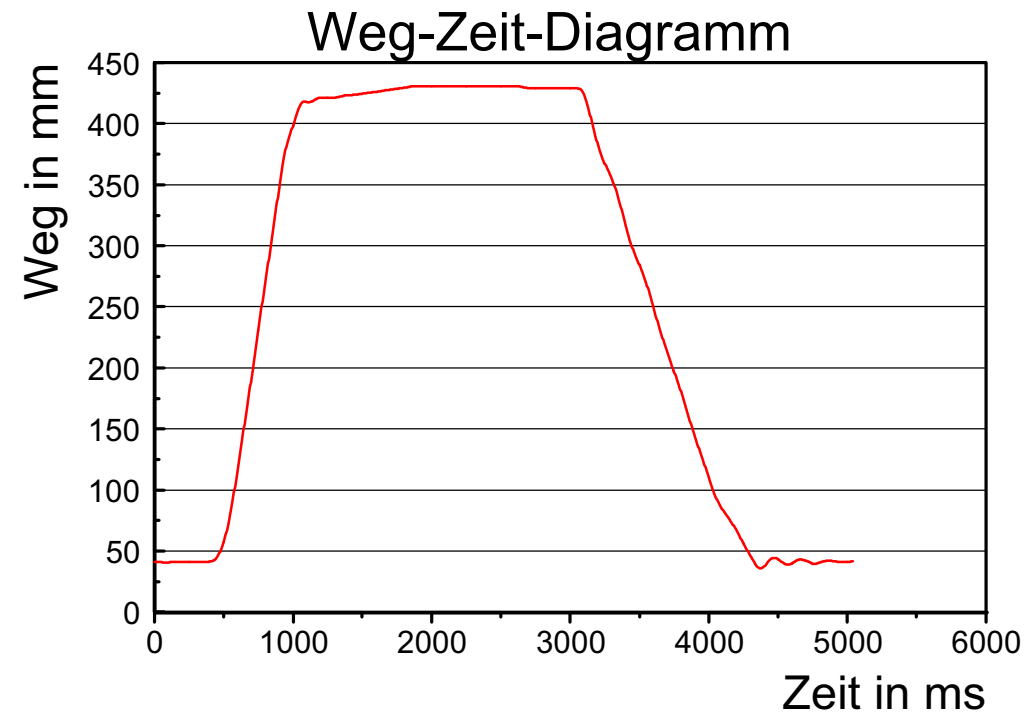


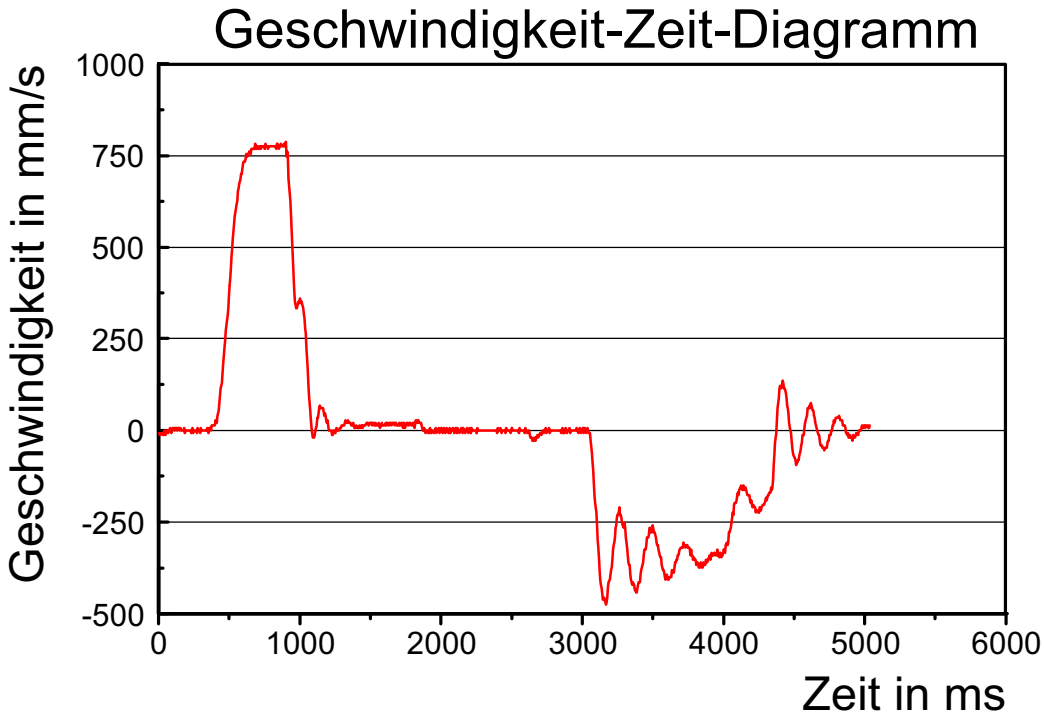


Maximalgeschwindigkeit: 768mm/s

Beschleunigungsweg: 115.74mm

Messung 8 mit Werkzeug

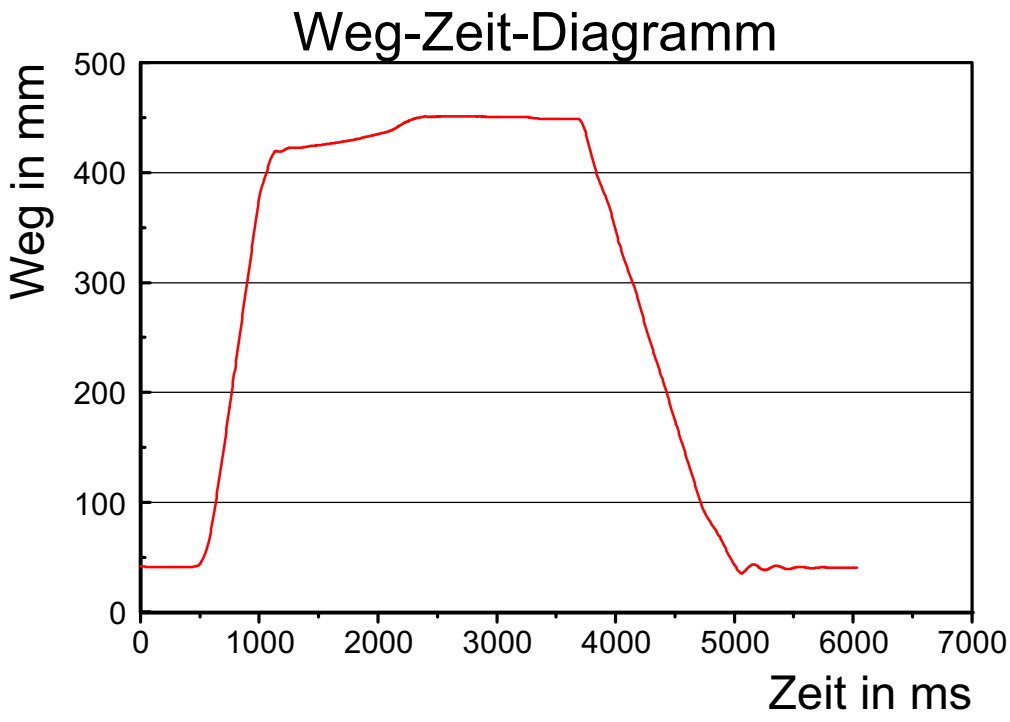


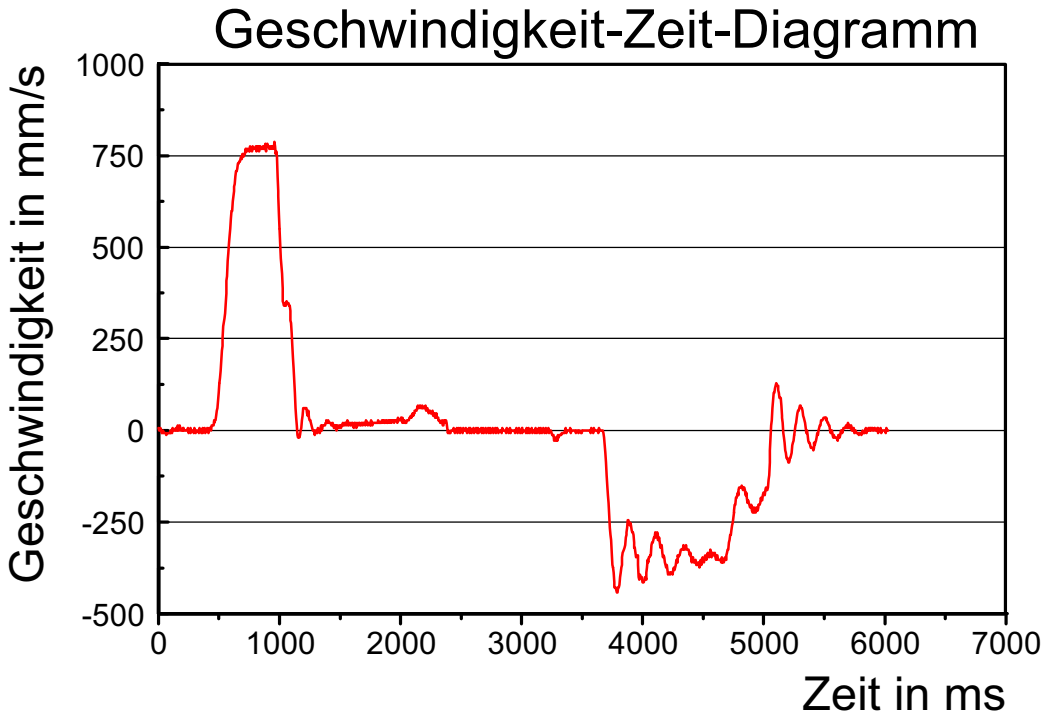


Maximalgeschwindigkeit: 768mm/s

Beschleunigungsweg: 120.56mm

Messung 9 mit Werkzeug

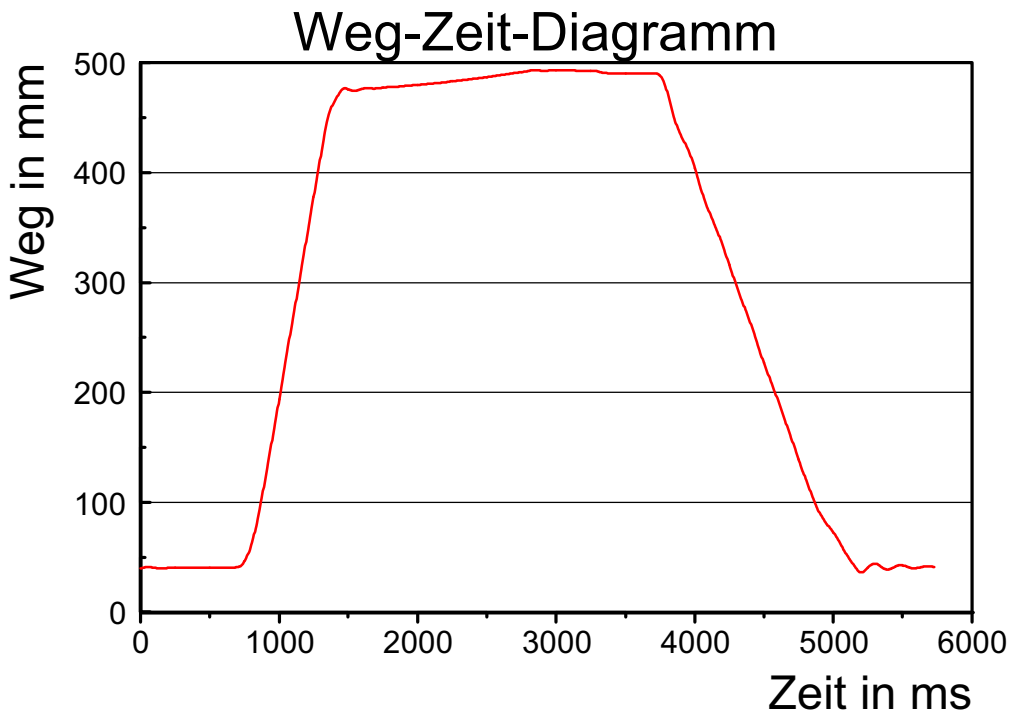


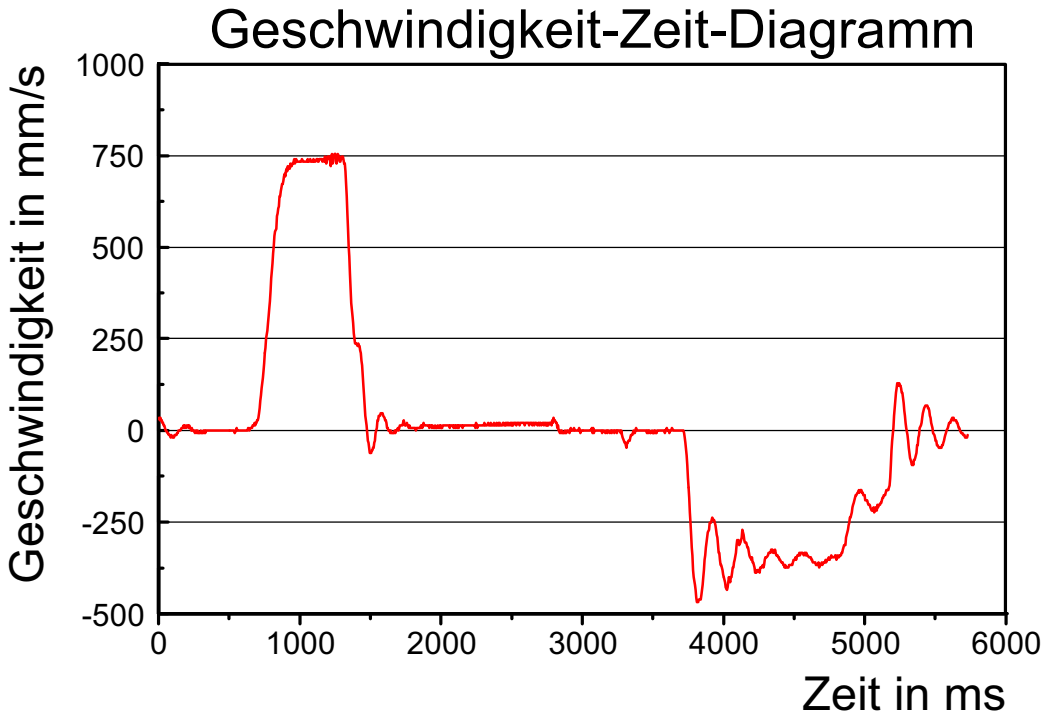


Maximalgeschwindigkeit: 768mm/s

Beschleunigungsweg: 121.93mm

Messung 0 ohne Werkzeug

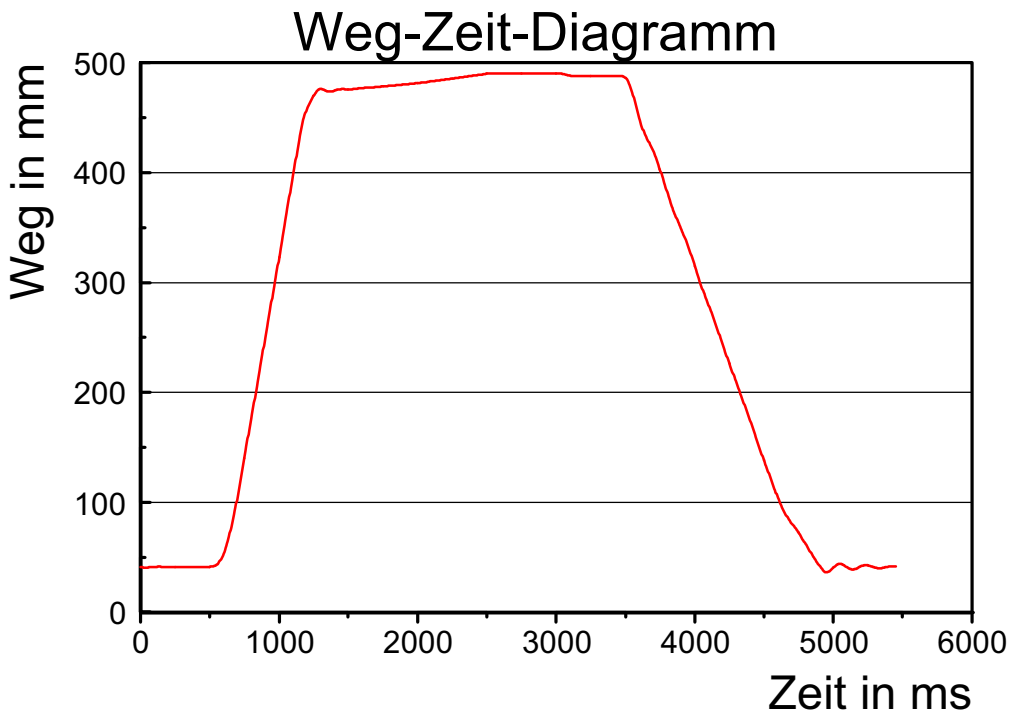


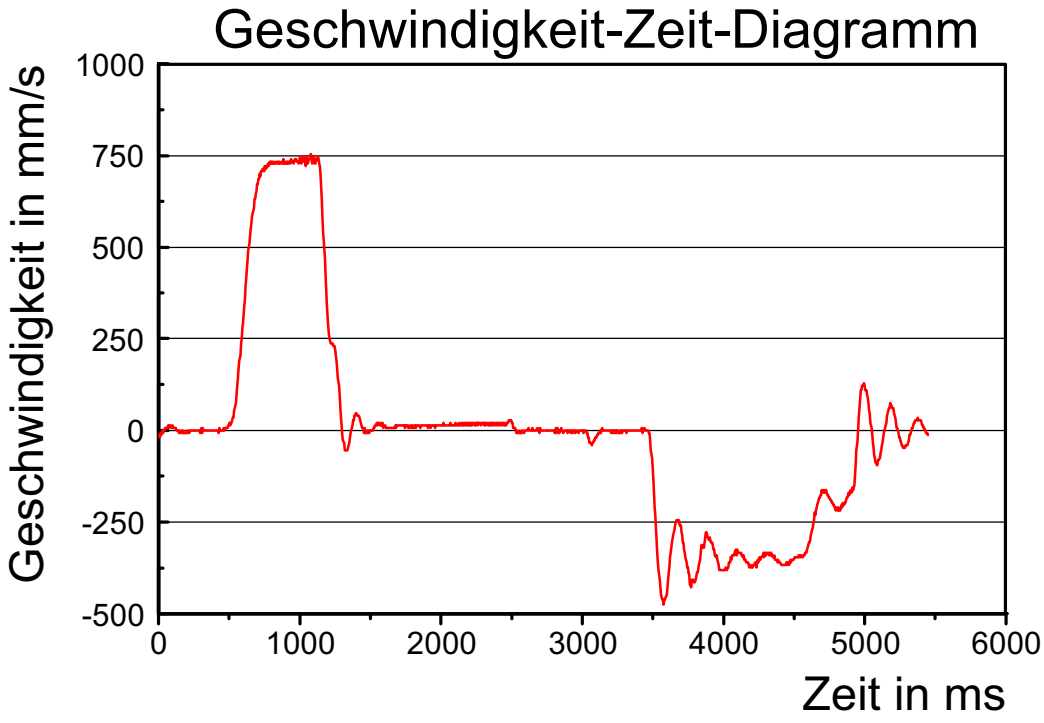


Maximalgeschwindigkeit: 741mm/s

Beschleunigungsweg: 127.4mm

Messung 1 ohne Werkzeug

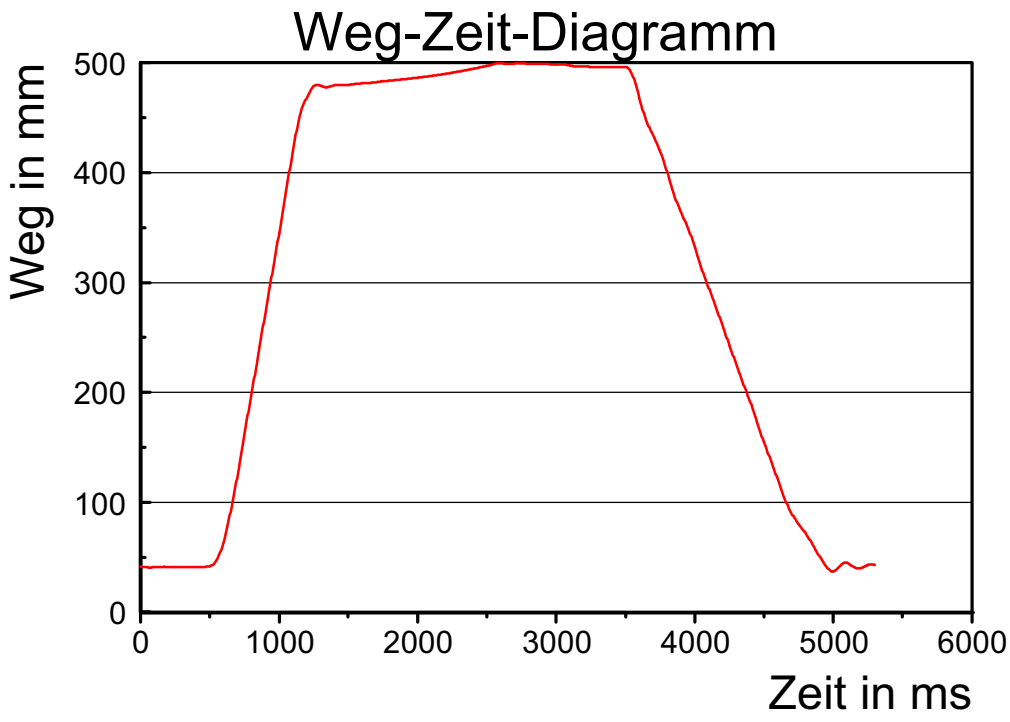


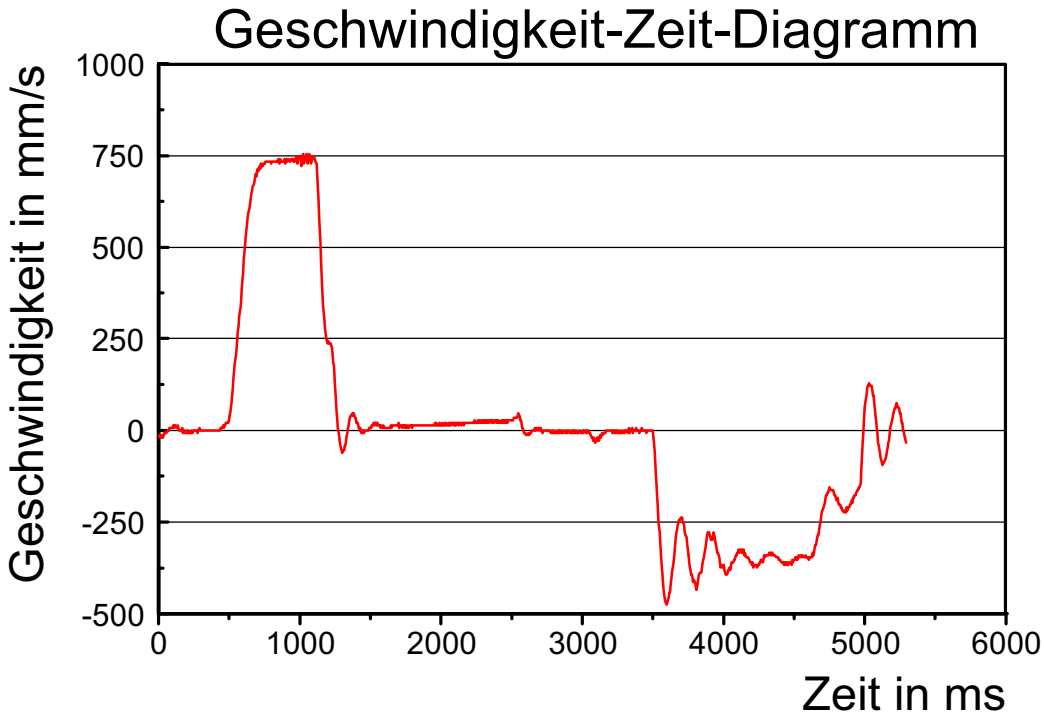


Maximalgeschwindigkeit: 734mm/s

Beschleunigungsweg: 127.94mm

Messung 2 ohne Werkzeug

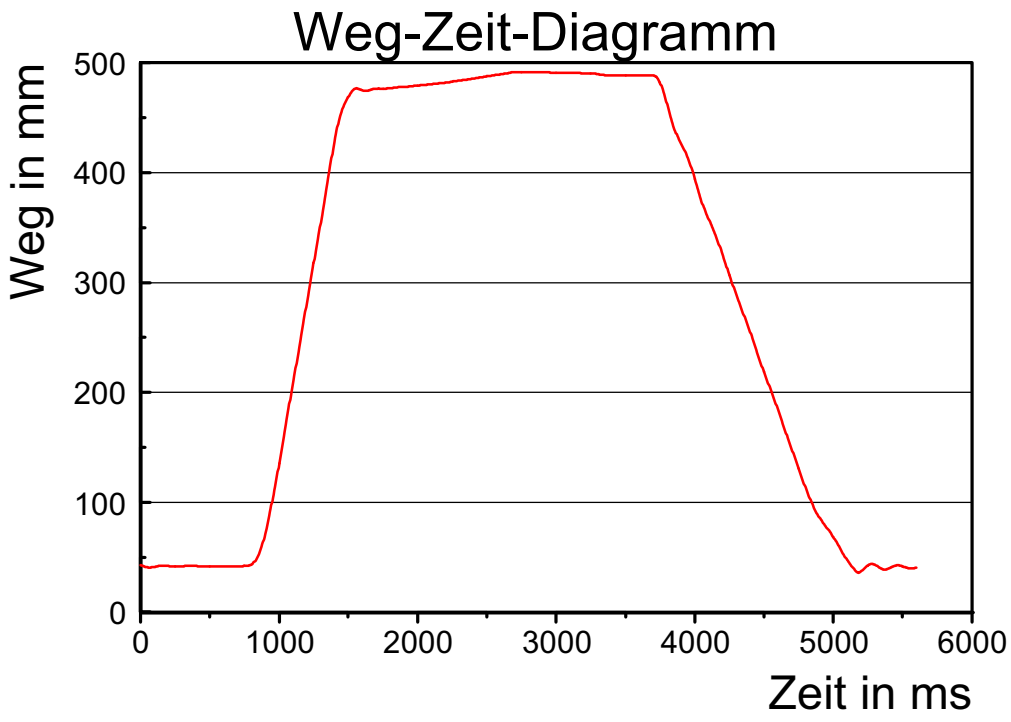


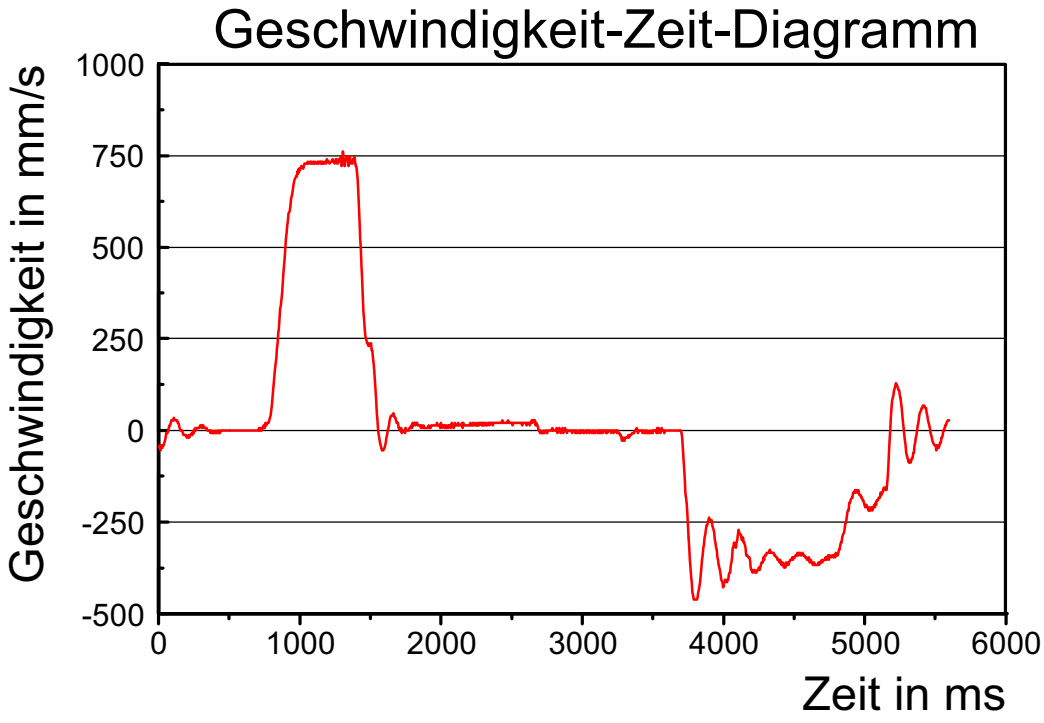


Maximalgeschwindigkeit: 734mm/s

Beschleunigungsweg: 127.16mm

Messung 3 ohne Werkzeug

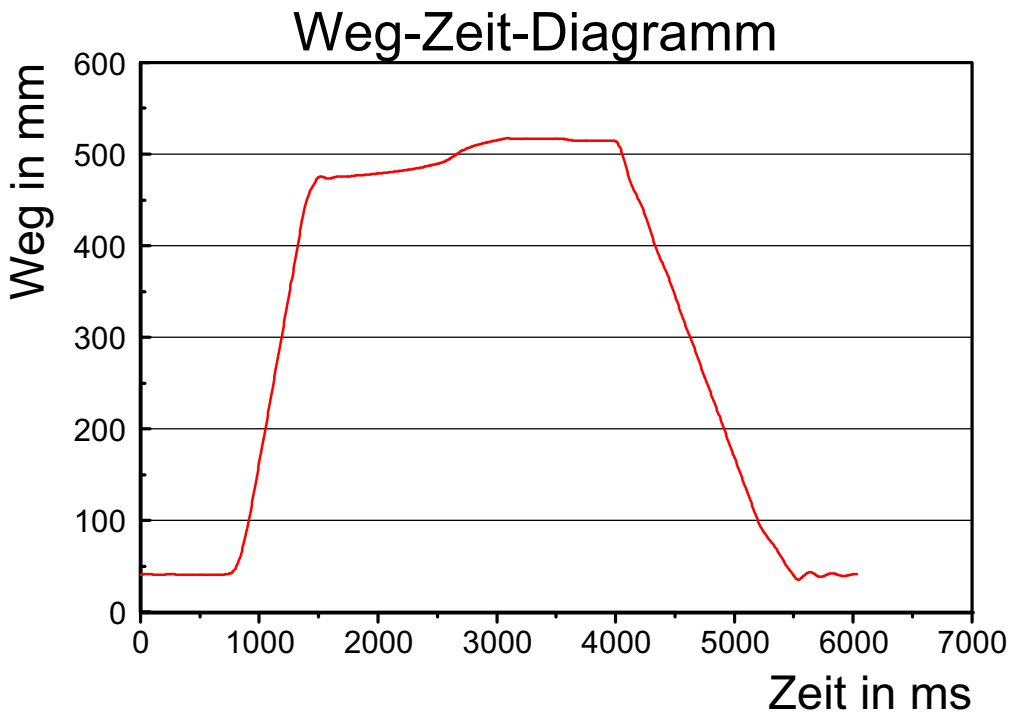


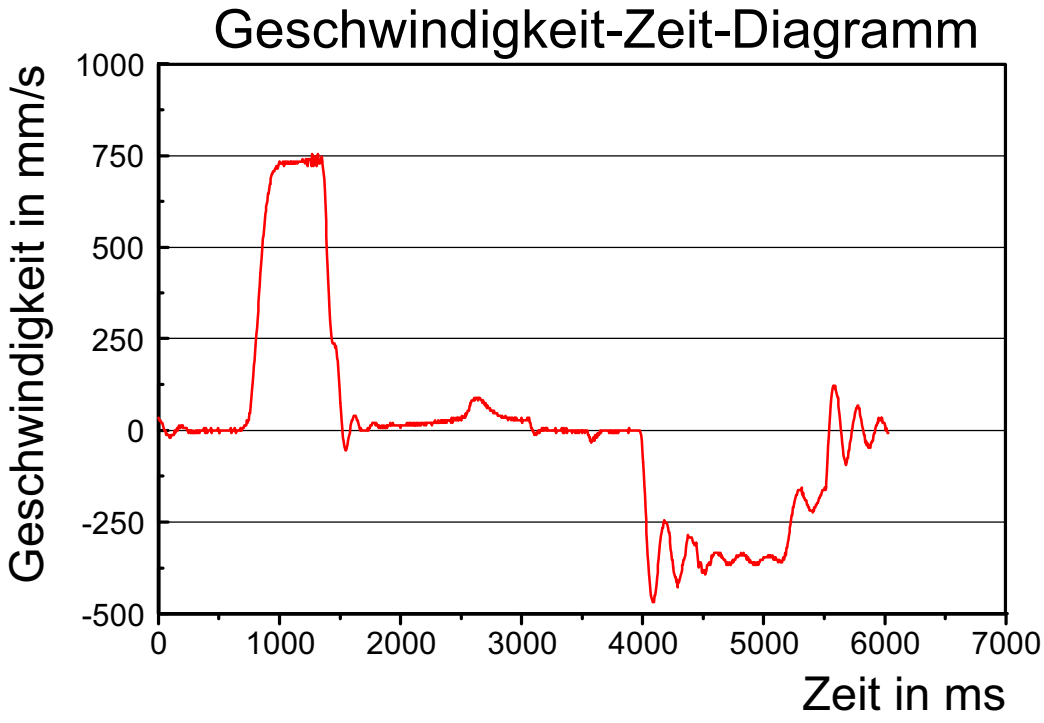


Maximalgeschwindigkeit: 734mm/s

Beschleunigungsweg: 136.62mm

Messung 4 ohne Werkzeug

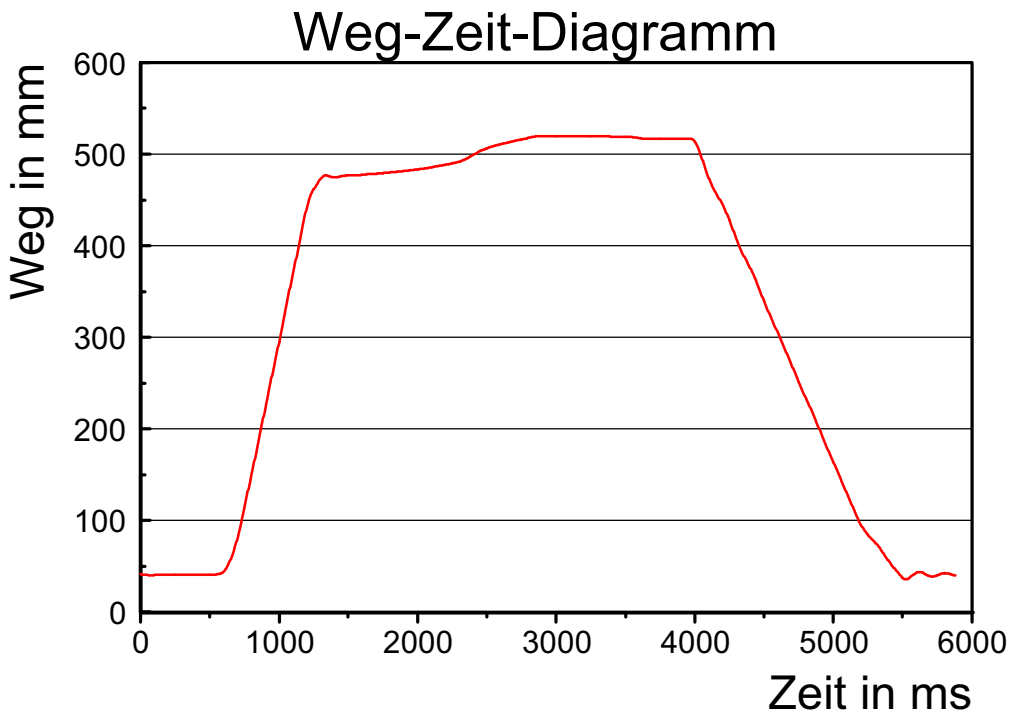


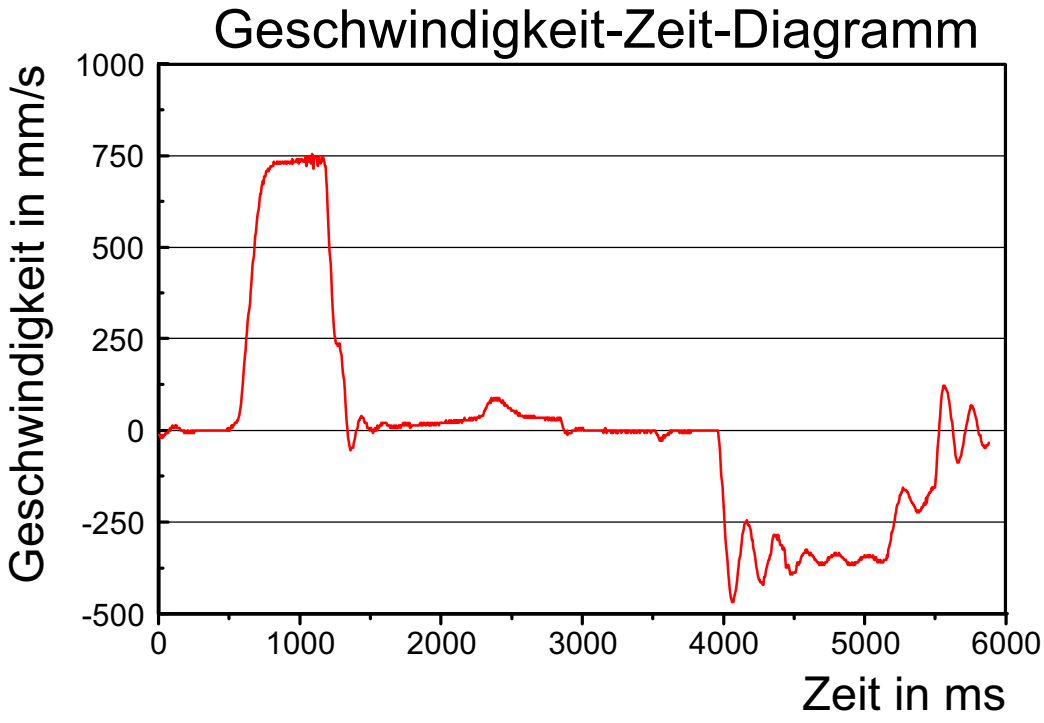


Maximalgeschwindigkeit: 734mm/s

Beschleunigungsweg: 125.46mm

Messung 5 ohne Werkzeug

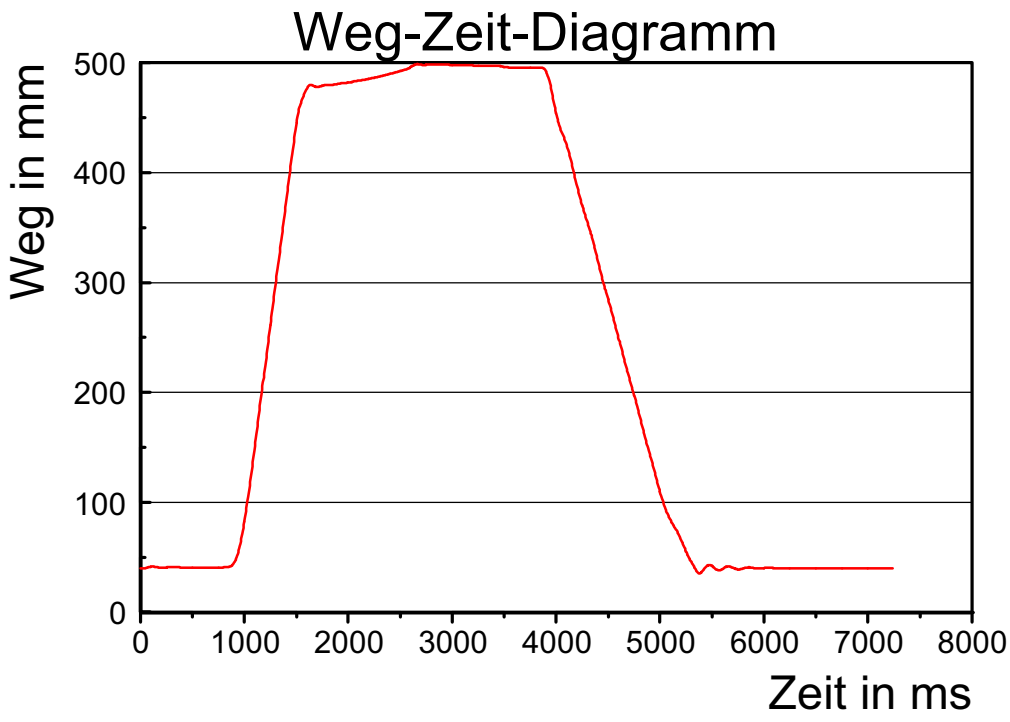


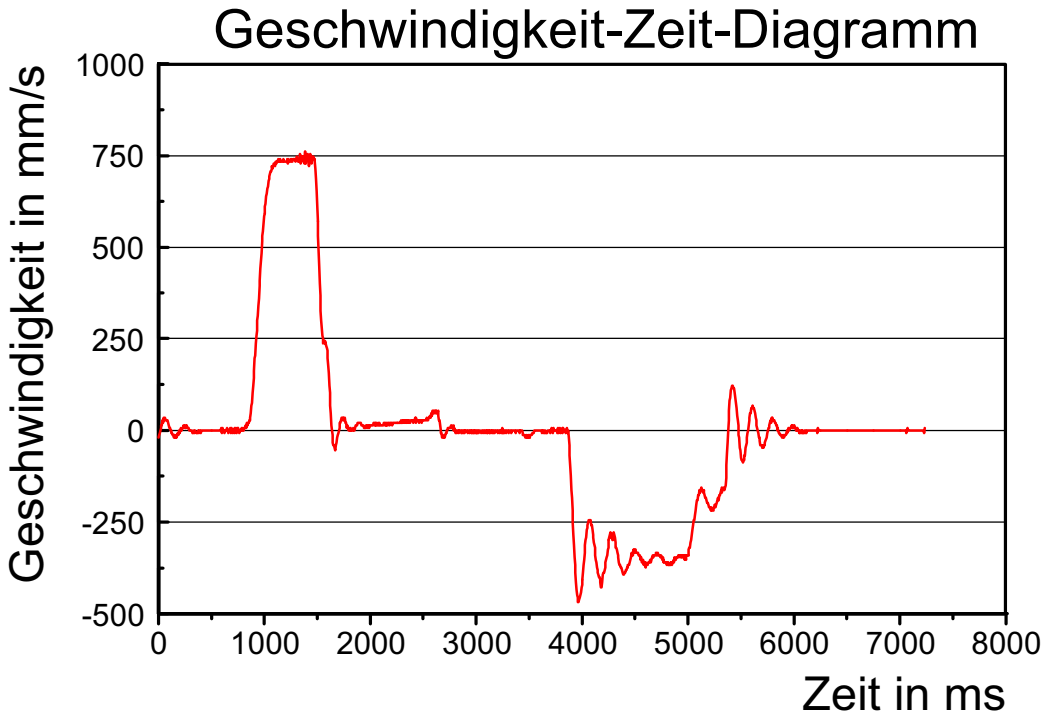


Maximalgeschwindigkeit: 734mm/s

Beschleunigungsweg: 123.05mm

Messung 6 ohne Werkzeug

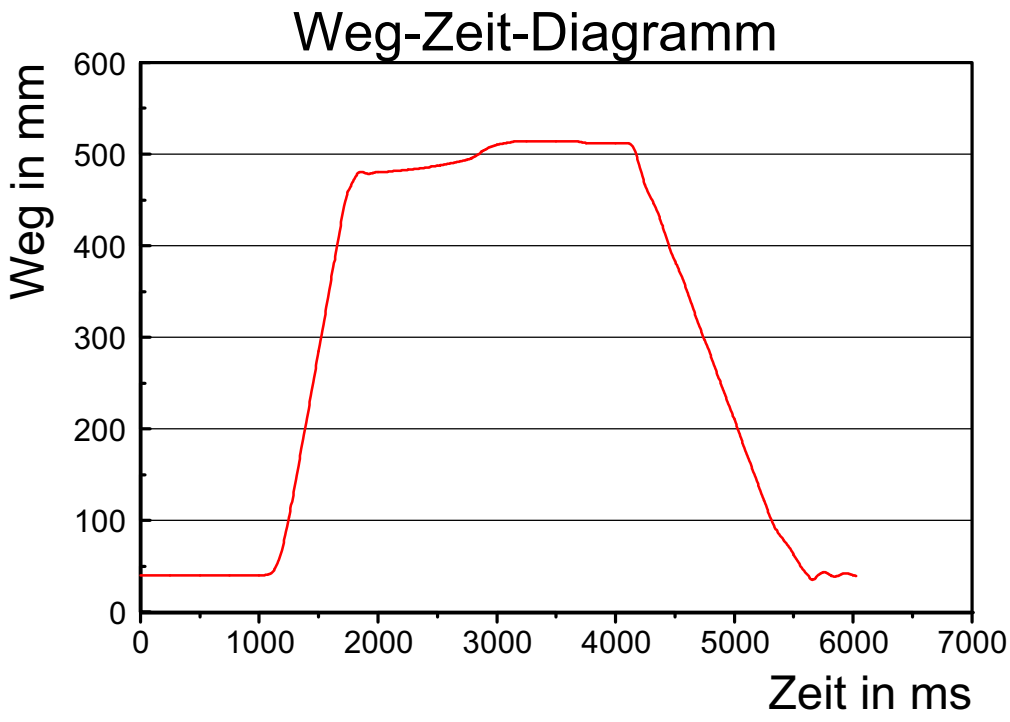


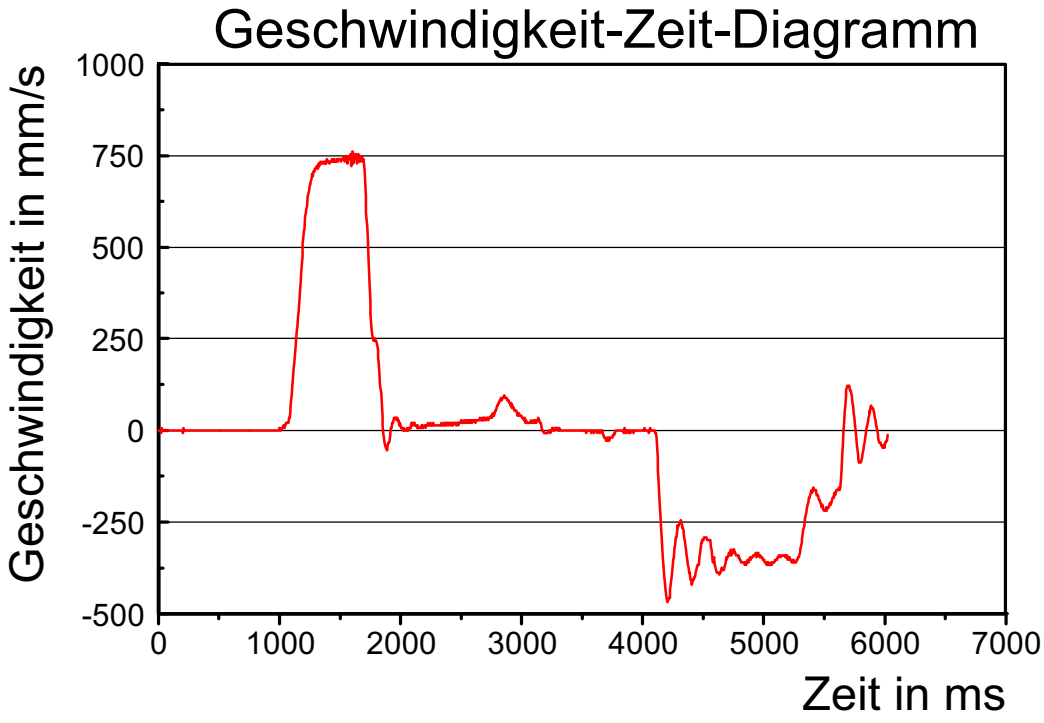


Maximalgeschwindigkeit: 734mm/s

Beschleunigungsweg: 117.40mm

Messung 7 ohne Werkzeug

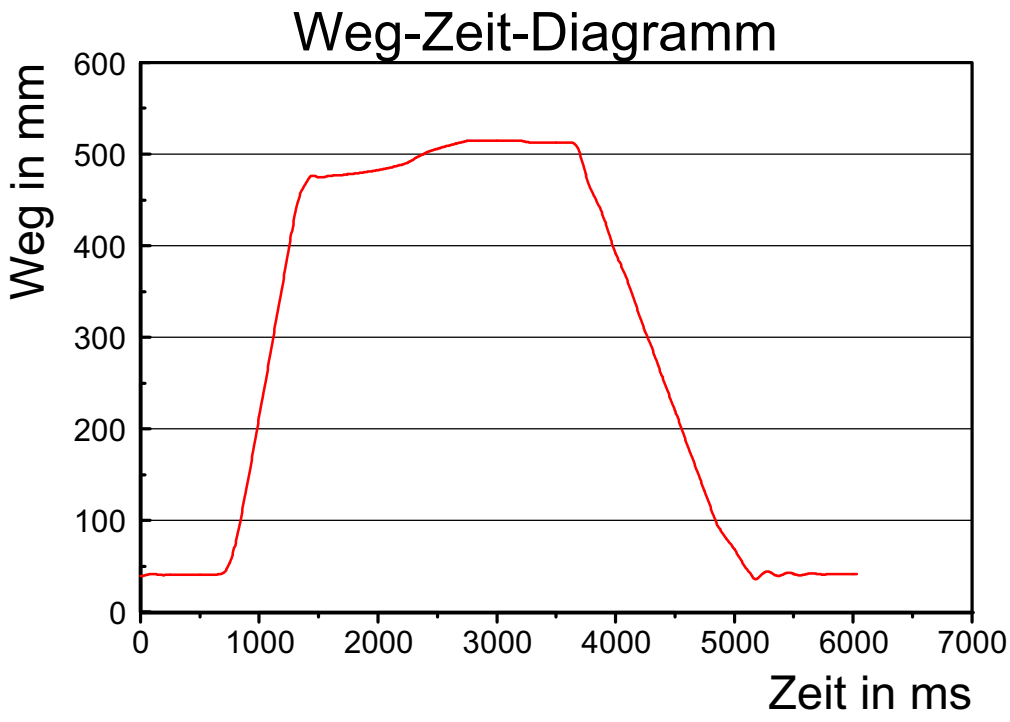


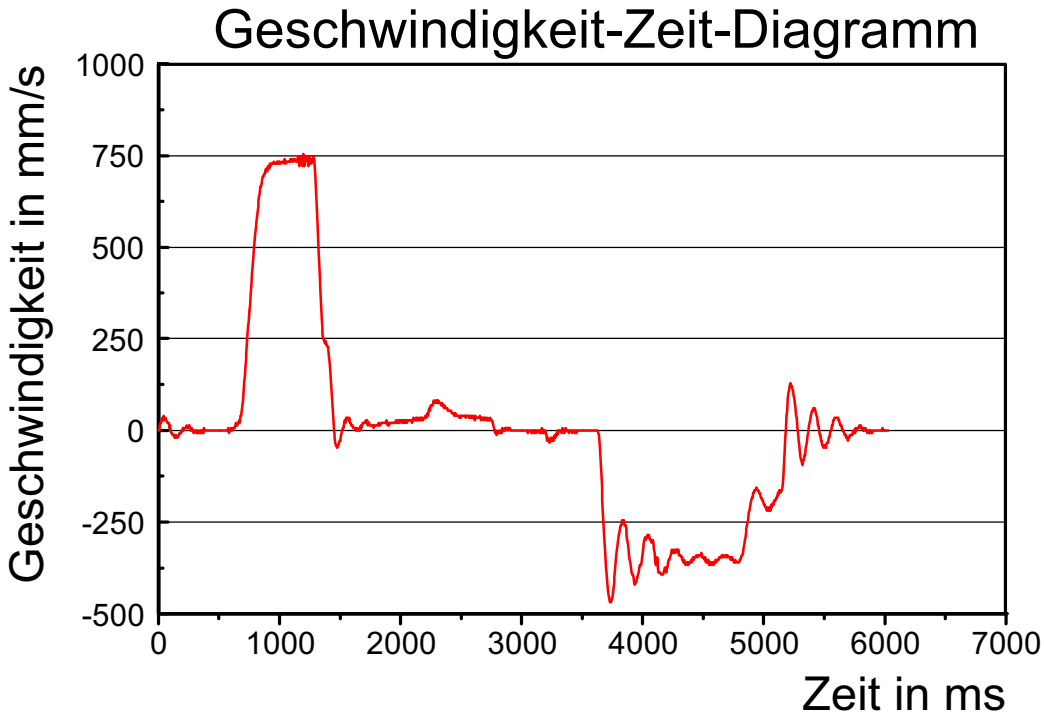


Maximalgeschwindigkeit: 734mm/s

Beschleunigungsweg: 127.12mm

Messung 8 ohne Werkzeug

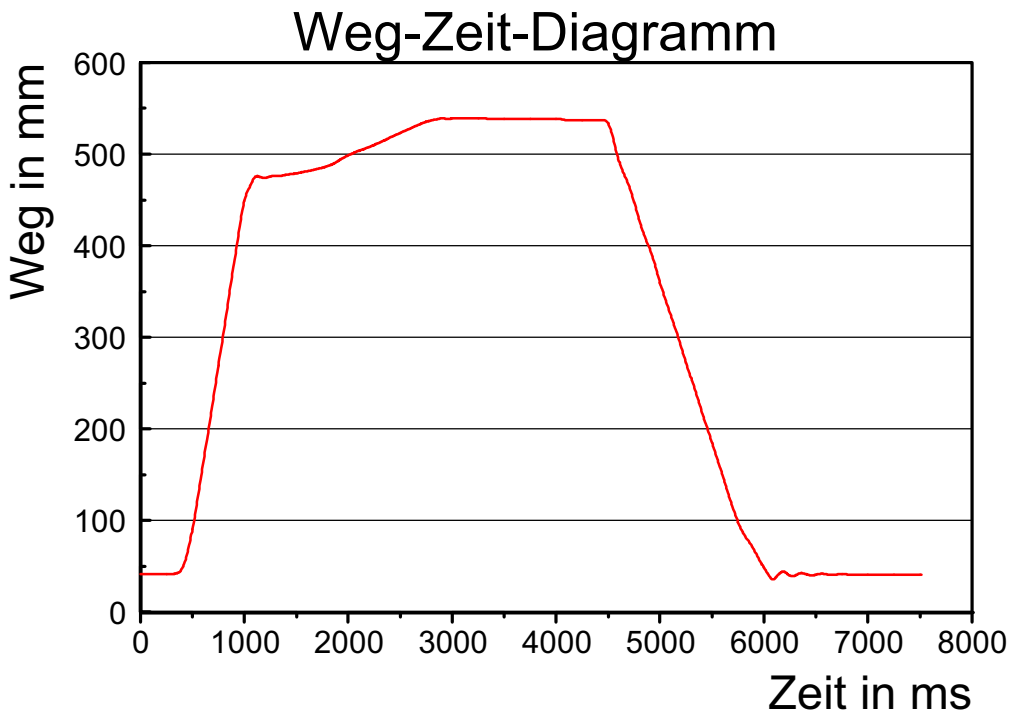


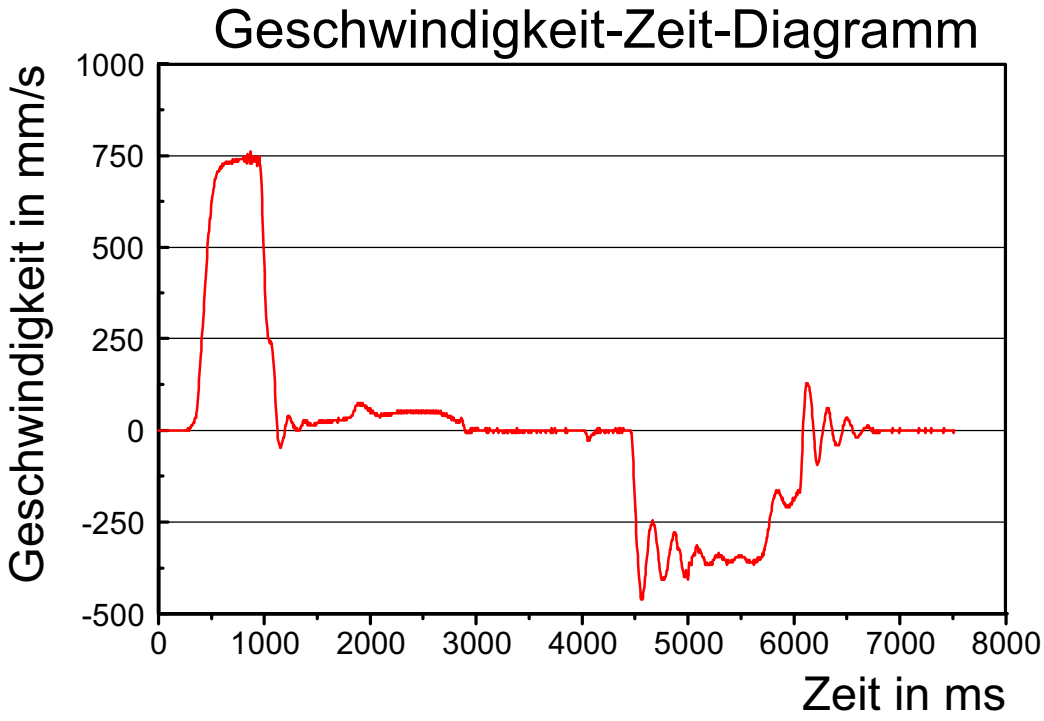


Maximalgeschwindigkeit: 728mm/s

Beschleunigungsweg: 114.48mm

Messung 9 ohne Werkzeug





Maximalgeschwindigkeit: 734mm/s

Beschleunigungsweg: 143.69mm

Vollständiges Protokoll einer Nachlaufwegmessung mit dem bestehenden System und eingebautem Werkzeug

Messergebnisse

Nachlaufweg in mm	Nachlaufzeit in ms
104.50	238
106.90	238
98.55	231
98.70	235
95.06	234
99.03	239
100.90	237
100.50	238
103.20	235
98.91	236

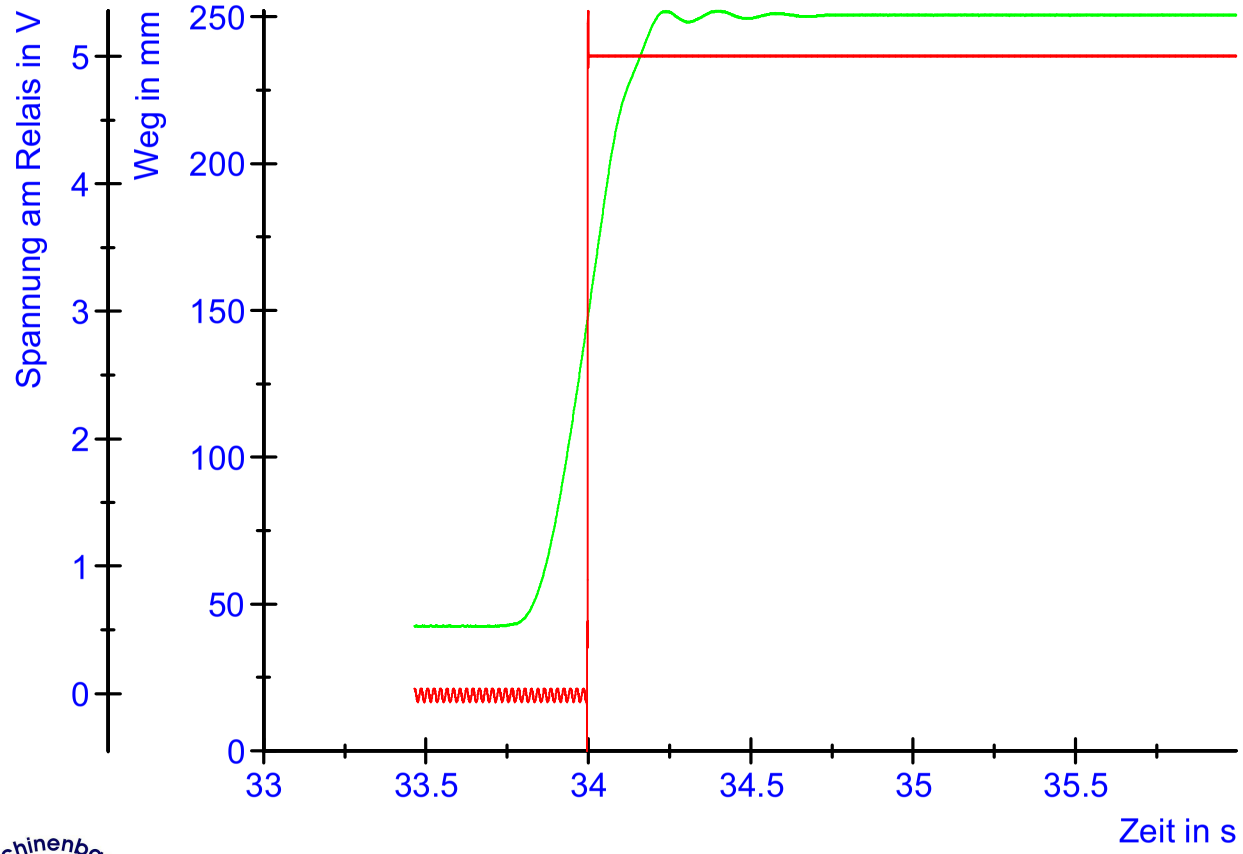
Maschinen-Identifikation: RZU 400
Schutzeinrichtung: Lichtvorhang
Messverfahren: Magnetostriktiv
Messgerät: MAB-A-A-850-N
Verifizierung: 2011
Schaltzeit der Schutzeinrichtung: 41ms
Annäherungsgeschwindigkeit: 1.6m/s

zweithöchste gemessene Nachlaufzeit: 238ms
gemessener Nachlaufweg bei zweithöchster Nachlaufzeit: 106.90mm
resultierender Sicherheitsabstand: 446.4mm

_____

Prüfer: Markus von Hebel
Datum:02/08/2011

Weg-Zeit-Verlauf der Messung 0

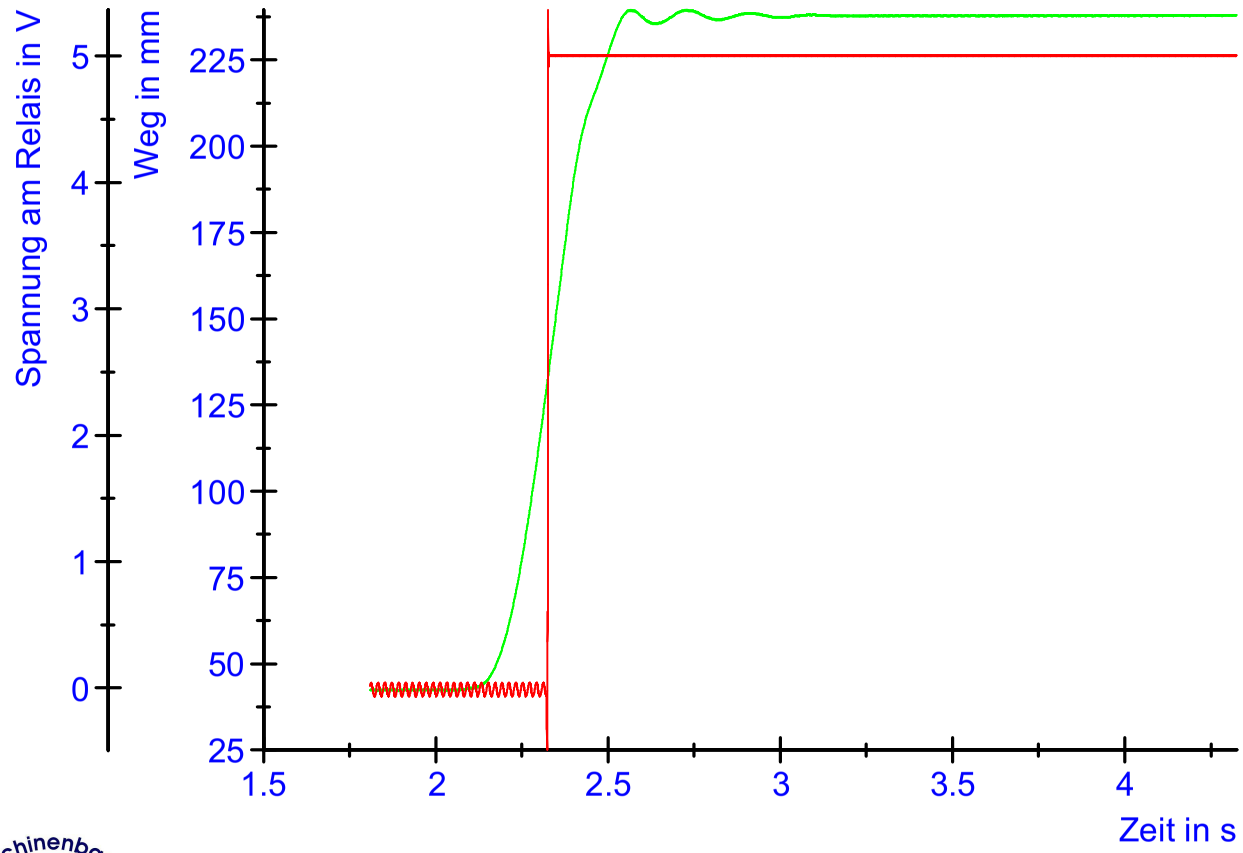


Nachlaufzeit:238ms
Nachlaufweg:104,90mm
max. Geschwindigkeit:743mm/s

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 1

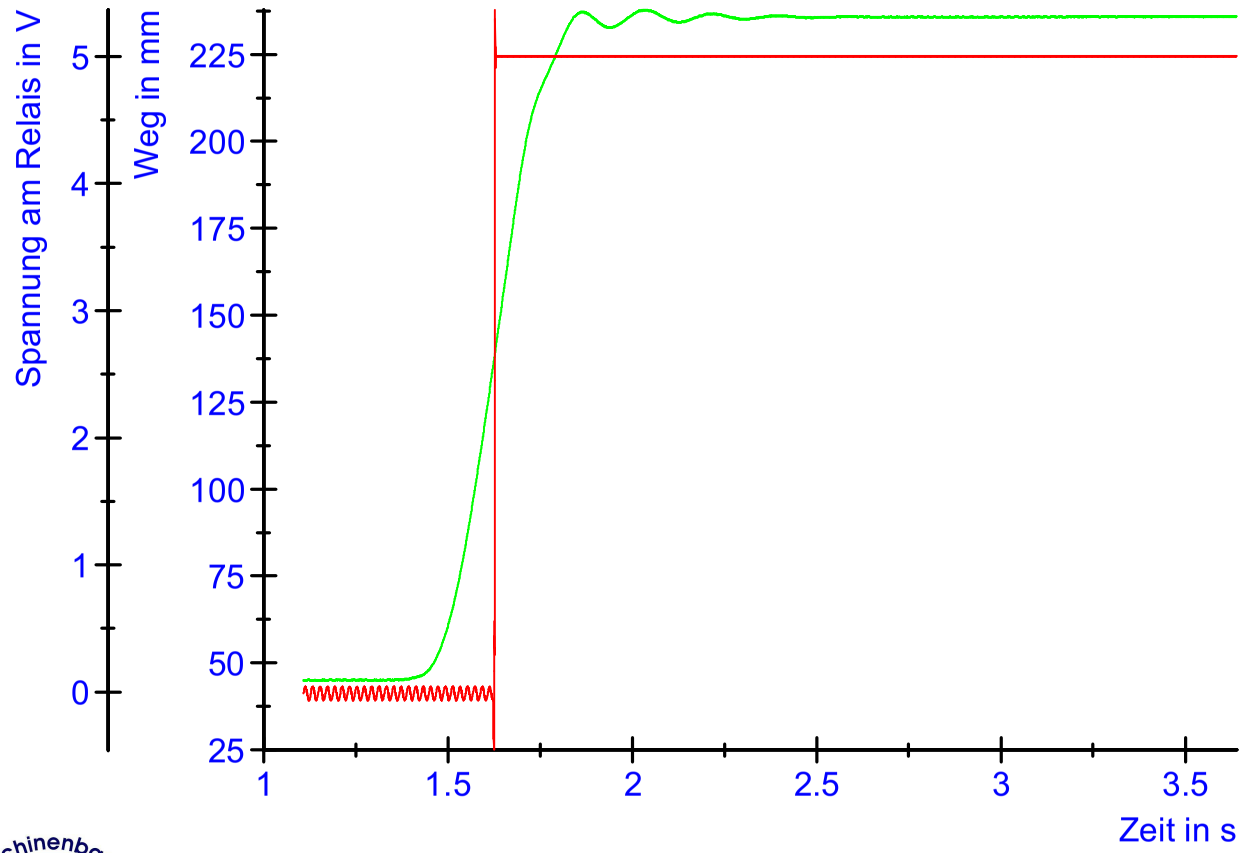


Nachlaufzeit:238ms
Nachlaufweg:106,90mm
max. Geschwindigkeit:749mm/s

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 2

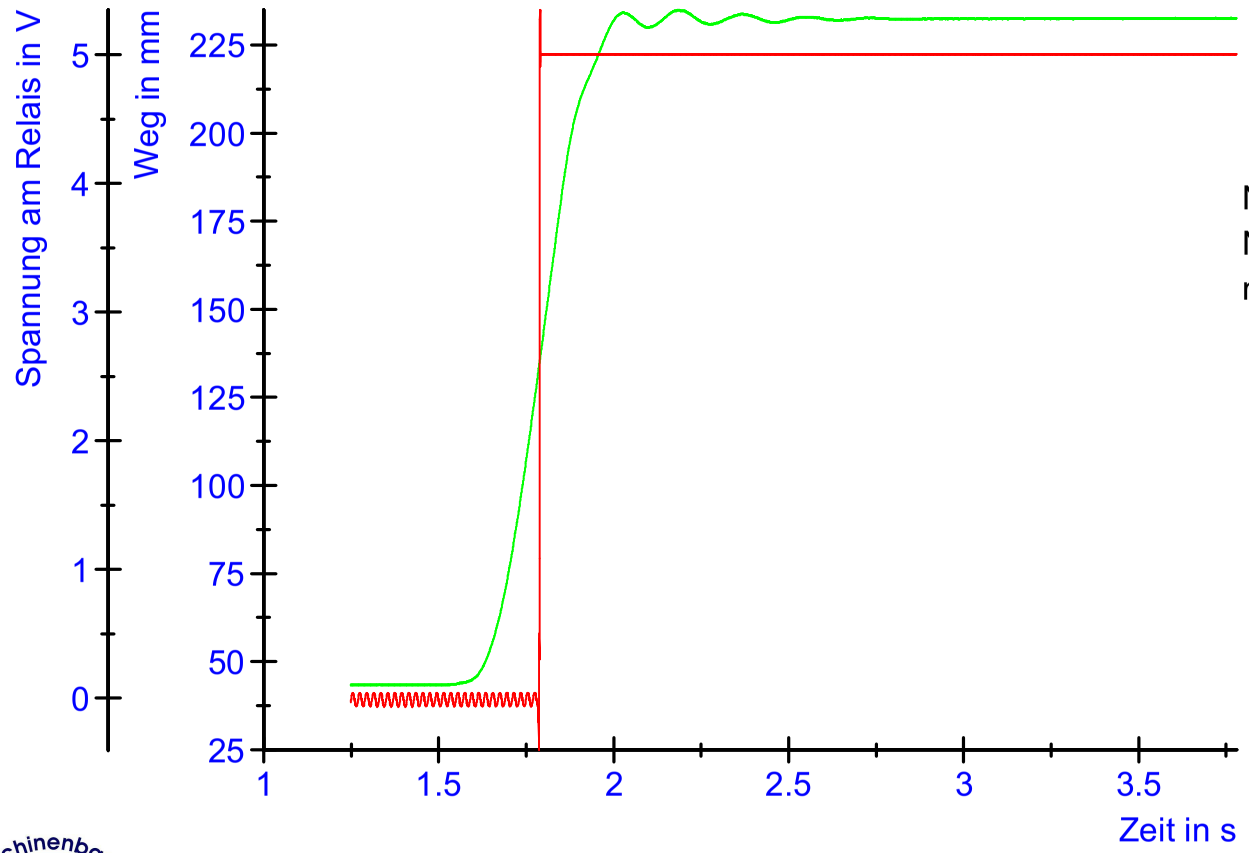


Nachlaufzeit:231ms
Nachlaufweg:98.60mm
max. Geschwindigkeit:745mm/s

Department Maschinenbau und Produktion

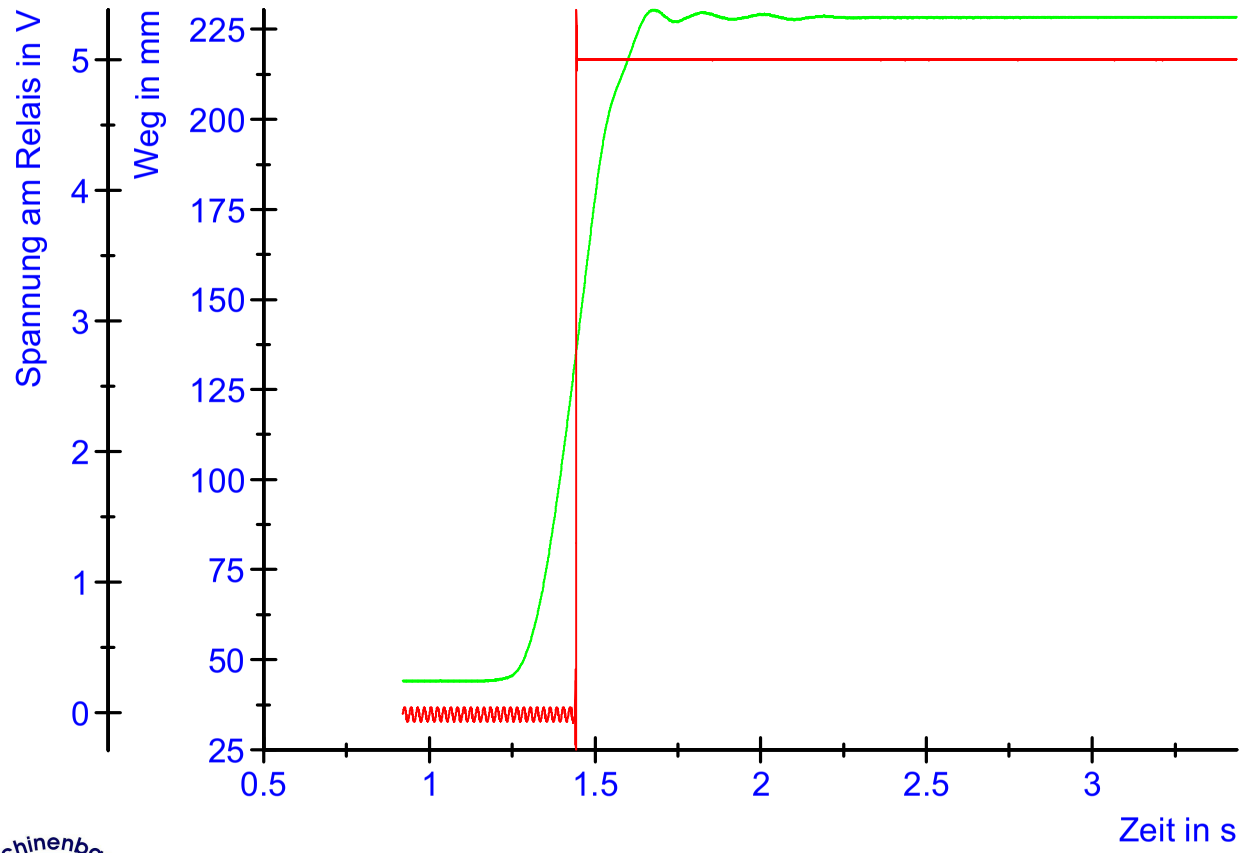
Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 3



Nachlaufzeit:235ms
Nachlaufweg:98.70mm
max. Geschwindigkeit:741mm/s

Weg-Zeit-Verlauf der Messung 4

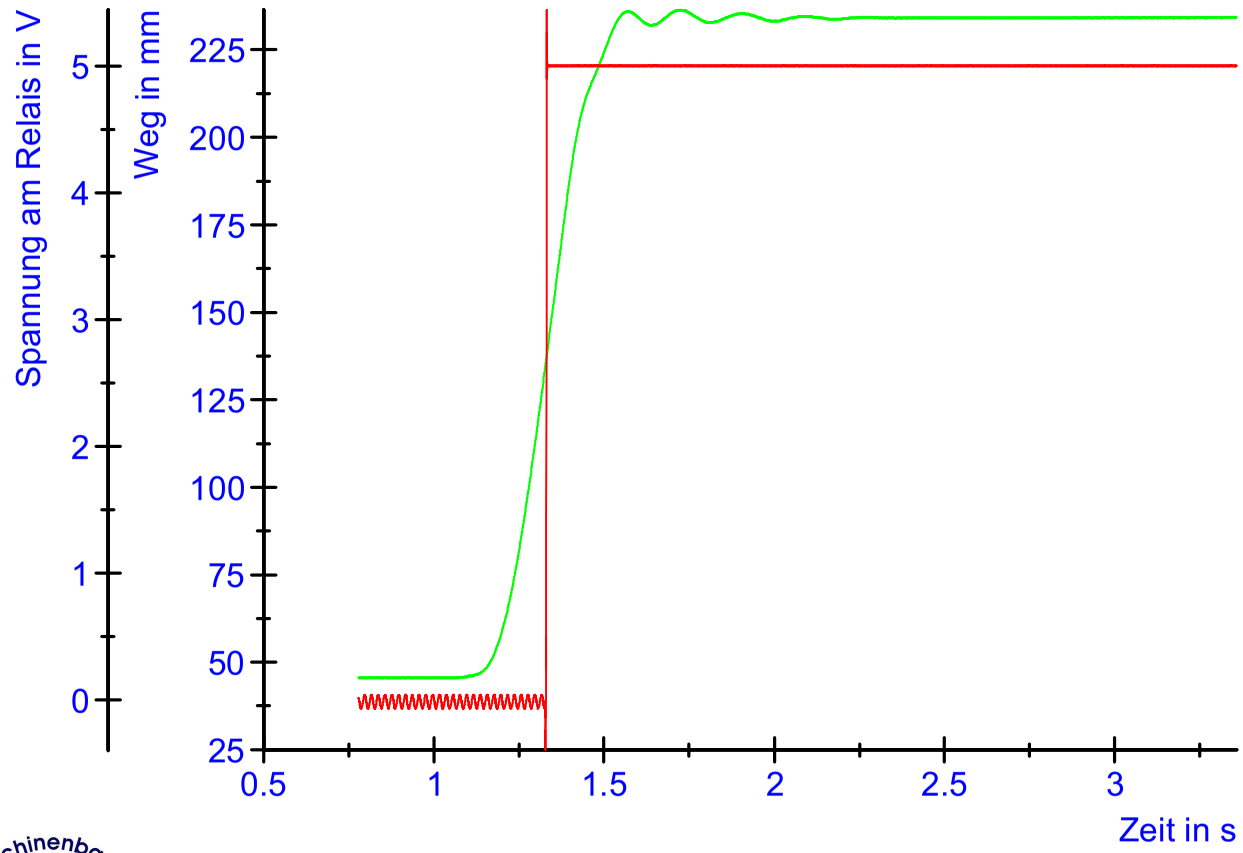


Nachlaufzeit:234ms
Nachlaufweg:95.06mm
max. Geschwindigkeit:747mm/s

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 5

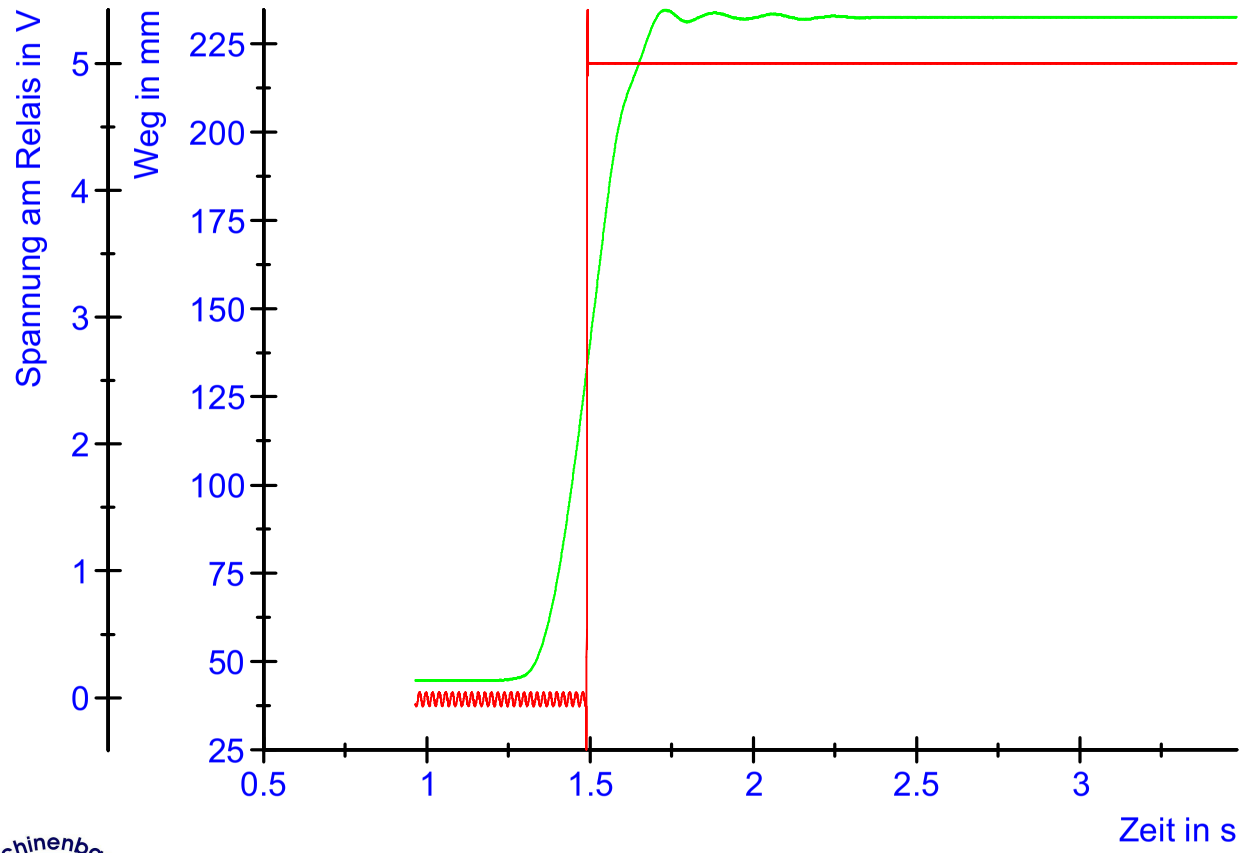


Nachlaufzeit:239ms
Nachlaufweg:99.03mm
max. Geschwindigkeit:726mm/s

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 6

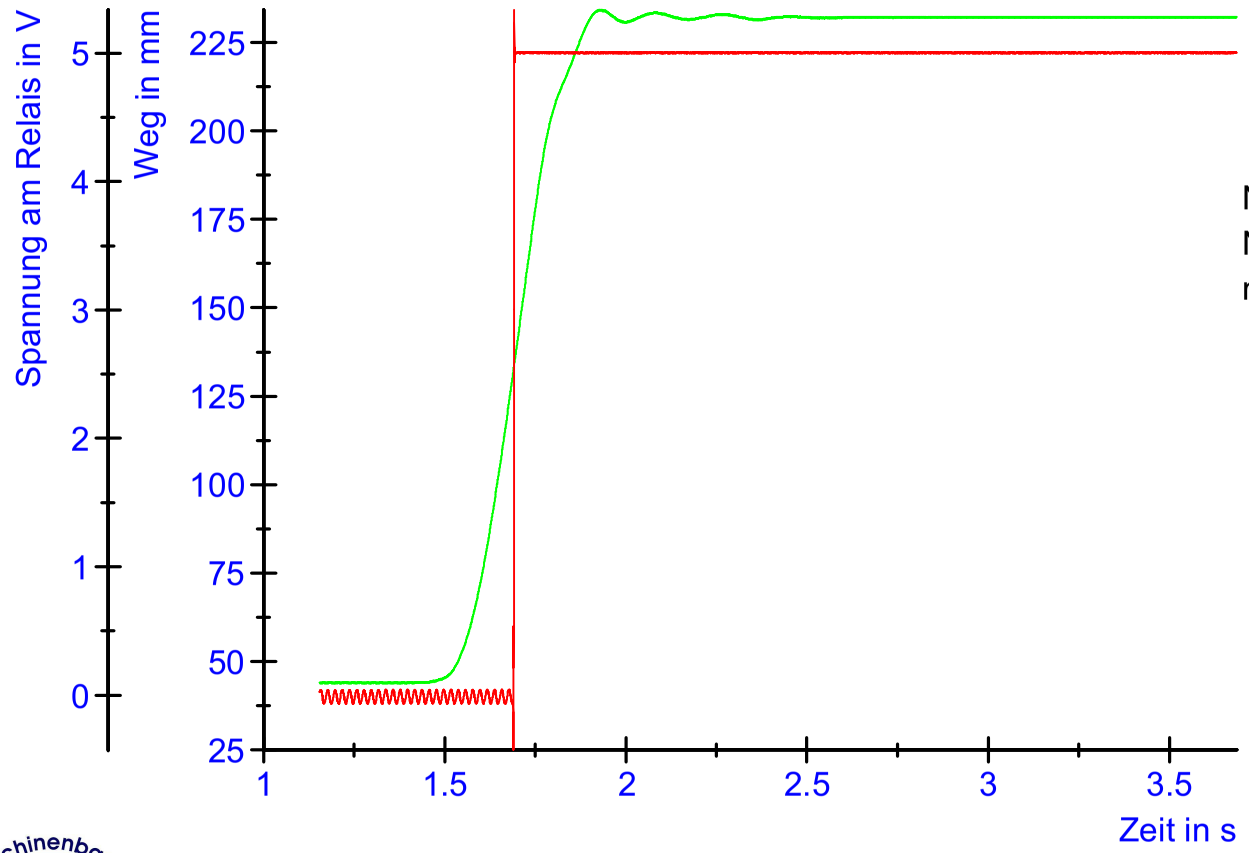


Nachlaufzeit:237ms
Nachlaufweg:100.90mm
max. Geschwindigkeit:734mm/s

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 7

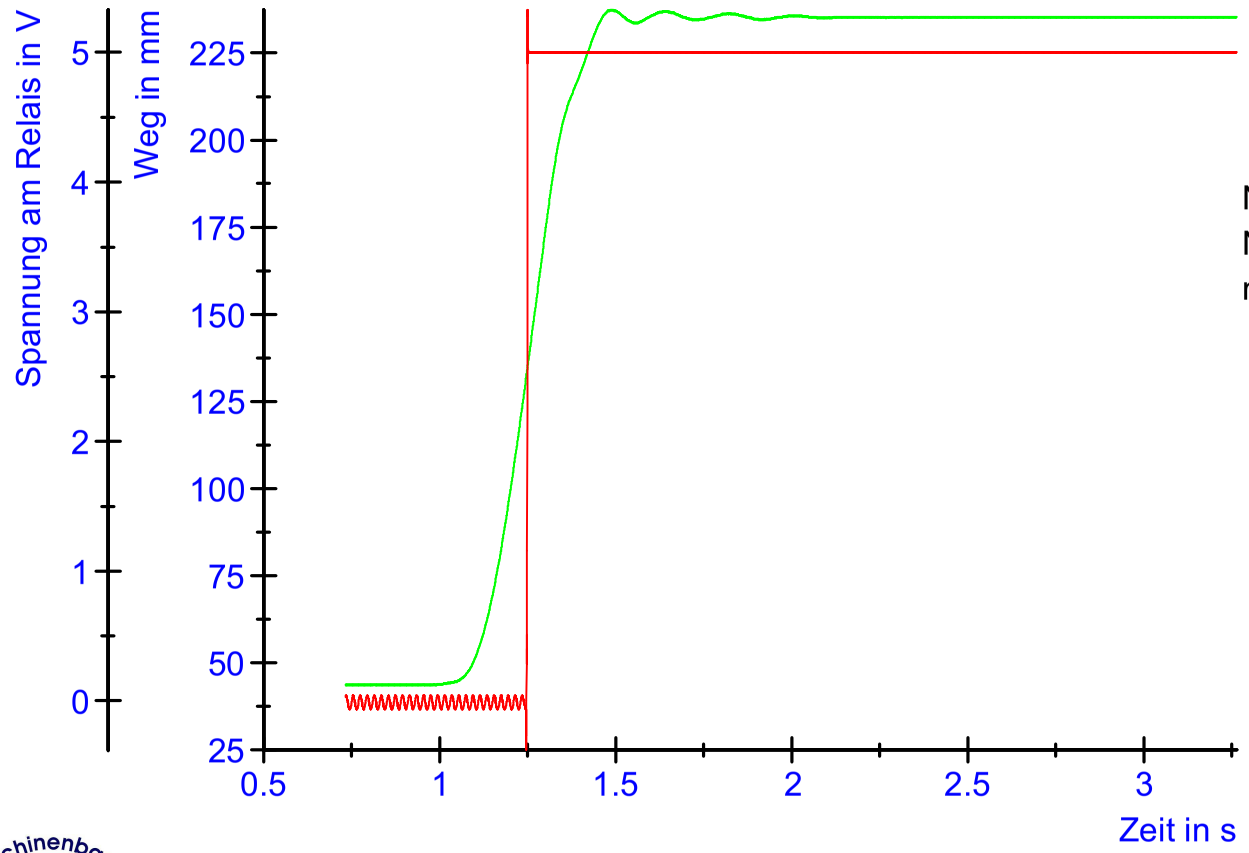


Nachlaufzeit:238ms
Nachlaufweg:100.50mm
max. Geschwindigkeit:746mm/s

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 8

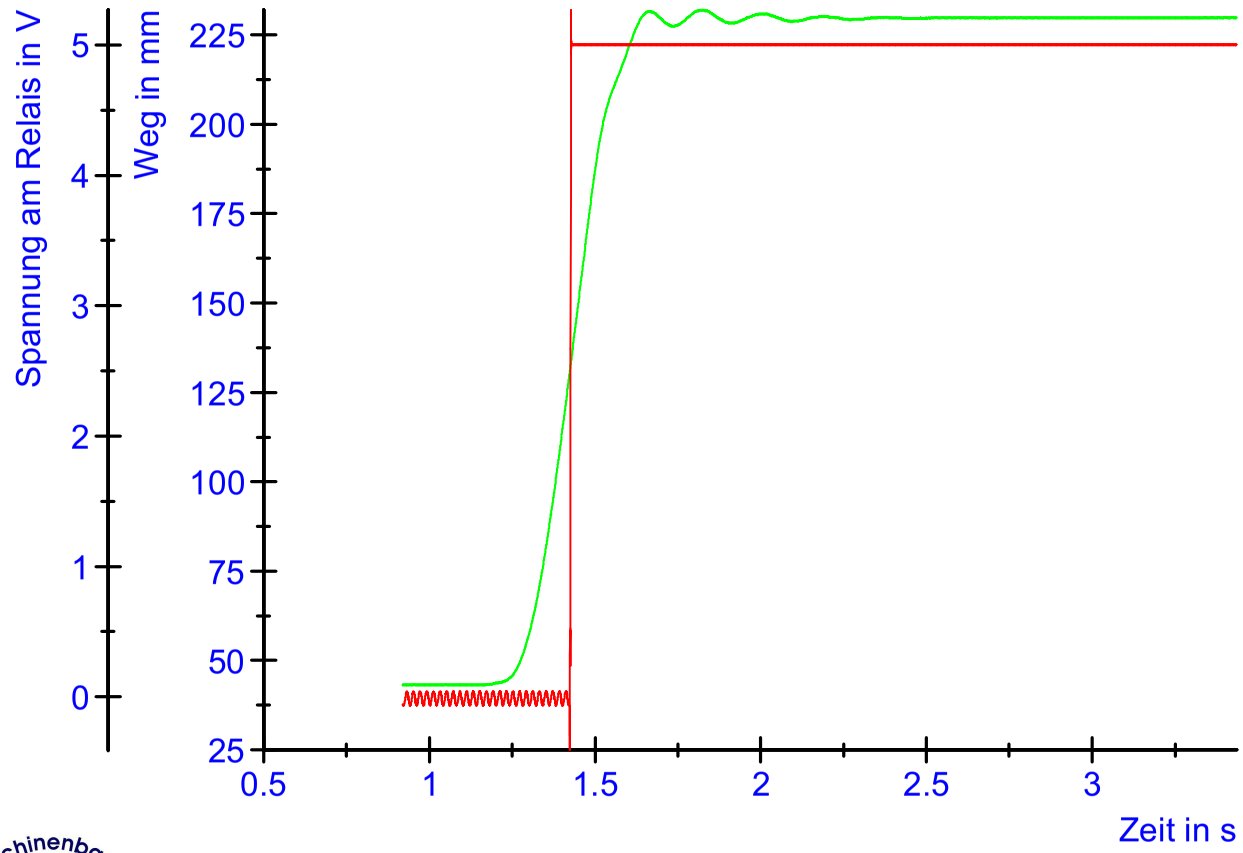


Nachlaufzeit:235ms
Nachlaufweg:103.2mm
max. Geschwindigkeit:728mm/s

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 9



Nachlaufzeit:236ms
Nachlaufweg:98.91mm
max. Geschwindigkeit:728mm/s

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Vollständiges Protokoll einer Nachlaufwegmessung nach DIN 13855 mit den Parametern des alten Systems und eingebautem Werkzeug

Messergebnisse

Nachlaufweg in mm	Nachlaufzeit in ms
102.00	240
98.53	232
104.11	243
102.51	239
107.17	240
104.11	236
101.76	236
98.74	231
102.71	239
101.46	235

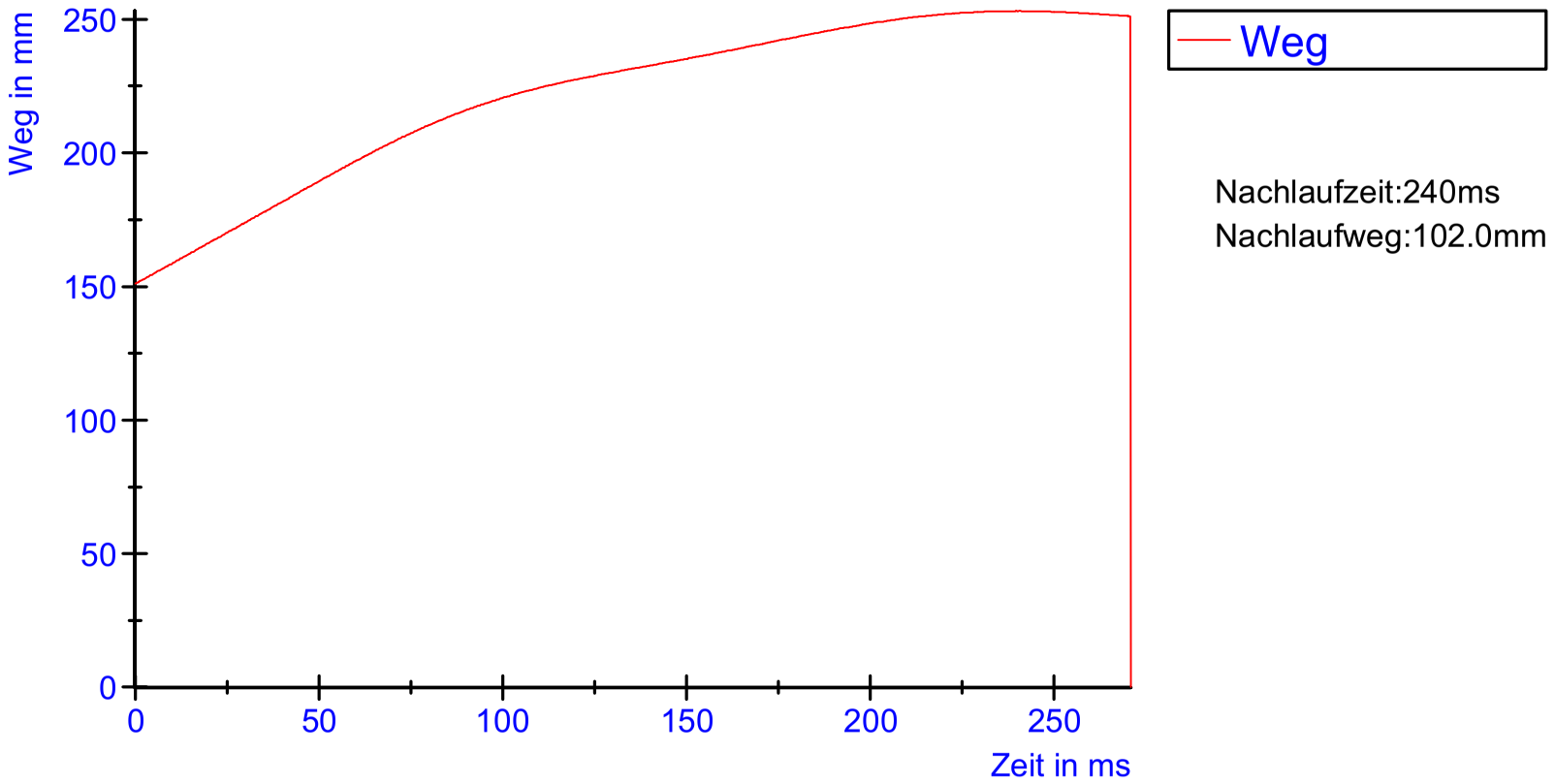
Maschinen-Identifikation: RZU 400
Schutzeinrichtung: Schutzvorhang
Messverfahren: Magnetisch striktiv
Messgerät: WayCon MAB-A-A-850-N
Verifizierung: 2011
Schaltzeit der Schutzeinrichtung: 41ms
Annäherungsgeschwindigkeit: 1.6m/s bei S>=500mm
2m/s bei S<500mm

gemessene maximale Nachlaufzeit: 242.92ms
gemessener Nachlaufweg bei maximaler Nachlaufzeit: 104.11mm
Mittelwert + 3Standardabweichungen: 248.36ms
resultierender Sicherheitsabstand: 500mm

_____

Prüfer: Markus von Hebel
Datum:02/08/2011

Weg-Zeit-Verlauf der Messung 0

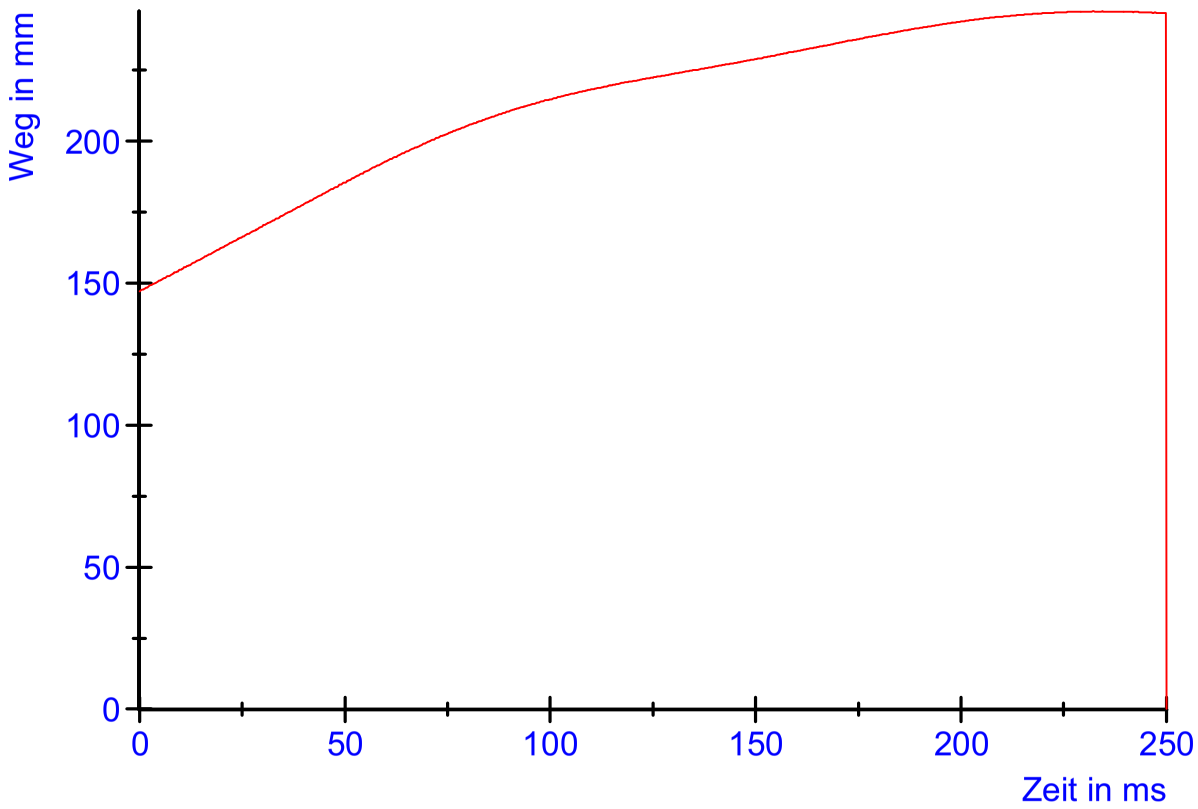


Nachlaufzeit:240ms
Nachlaufweg:102.0mm

Department Maschinenbau und Produktion

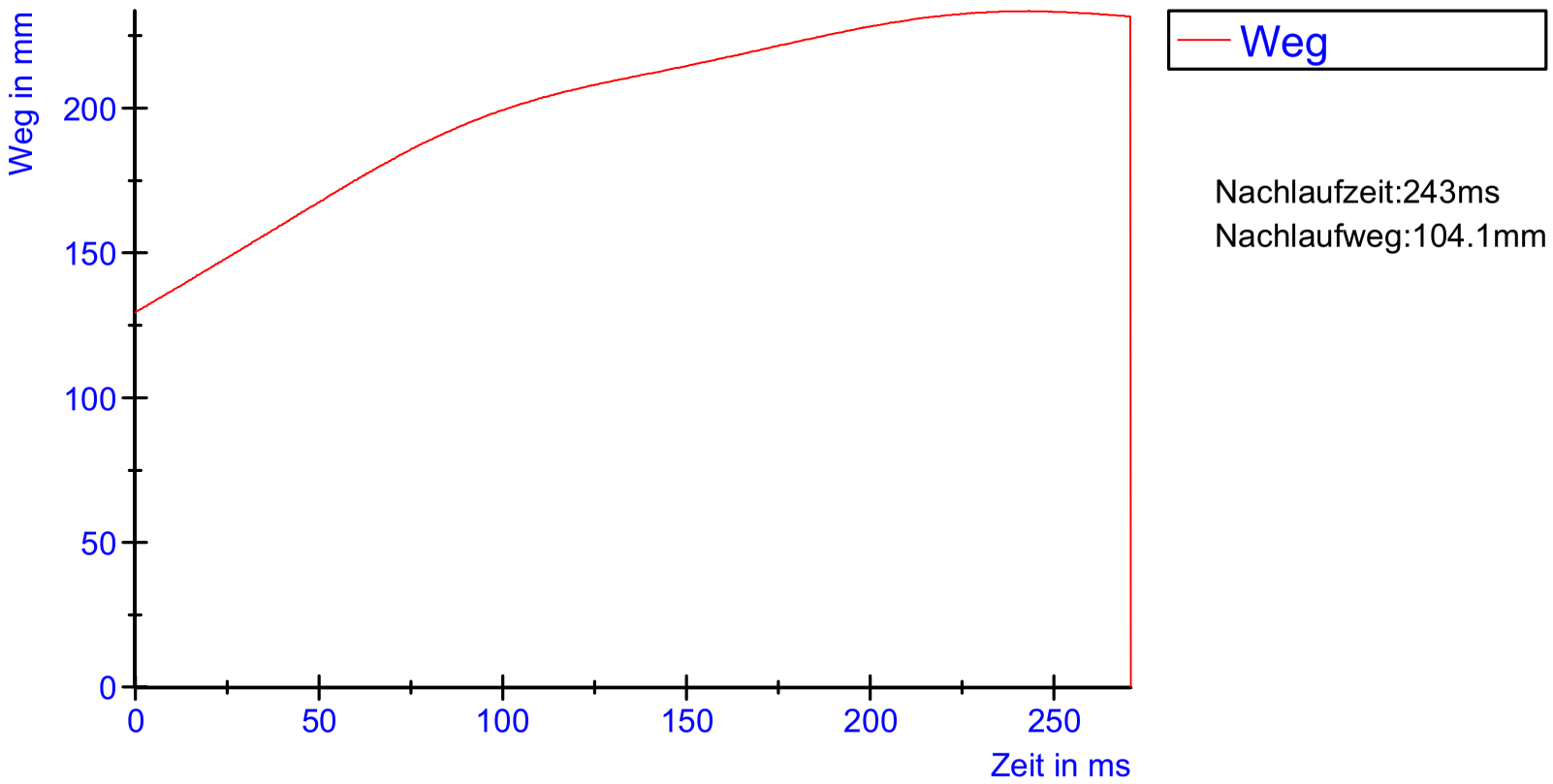
Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 1



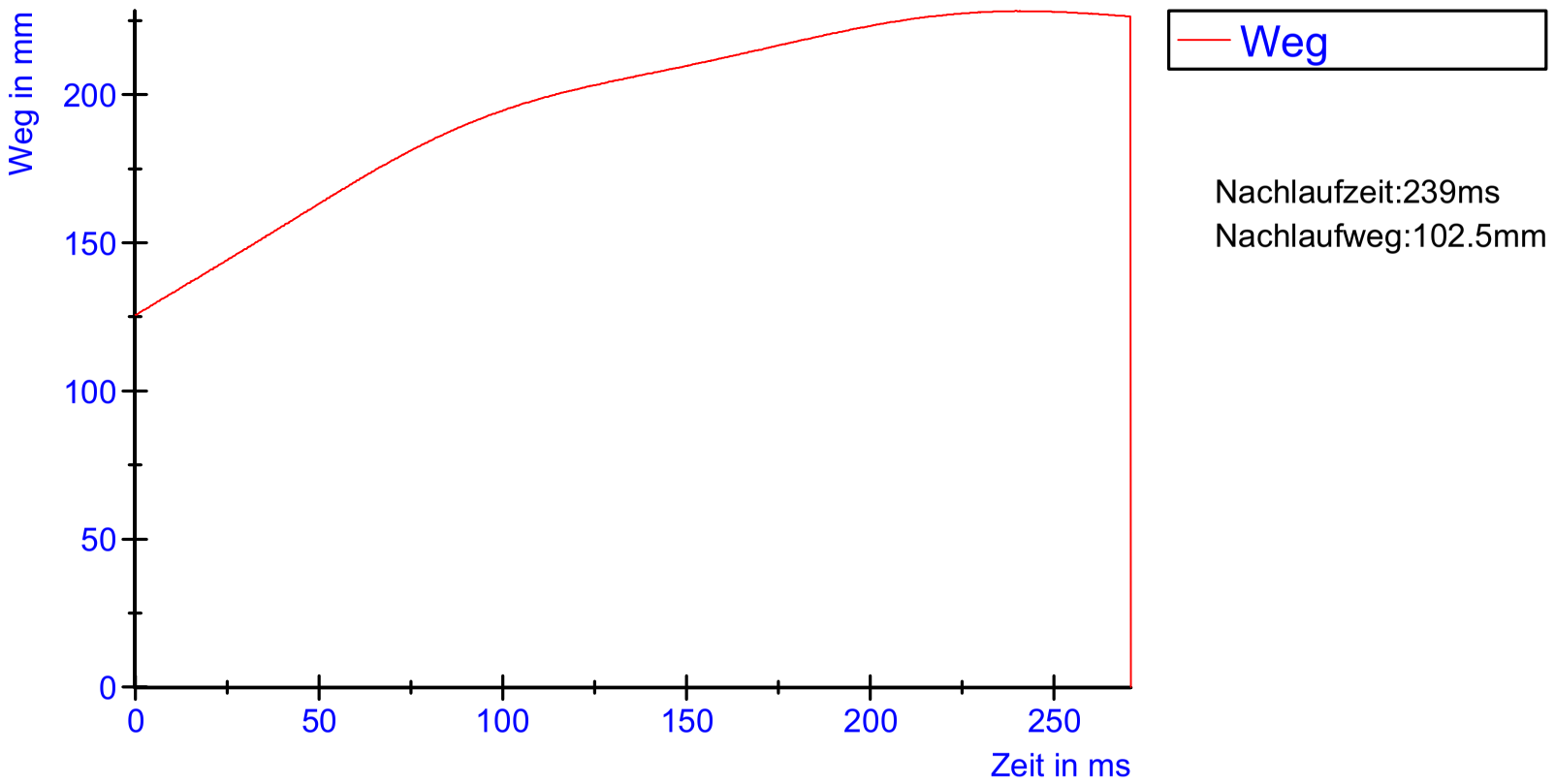
Nachlaufzeit:232ms
Nachlaufweg:98.5mm

Weg-Zeit-Verlauf der Messung 2



Nachlaufzeit:243ms
Nachlaufweg:104.1mm

Weg-Zeit-Verlauf der Messung 3

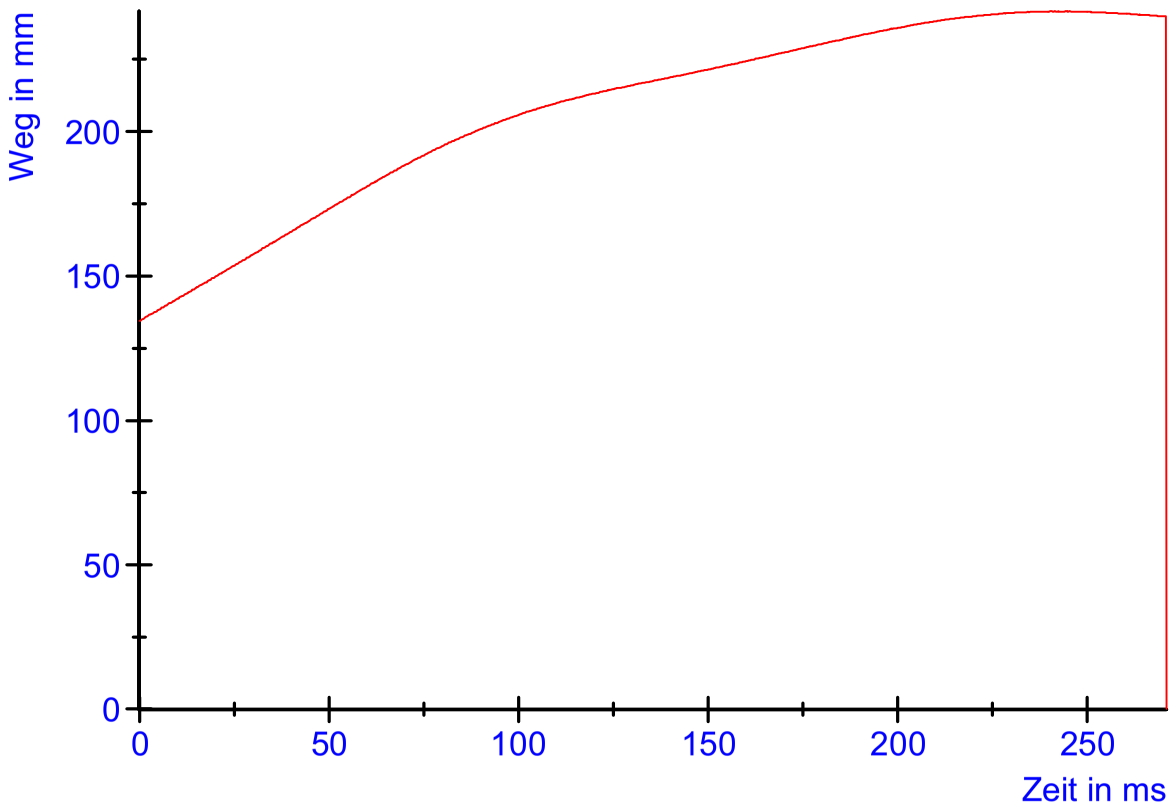


Nachlaufzeit:239ms
Nachlaufweg:102.5mm

Department Maschinenbau und Produktion

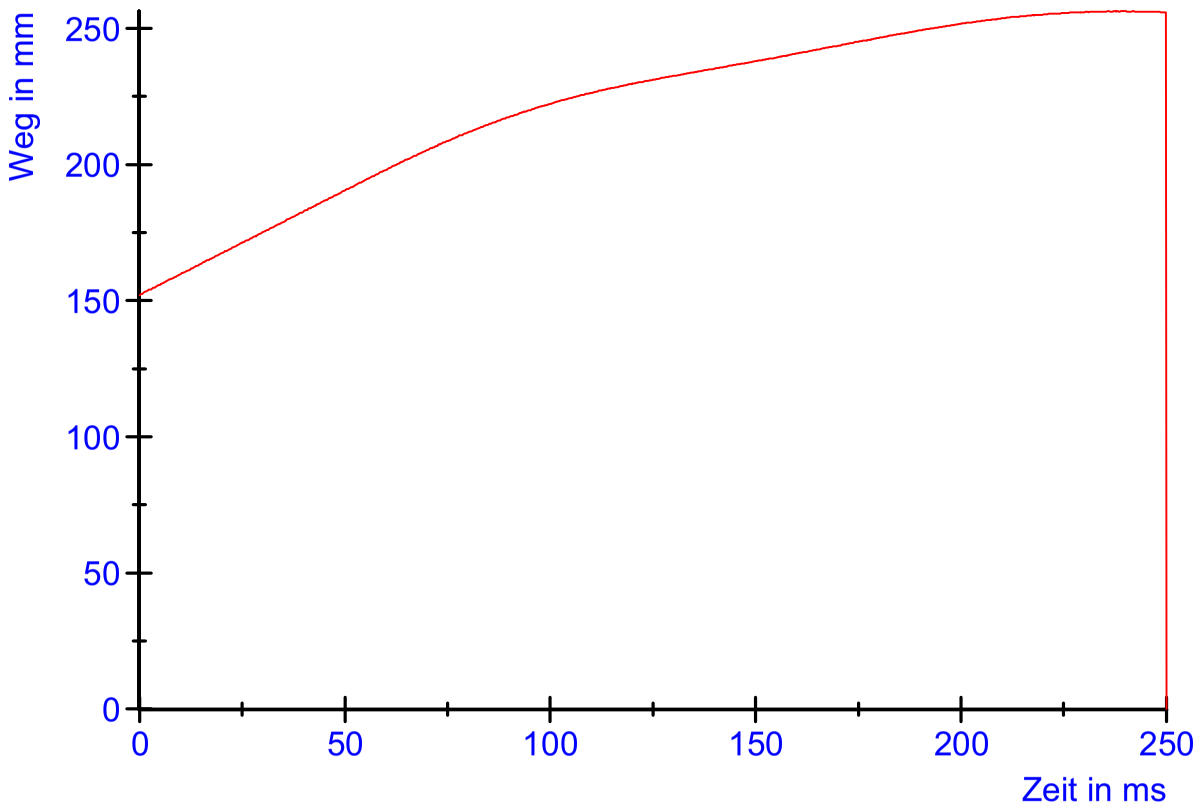
Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 4



Nachlaufzeit:240ms
Nachlaufweg:107.2mm

Weg-Zeit-Verlauf der Messung 5

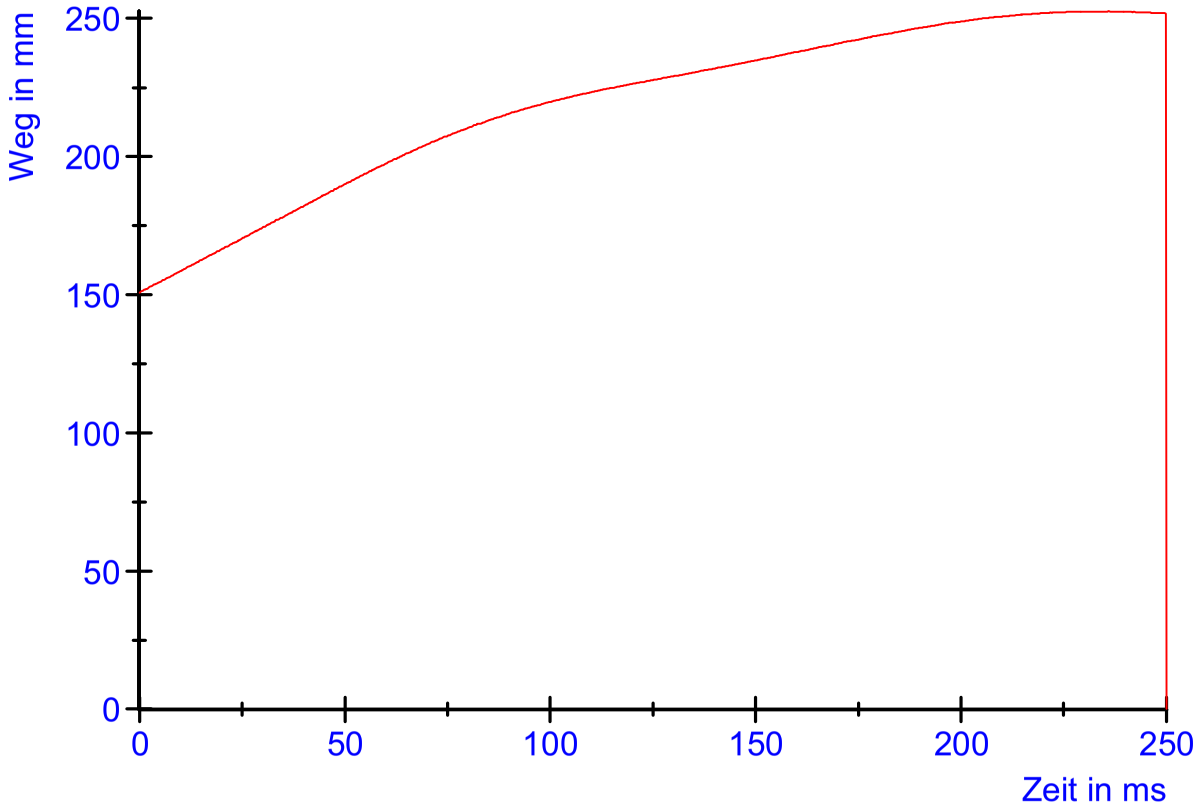


Nachlaufzeit:236ms
Nachlaufweg:104.1mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 6

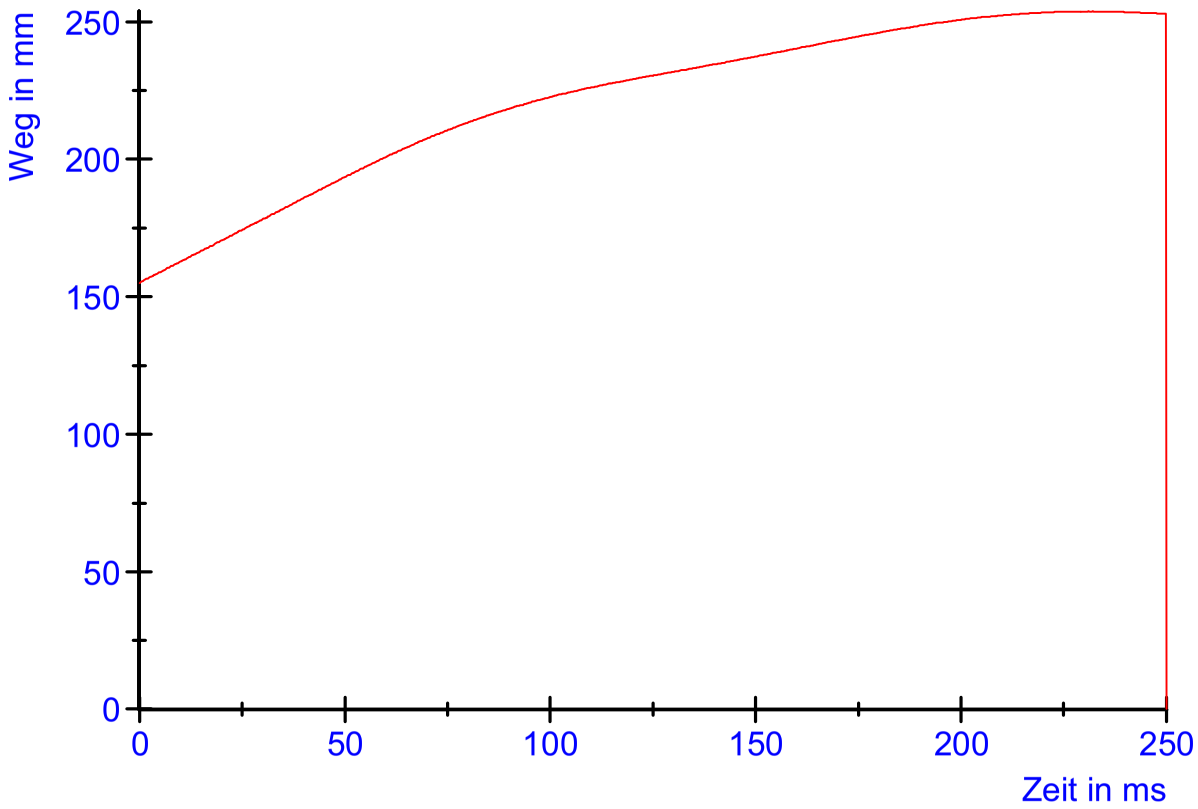


Nachlaufzeit:236ms
Nachlaufweg:101.8mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 7

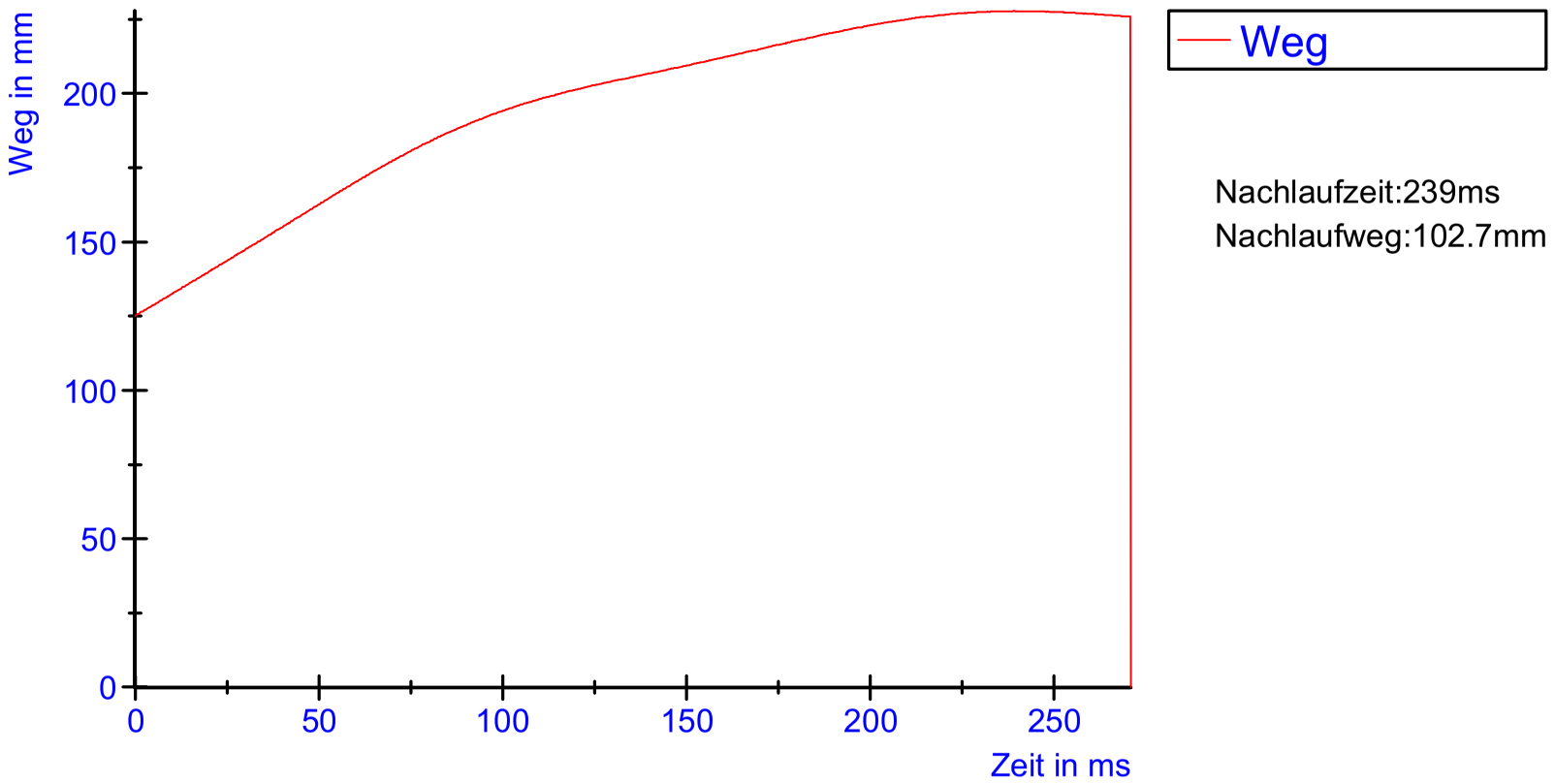


Nachlaufzeit:231ms
Nachlaufweg:98.7mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 8

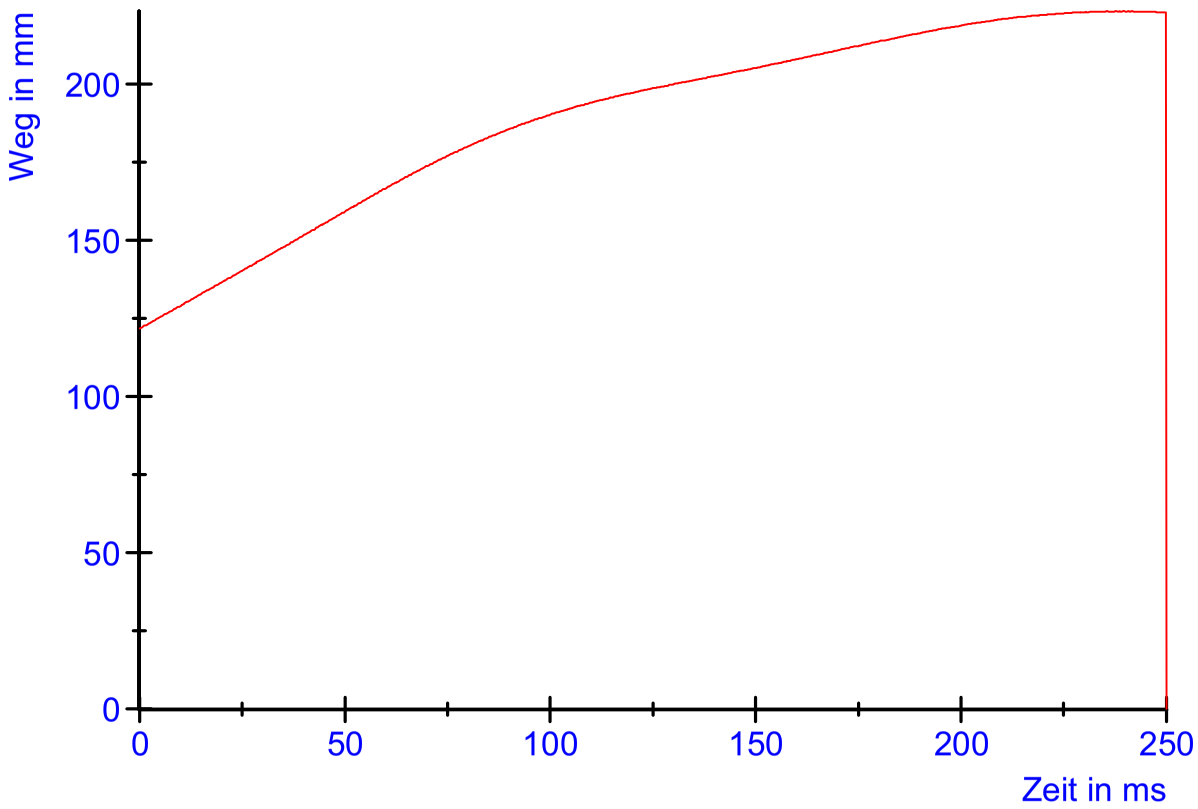


Nachlaufzeit:239ms
Nachlaufweg:102.7mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 9



Nachlaufzeit:235ms
Nachlaufweg:101.5mm

Vollständiges Protokoll einer Nachlaufwegmessung nach DIN 13855 mit den empfohlenen Parametern und eingebautem Werkzeug

Messergebnisse

Nachlaufweg in mm	Nachlaufzeit in ms
105.23	237
104.55	246
100.16	230
104.96	239
102.10	234
103.02	237
104.58	233
104.72	230
104.92	232
105.88	233

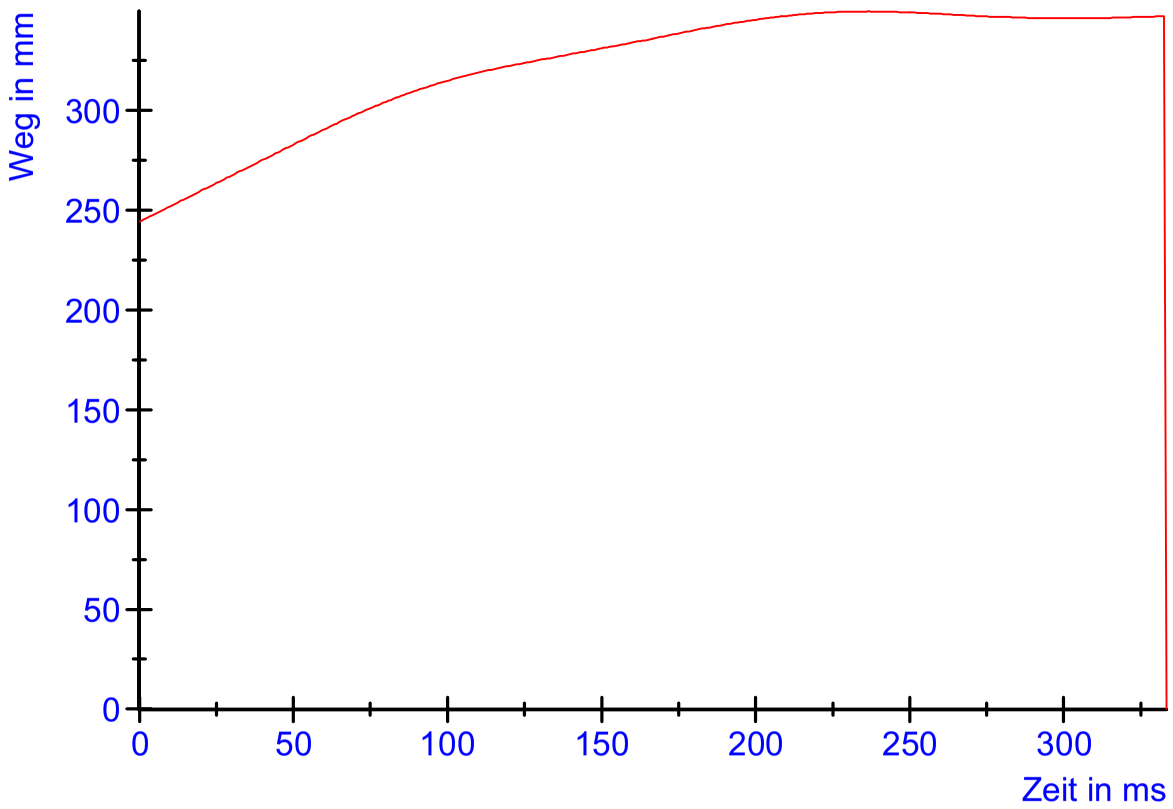
Maschinen-Identifikation: RZU 400
Schutzeinrichtung: Schutzvorhang
Messverfahren: Magnetisch striktiv
Messgerät: WayCon MAB-A-A-850-N
Verifizierung: 2011
Schaltzeit der Schutzeinrichtung: 41ms
Annäherungsgeschwindigkeit: 1.6m/s bei S>=500mm
2m/s bei S<500mm

gemessene maximale Nachlaufzeit: 245.83ms
gemessener Nachlaufweg bei maximaler Nachlaufzeit: 104.55mm
Mittelwert + 3Standardabweichungen: 249.49ms
resultierender Sicherheitsabstand: 500mm

_____

Prüfer: Markus von Hebel
Datum:02/08/2011

Weg-Zeit-Verlauf der Messung 0



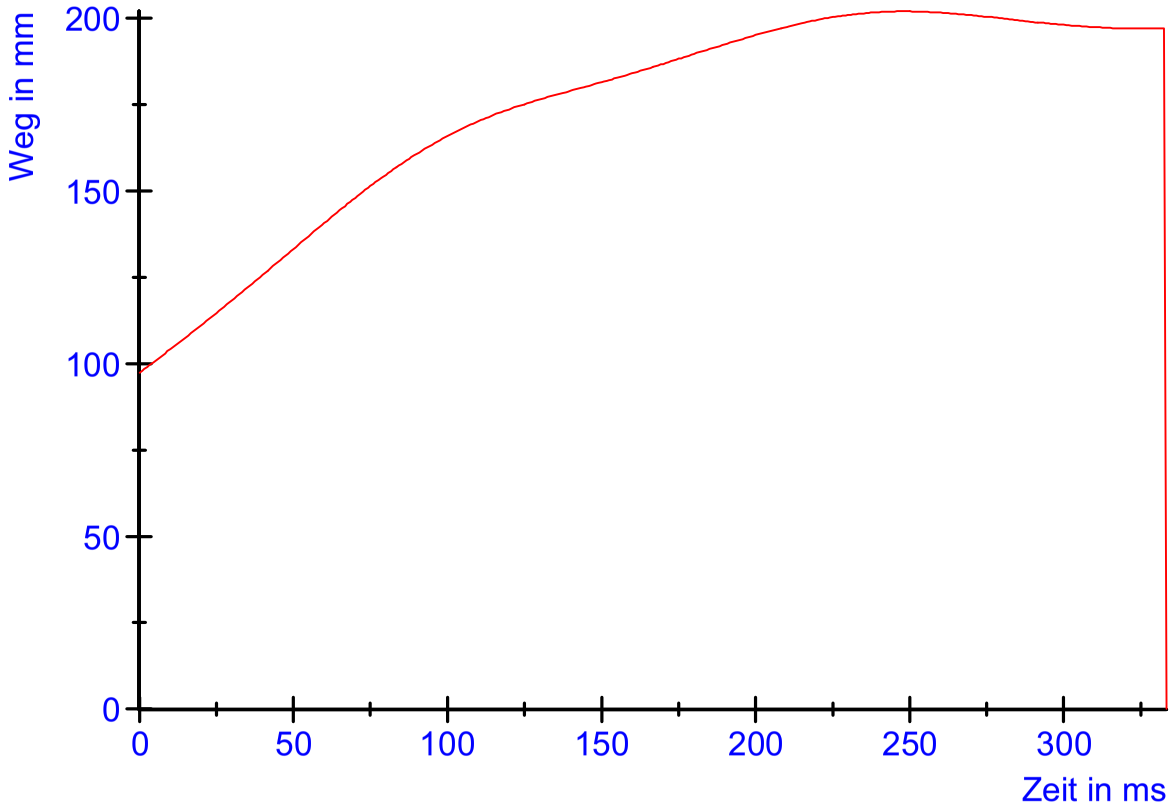
Weg

Nachlaufzeit:237ms
Nachlaufweg:105.23mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 1

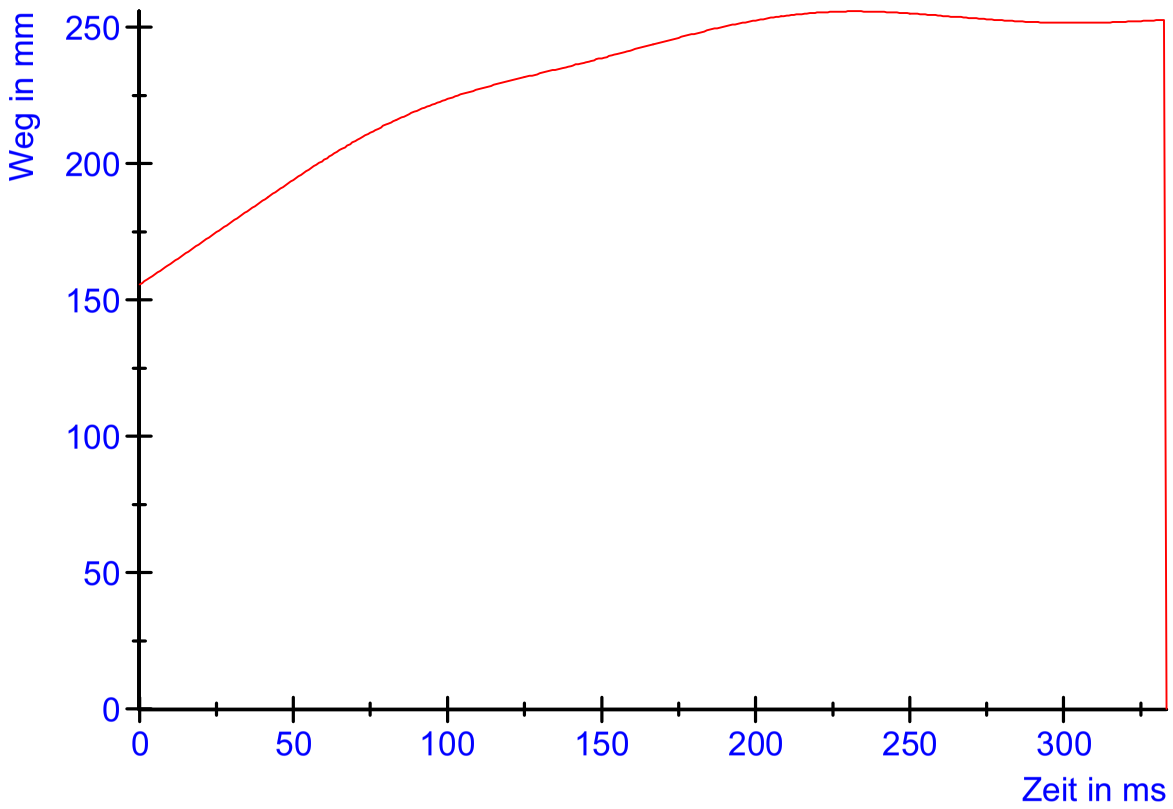


Nachlaufzeit:246ms
Nachlaufweg:105.23mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 2

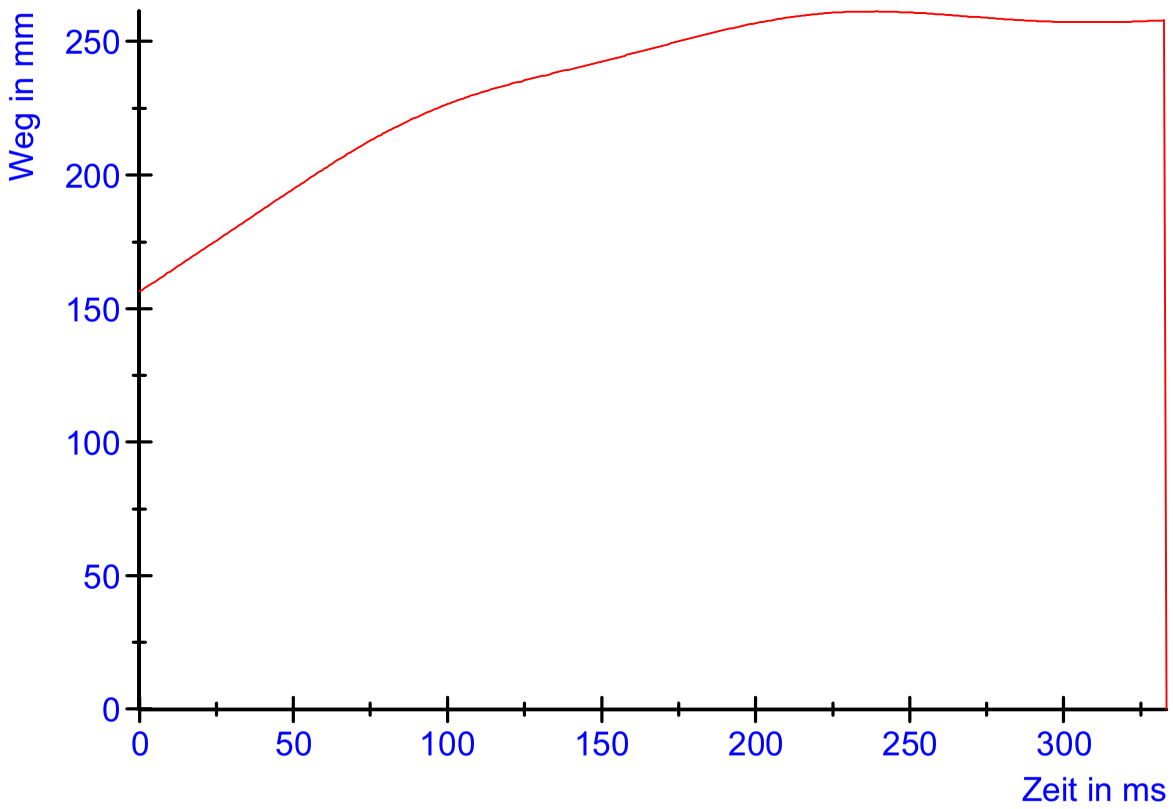


Nachlaufzeit:230ms
Nachlaufweg:100.16mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 3

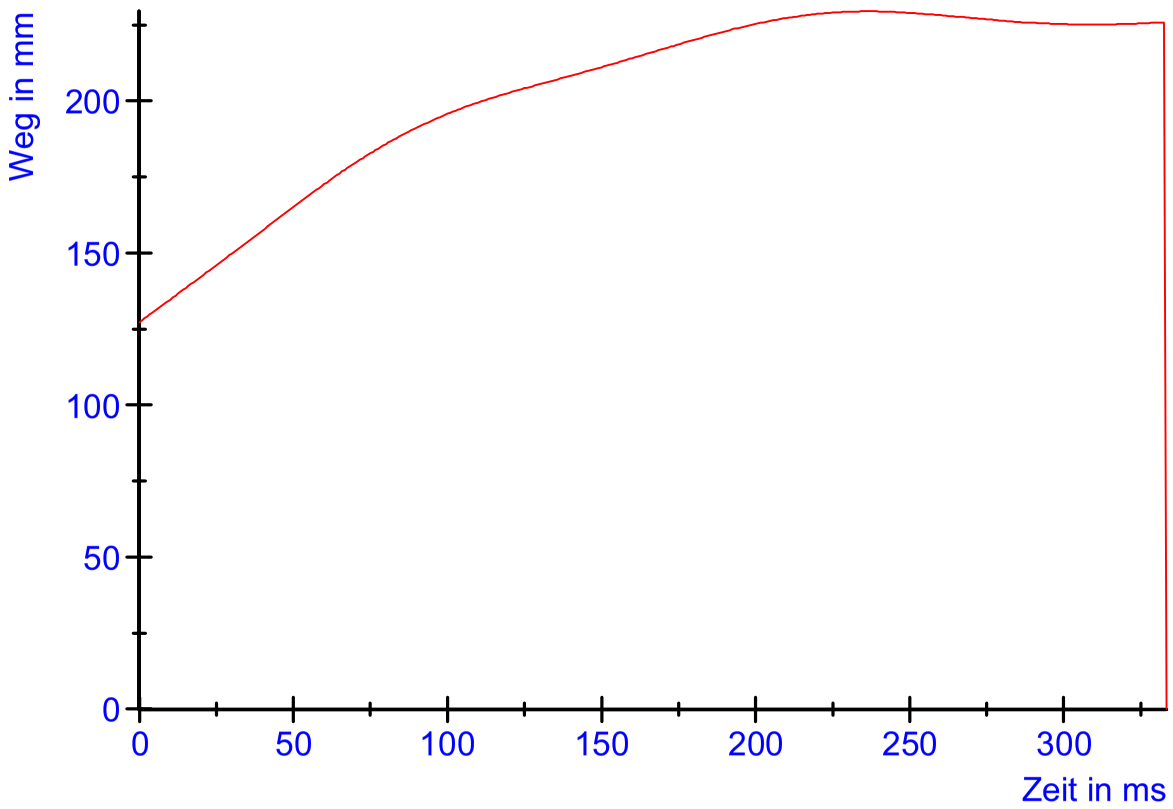


Nachlaufzeit:239ms
Nachlaufweg:104.96mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 4

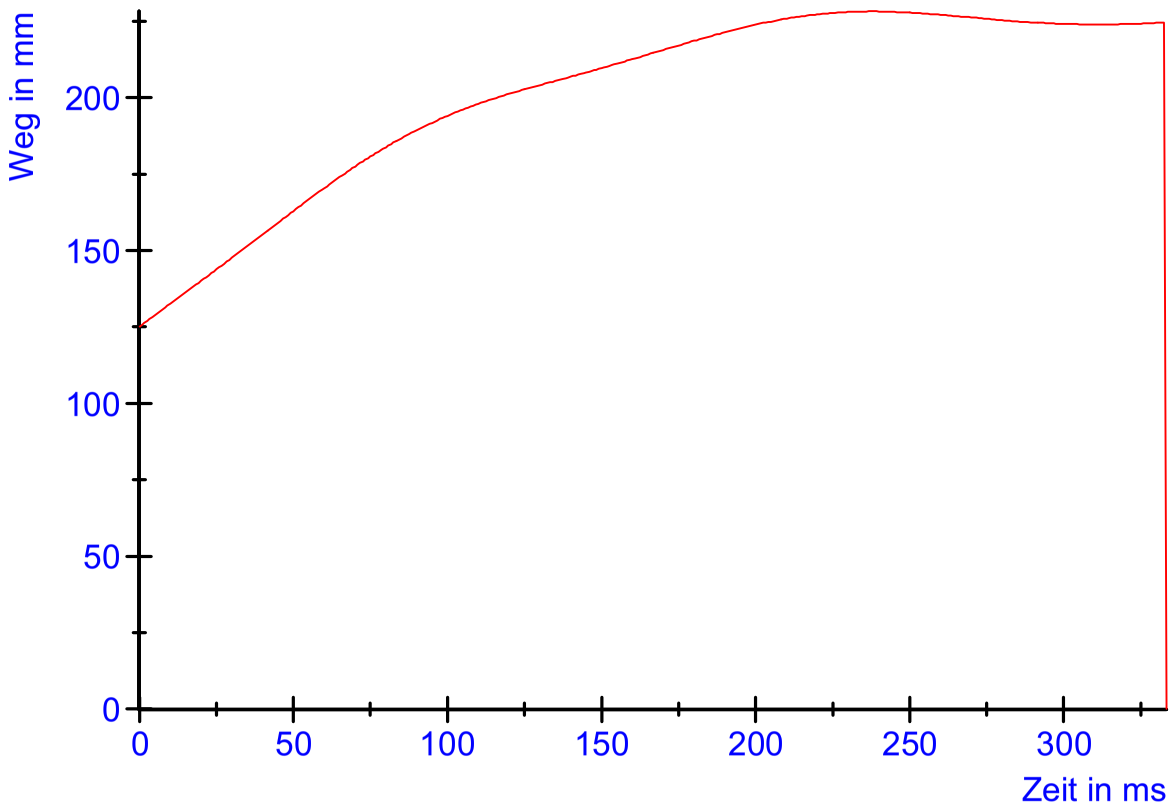


Nachlaufzeit:234ms
Nachlaufweg:102.10mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 5

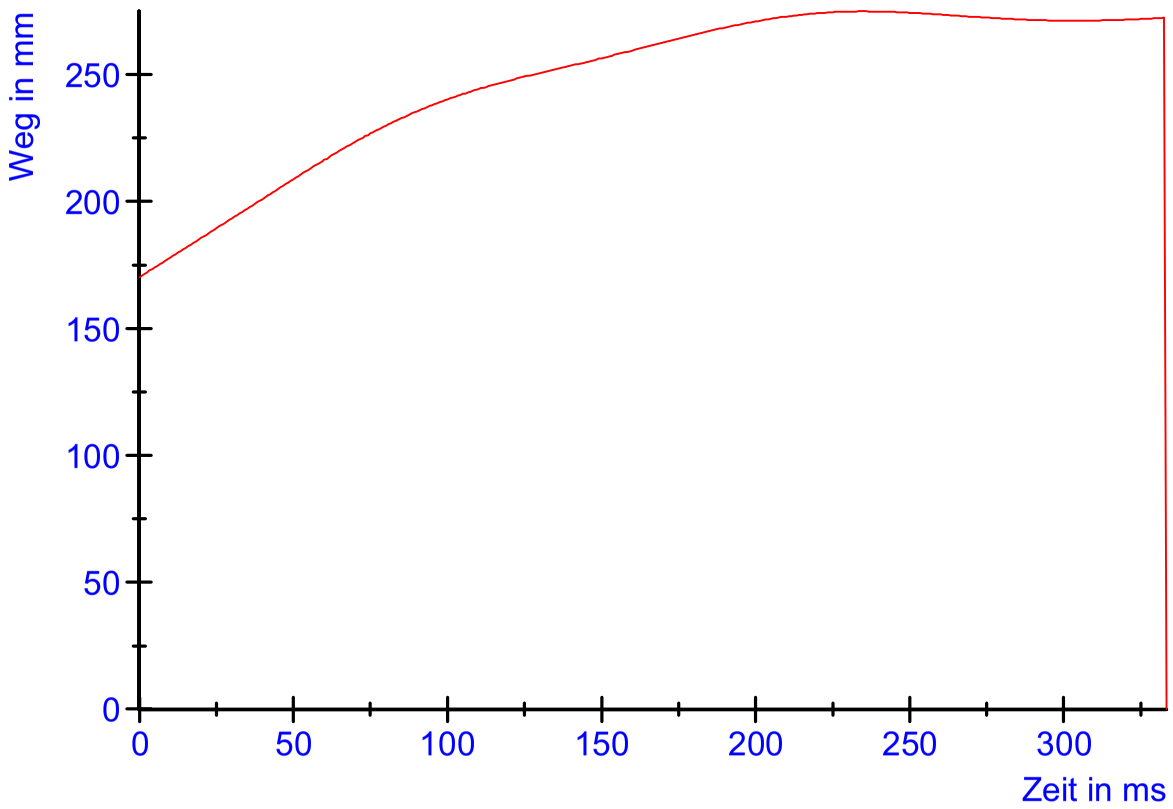


Nachlaufzeit:237ms
Nachlaufweg:103.02mm

Department Maschinenbau und Produktion

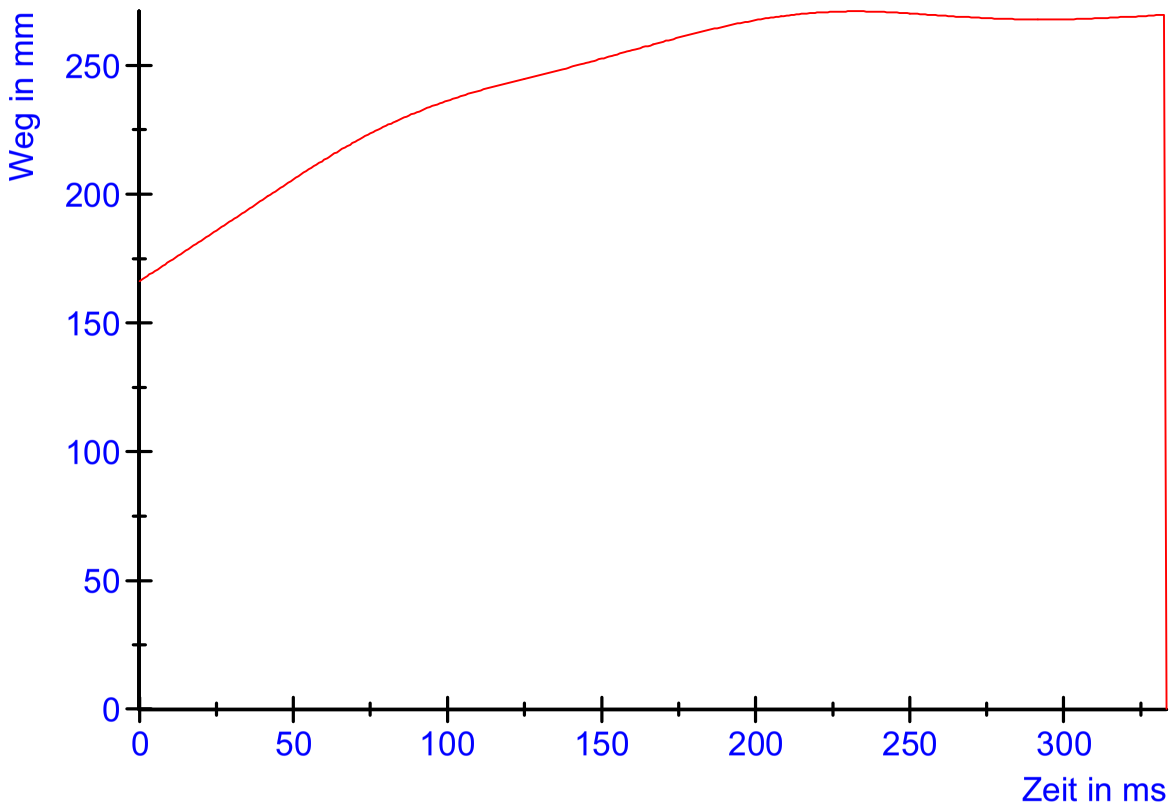
Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 6



Nachlaufzeit:233ms
Nachlaufweg:104.58mm

Weg-Zeit-Verlauf der Messung 7

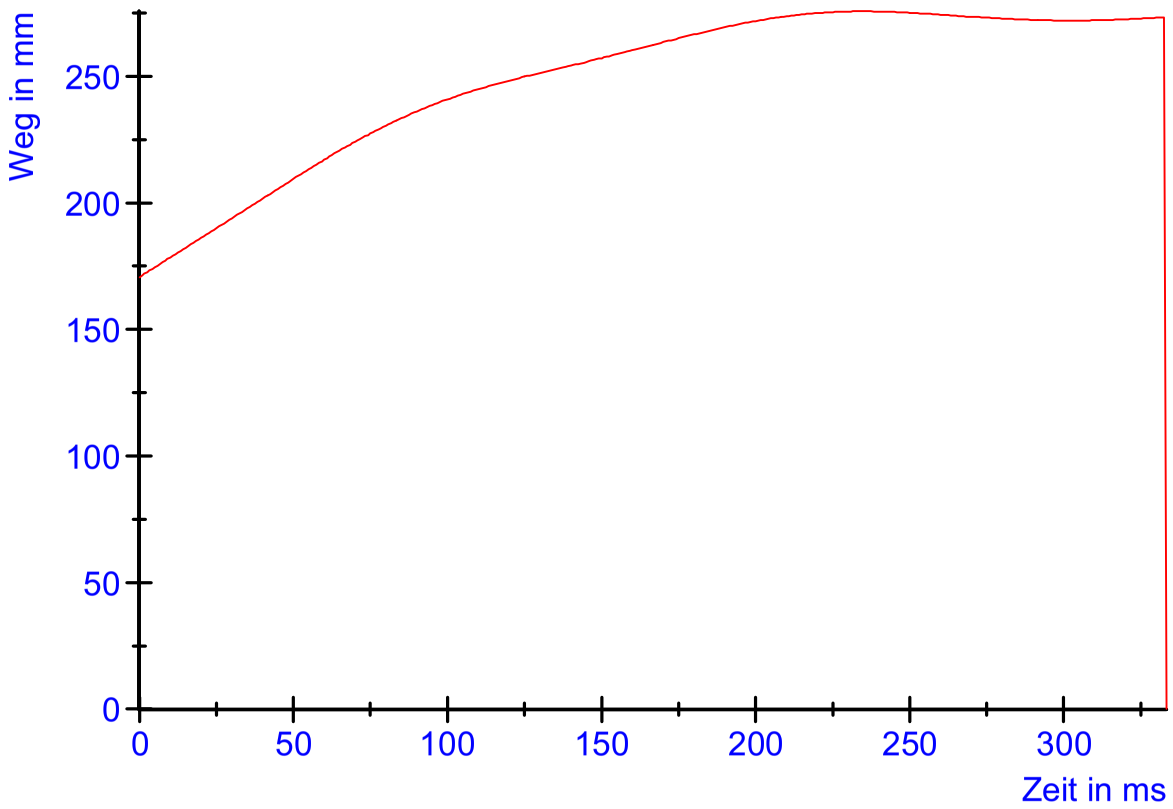


Nachlaufzeit:230ms
Nachlaufweg:104.72mm

Department Maschinenbau und Produktion

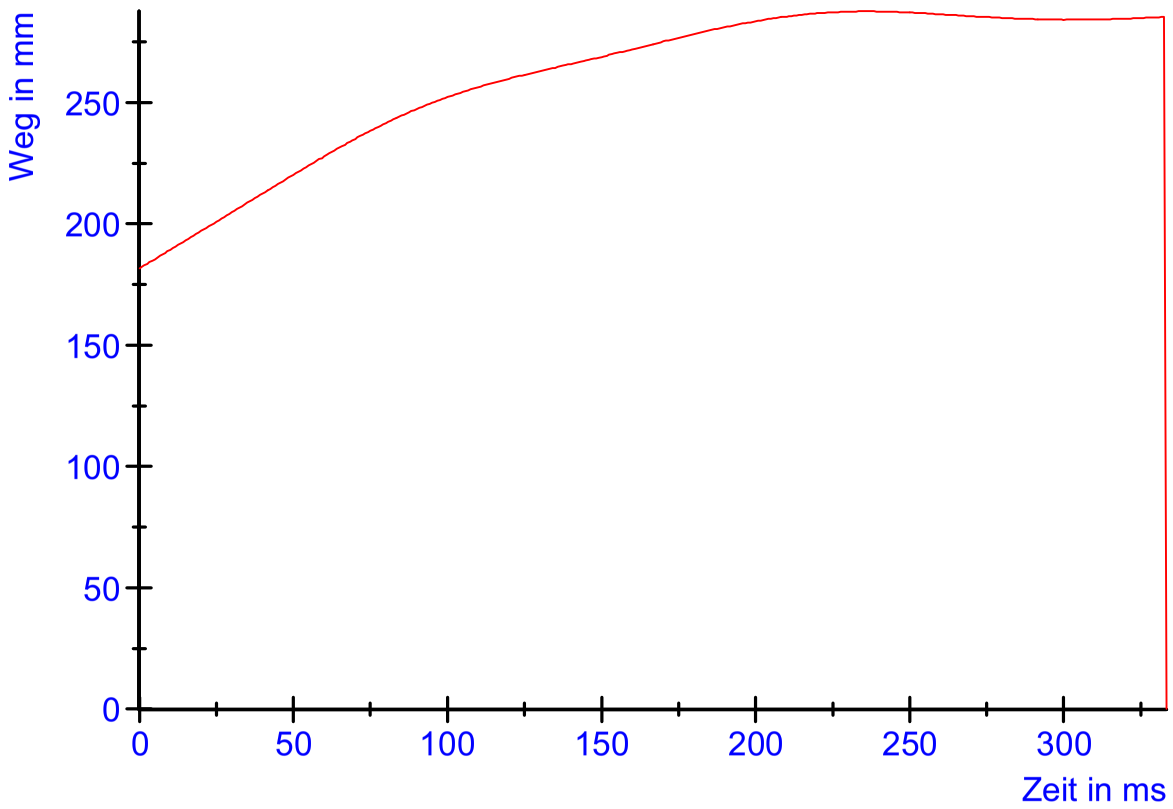
Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 8



Nachlaufzeit:232ms
Nachlaufweg:104.92mm

Weg-Zeit-Verlauf der Messung 9



Nachlaufzeit:233ms
Nachlaufweg:105.88mm

Vollständiges Protokoll einer Nachlaufwegmessung nach DIN 13855 mit den empfohlenen Parametern ohne Werkzeug

Messergebnisse

Nachlaufweg in mm	Nachlaufzeit in ms
82.31	217
89.76	223
82.62	217
85.17	221
85.92	221
86.39	218
87.24	217
86.60	225
90.34	223
87.52	225

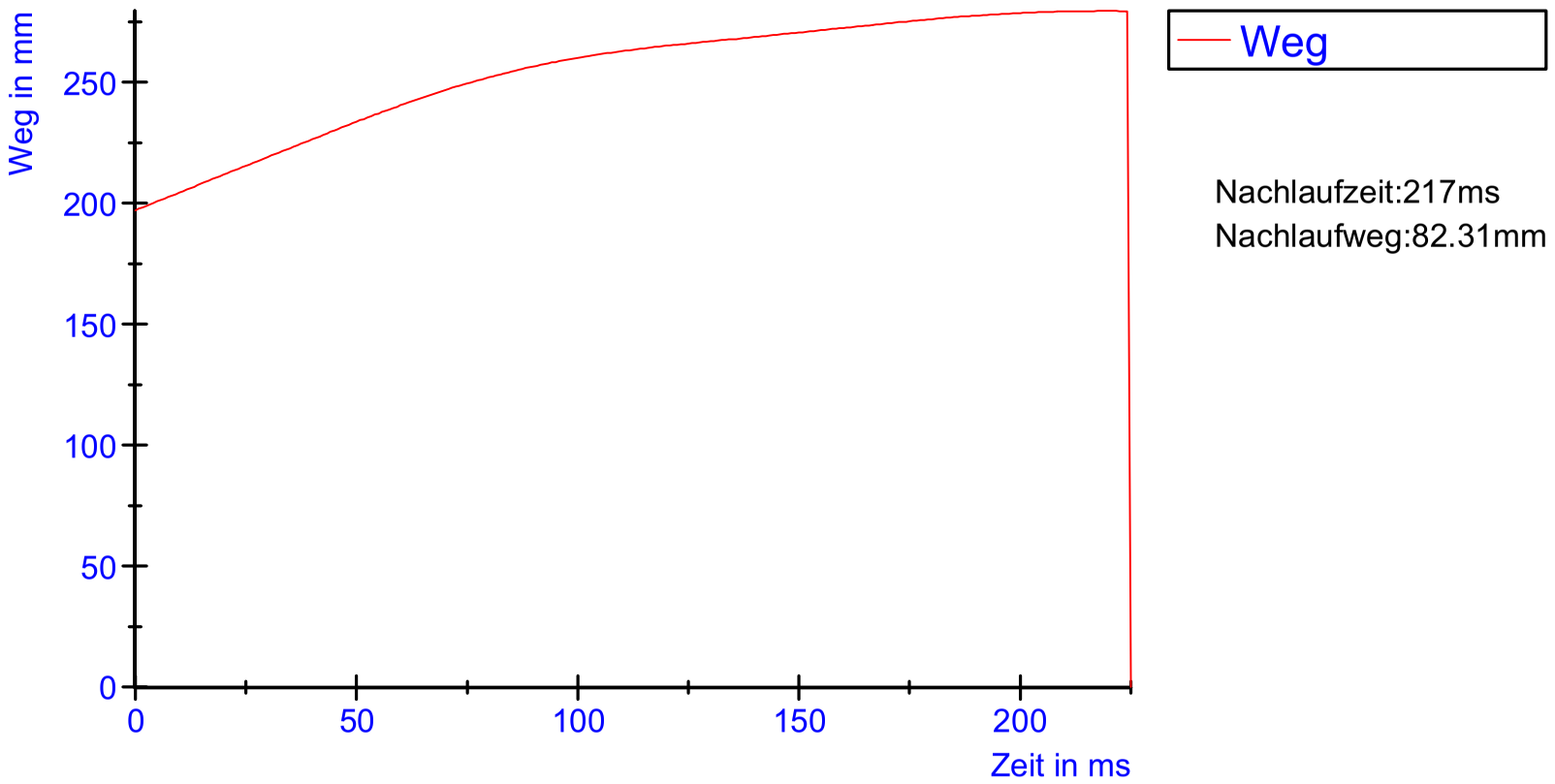
Maschinen-Identifikation: RZU 400
Schutzeinrichtung: Schutzvorhang
Messverfahren: Magnetisch striktiv
Messgerät: WayCon MAB-A-A-850-N
Verifizierung: 2011
Schaltzeit der Schutzeinrichtung: 41ms
Annäherungsgeschwindigkeit: 1.6m/s bei S>=500mm
2m/s bei S<500mm

gemessene maximale Nachlaufzeit: 225ms
gemessener Nachlaufweg bei maximaler Nachlaufzeit: 87.52mm
Mittelwert + 3Standardabweichungen: 229.93ms
resultierender Sicherheitsabstand: 500mm

_____

Prüfer: Markus von Hebel
Datum:02/08/2011

Weg-Zeit-Verlauf der Messung 0

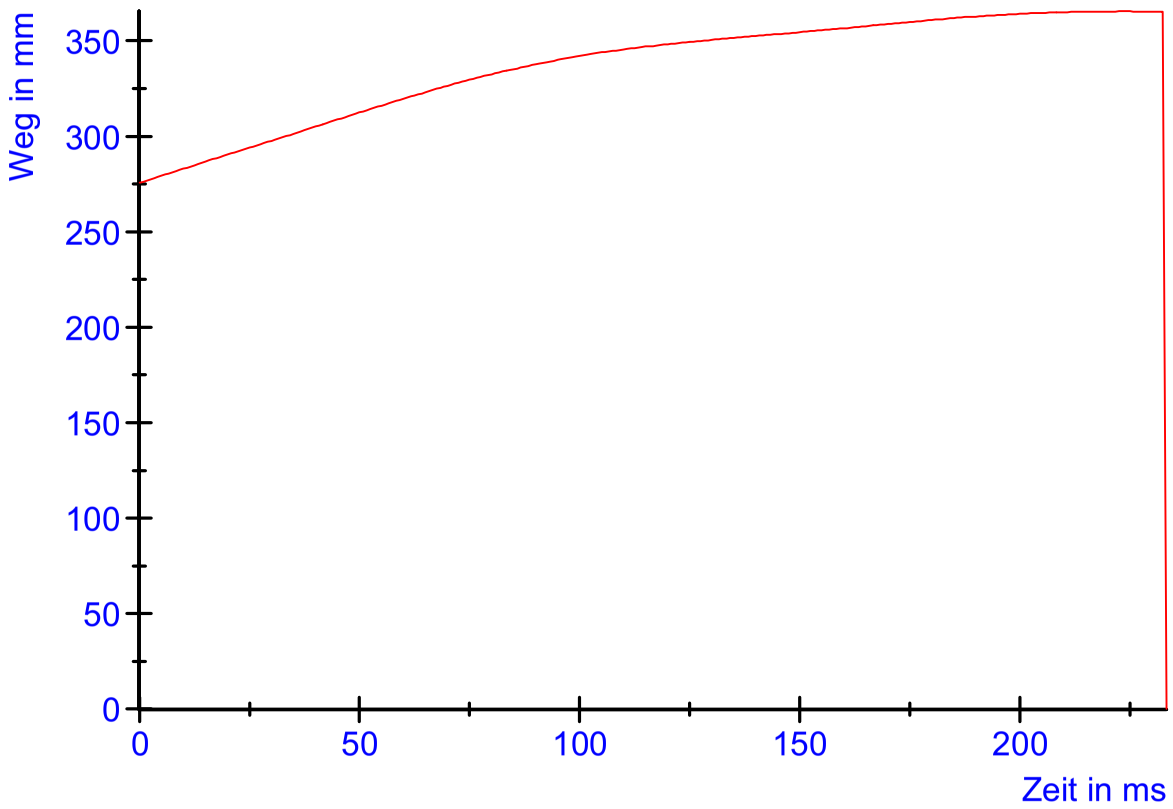


Nachlaufzeit:217ms
Nachlaufweg:82.31mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 1



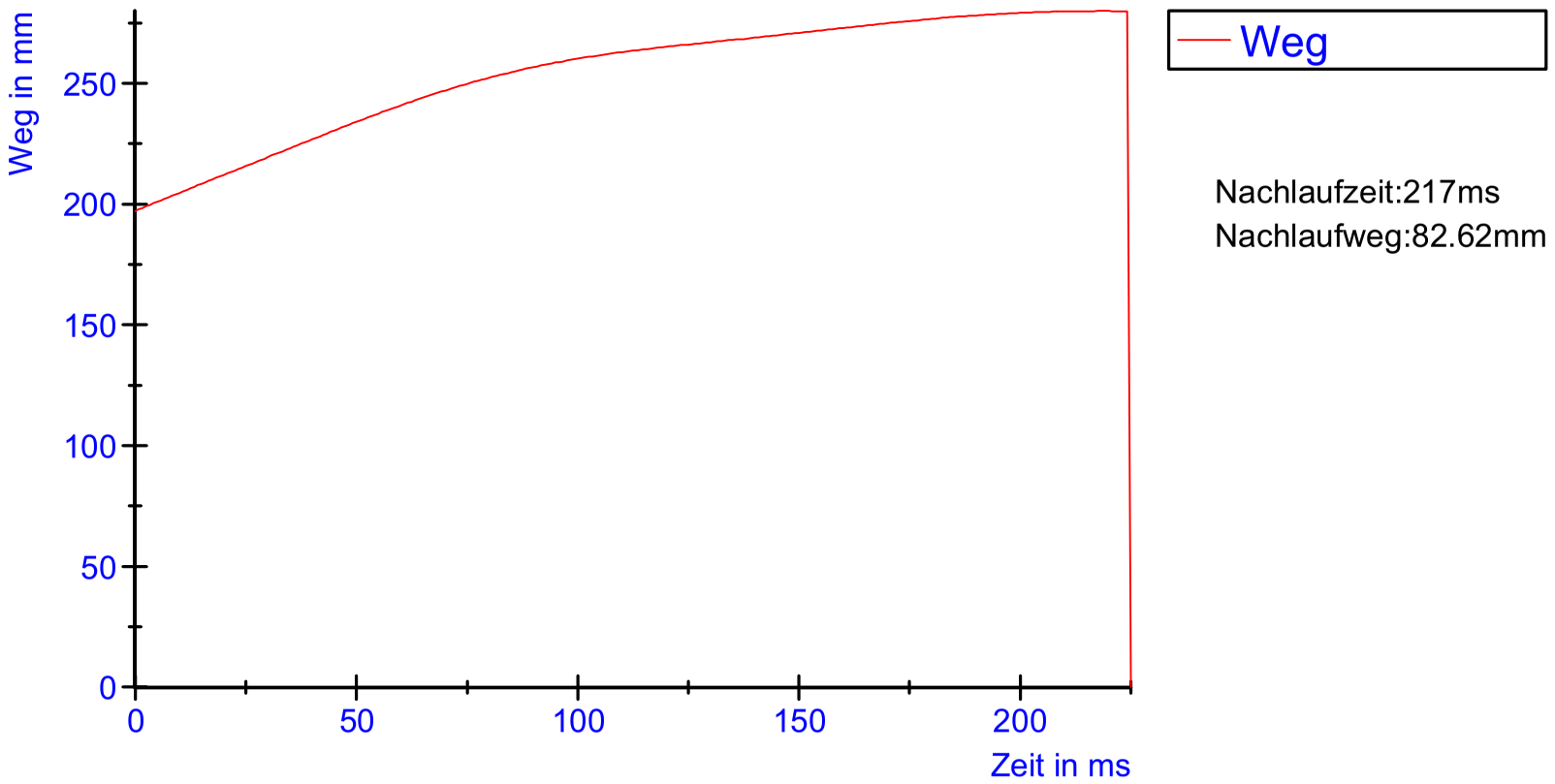
Weg

Nachlaufzeit:223ms
Nachlaufweg:89.76mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 2

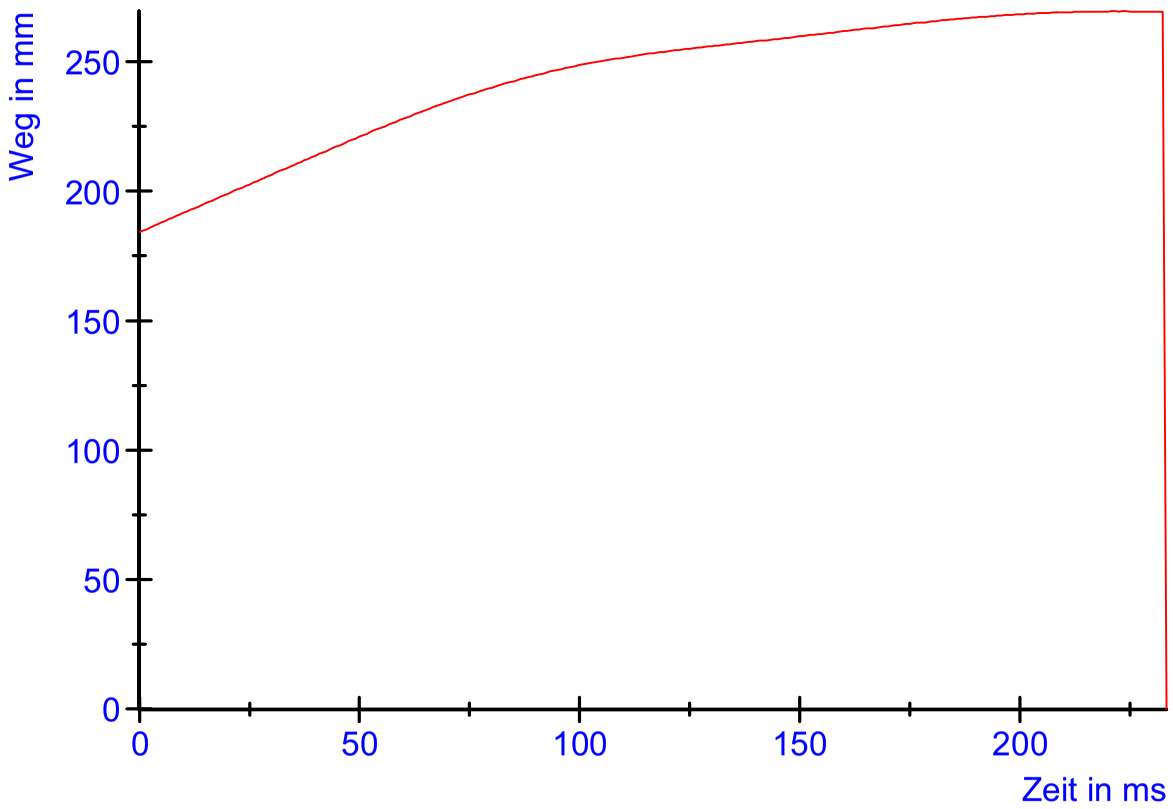


Nachlaufzeit:217ms
Nachlaufweg:82.62mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 3



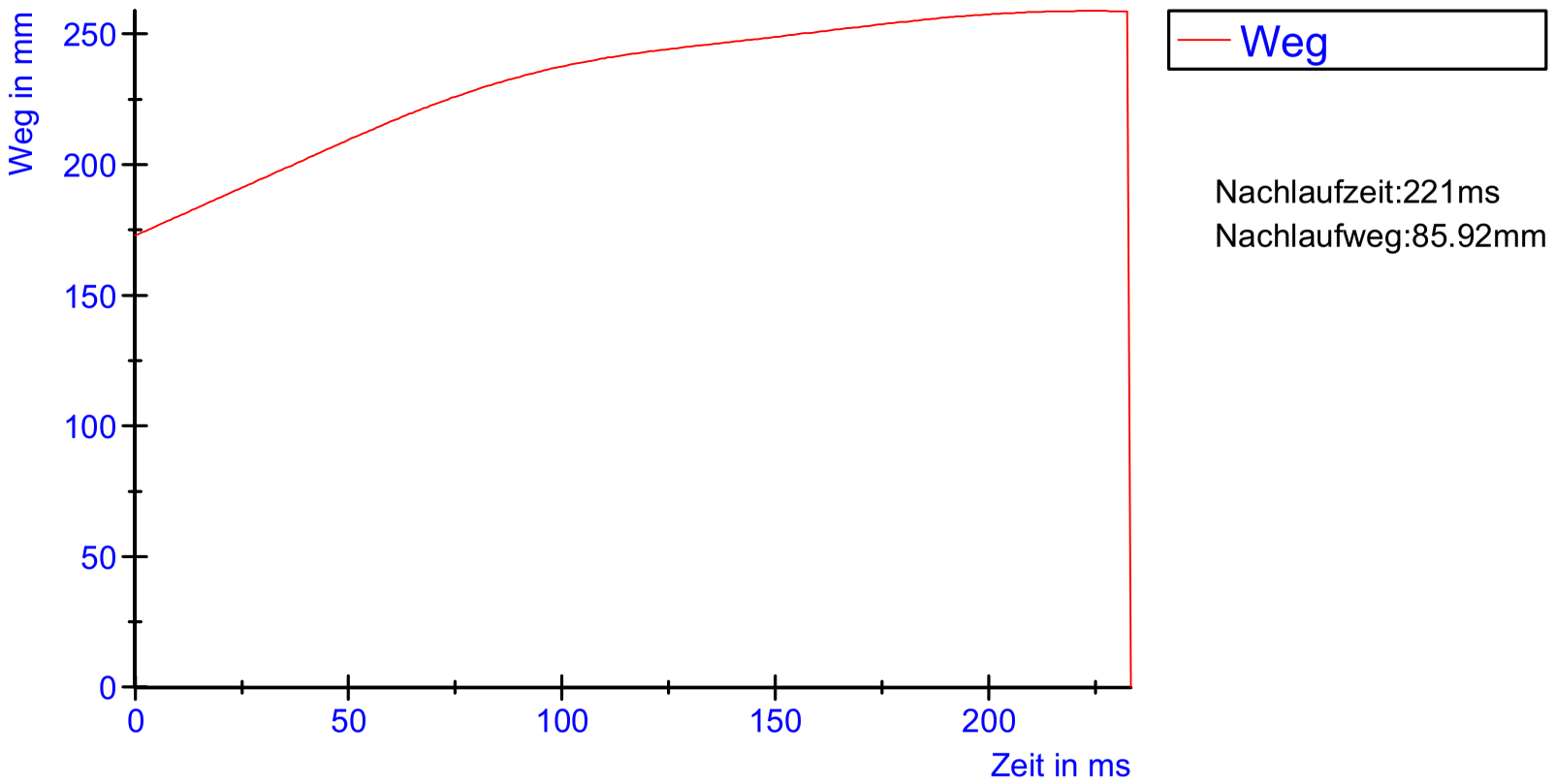
Weg

Nachlaufzeit:221ms
Nachlaufweg:85.17mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 4

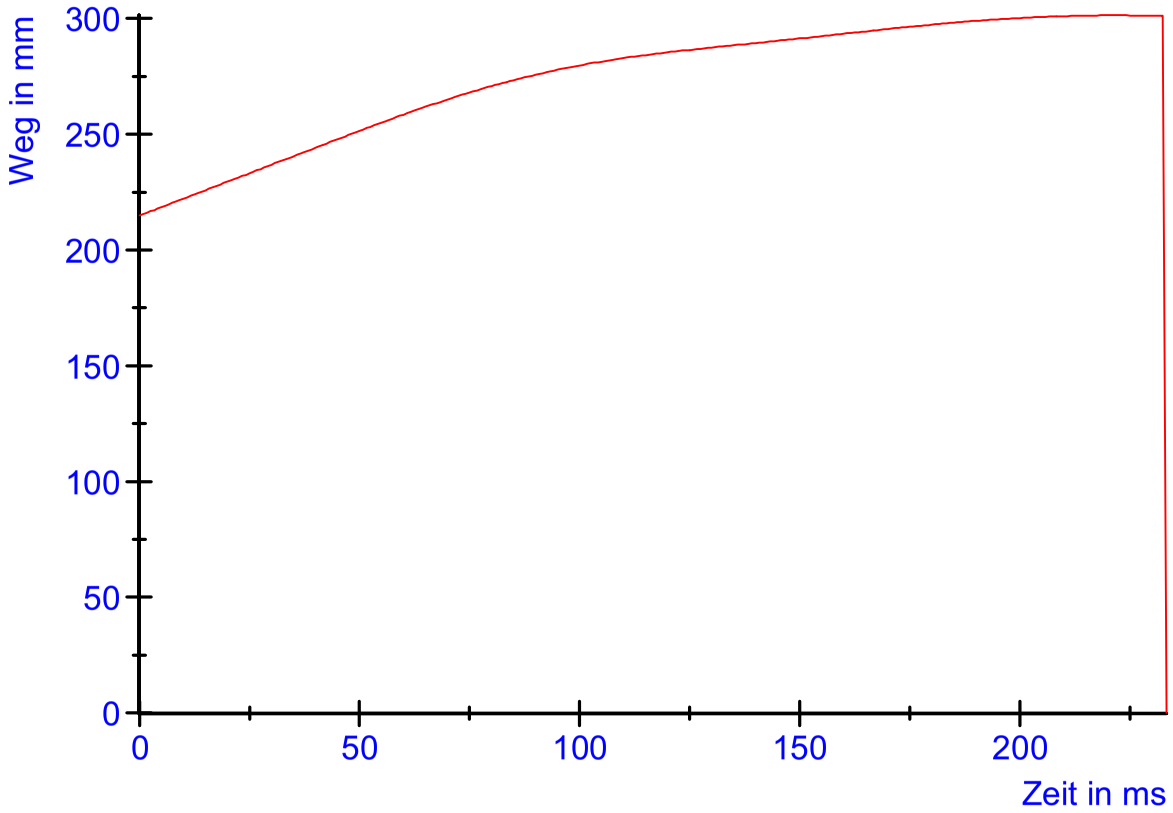


Nachlaufzeit:221ms
Nachlaufweg:85.92mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 5



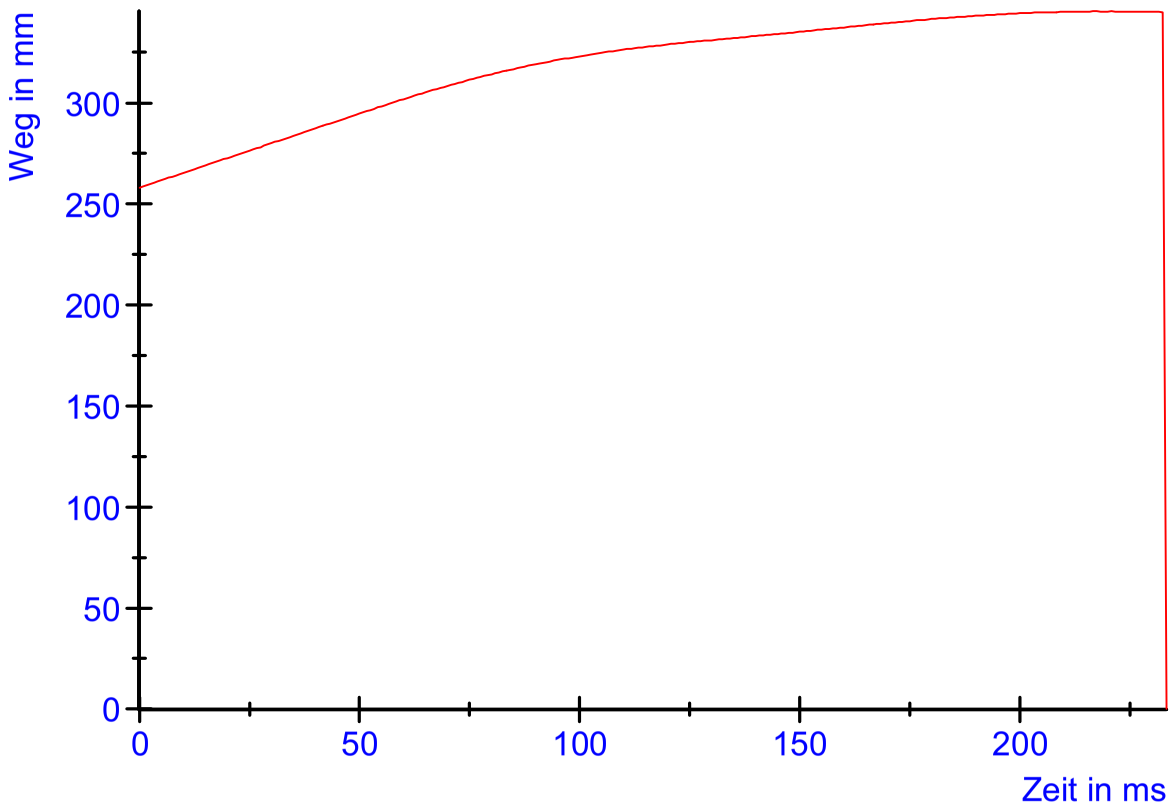
Weg

Nachlaufzeit:218.39ms
Nachlaufweg:86.39mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 6



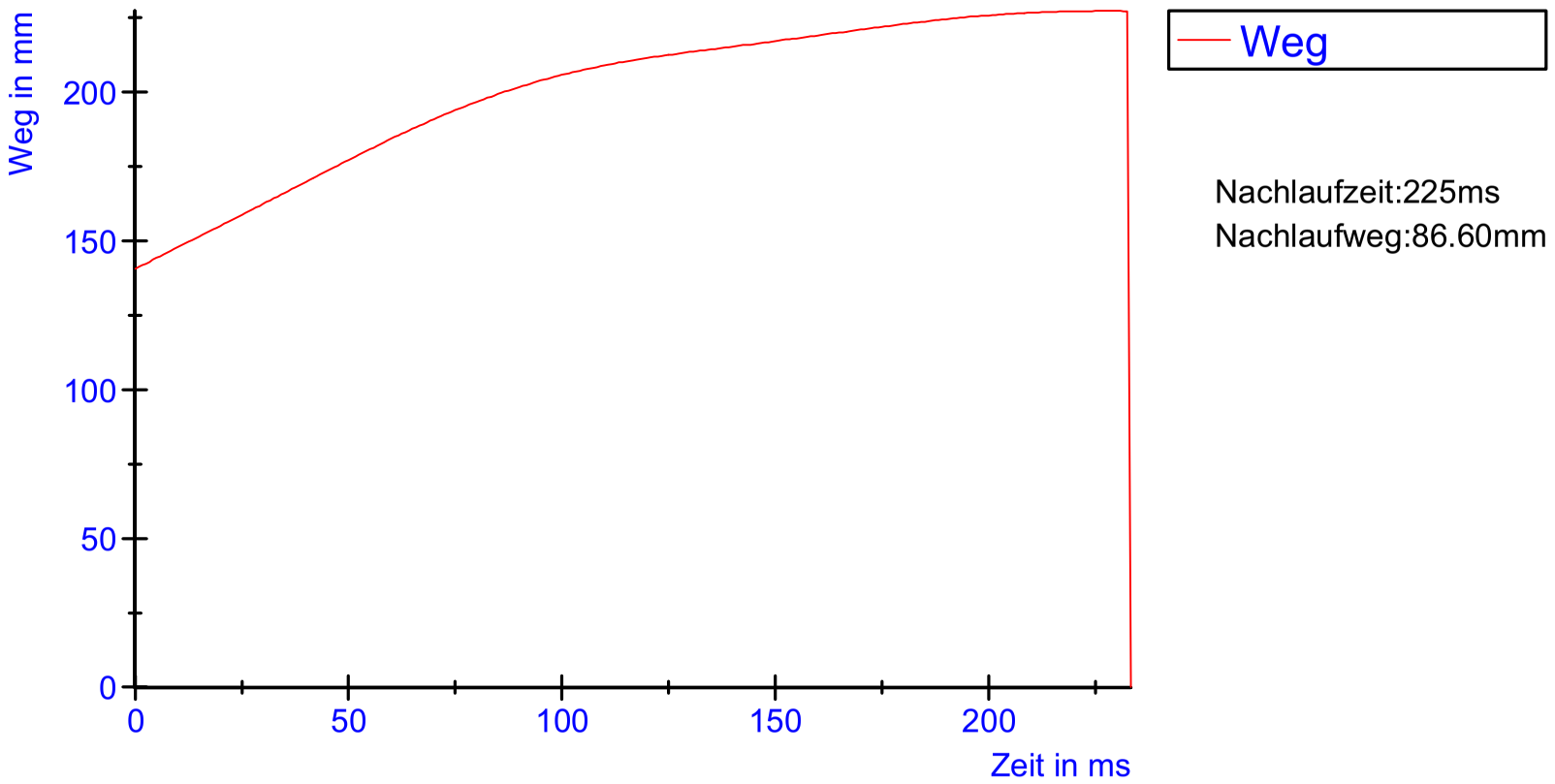
Weg

Nachlaufzeit:217ms
Nachlaufweg:87.24mm

Department Maschinenbau und Produktion

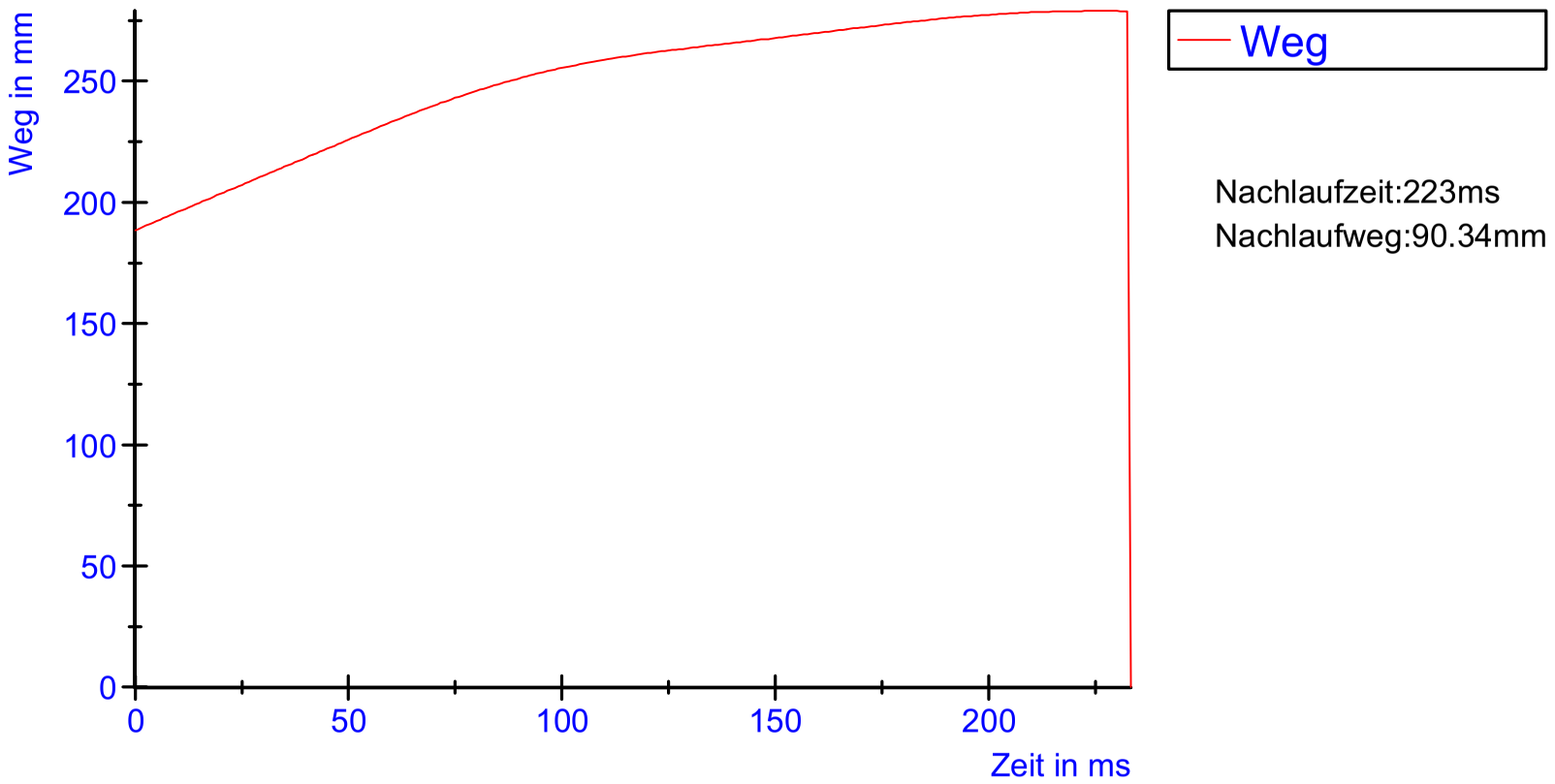
Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 7



Nachlaufzeit:225ms
Nachlaufweg:86.60mm

Weg-Zeit-Verlauf der Messung 8

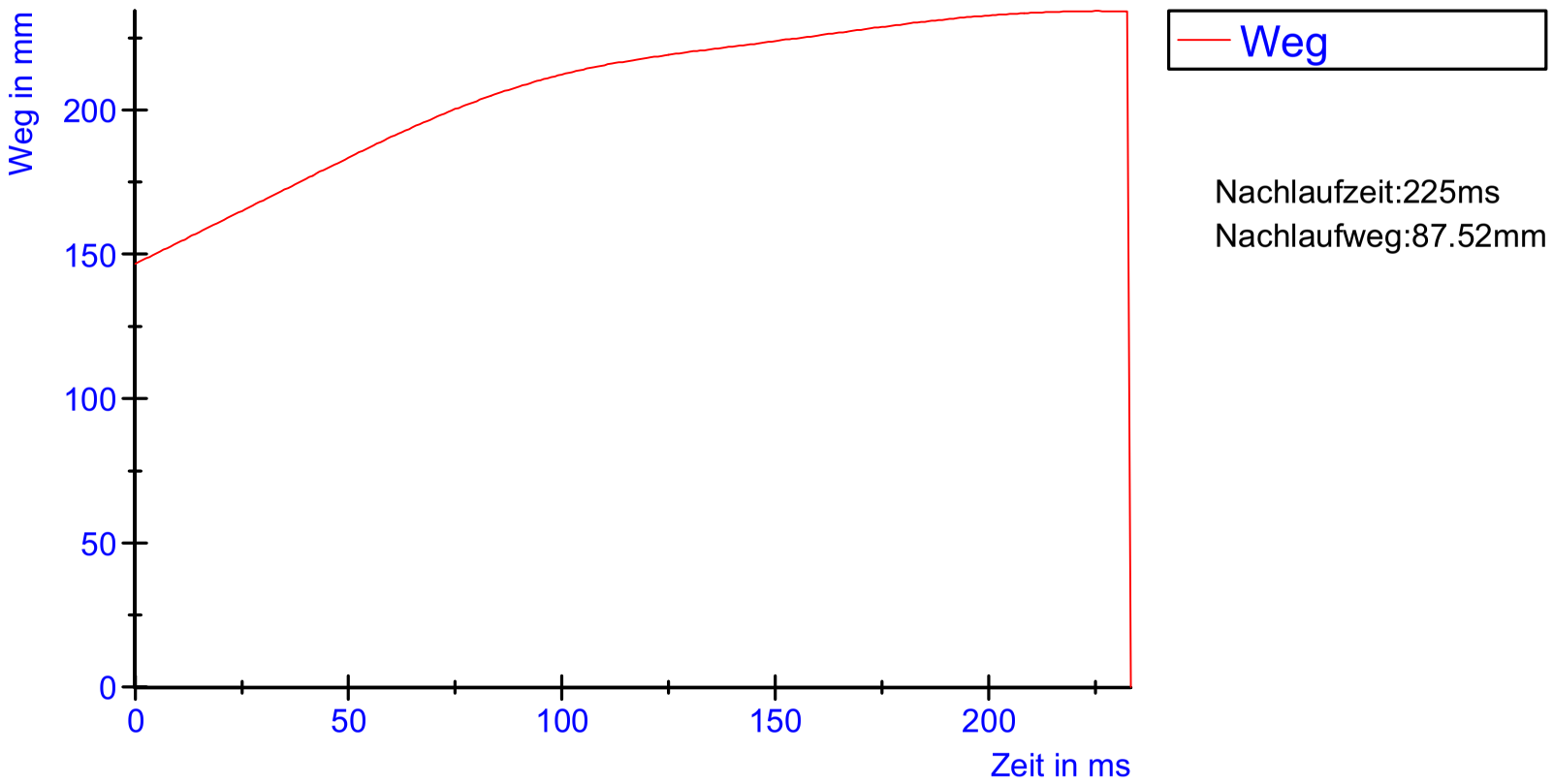


Nachlaufzeit:223ms
Nachlaufweg:90.34mm

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Weg-Zeit-Verlauf der Messung 9



Nachlaufzeit:225ms
Nachlaufweg:87.52mm

Zu 6.2

Erstellen von Programmen zur Messung und Steuerung eines Messverstärkers des Typs Spider8 in LabVIEW2010

Erstellt von Markus von Hebel

Inhaltverzeichnis

Vorwort

Diese Anleitung soll es dem Leser ermöglichen selbstständig Programme zu erstellen, die auf einen Messverstärker vom Typ Spider8 zugreifen bzw. ihn steuern. Dabei sollten Vorkenntnisse im Bereich LabVIEW vorhanden sein, die z. B. durchs Bearbeiten des Basic I –Kurses von National Instruments erworben werden können. Das Kapitel 4 „Erläuterung aller Elemente und Funktionen“ ist zum Nachschlagen bei Problemen gedacht und enthält zusätzliche Informationen.

Besonders ungeduldige Leser können auch direkt zum erläuterten Beispiel springen und dieses nachbauen oder modifizieren.

Um überhaupt mittels LabVIEW mit dem Spider8 zu kommunizieren, ist mindestens die Firmware Version 2.0 notwendig.

1 Anleitung zur Installation der benötigten Treiber und Tools zur Kommunikation zwischen LabVIEW und Spider8

1.1 Beschreibung der einzelnen Programmpakete

1.1.1 VI-Bibliothek

Die VI-Bibliothek enthält eine Sammlung von VIs mit deren Hilfe Programme geschrieben werden können, die auf den Messverstärker des Typs Spider8 zugreifen. Außerdem enthält die Bibliothek auch Programmbeispiele, um die wichtigsten Funktionen zu erläutern. Diese VIs funktionieren nicht mit Vorgängerversionen von LabVIEW 2010.

1.1.2 Spider32 Setup

Das Programm Spider32 Setup ermöglicht es, Kanal- und Geräteeinstellungen für den Spider8 vorzunehmen. Diese Konfigurationen lassen sich dort als *.SP8-Dateien speichern und in einem LabVIEW-Programm laden. Die LabVIEW VIs benötigen zudem zur Kommunikation die Treiber-Dateien von Spider32 Setup. Dieses Programm wird von HBM bereitgestellt und ist auf der Installations-CD des Spider8 zu finden.

1.1.3 DIAdem Connectivity Toolkit

Für die reine Funktionsfähigkeit der Spider8-VIs ist das DIAdem Connectivity Toolkit nicht notwendig. Es enthält lediglich Tools zur erweiterten Kommunikation zwischen LabVIEW und DIAdem. Durch die Verwendung dieser VIs, ist es möglich, das Erstellen eines Protokolls aus LabVIEW heraus gezielter zu steuern als es mit den ExpressVIS der Standard-Installation möglich wäre.

1.1.4 Spider8 Hilfedateien

Diese Hilfedateien erläutern wie der Treiber richtig angesprochen wird und enthalten Erklärungen zu den möglichen Fehlercodes, die von den Treiber-Dateien erzeugen werden können. Es ist keine Hilfe für die LabVIEW-VIs direkt aber die LabVIEW-VIs sprechen die Treiber-Dateien gemäß der Anleitung diesen Hilfe-Dateien an. Zudem kann hier der IDS-Code, der vom Treiber verwendet wird, in lesbare Werte übersetzt werden.

1.2 Installation

1.2.1 Inhalt des Installationsordners

- Installationsprogramm für die Bibliothek und des DIAdem Connectivity Toolkits
- Installationsprogramm für Spider32 Setup

1.2.2 Installation der VI-Bibliothek

Um die VI-Bibliothek, die Spider8 Hilfedateien und das DIAdem Connectivity Toolkit zu installieren steht ein Installierprogramm zur Verfügung. Dieses wird durch Ausführen von Setup.exe im Verzeichnis Bibliothek gestartet. Danach folgen sie den Anweisungen auf dem Bildschirm. Als Installationsverzeichnis geben sie bitte das Verzeichnis an, in dem LabVIEW installiert wurde.

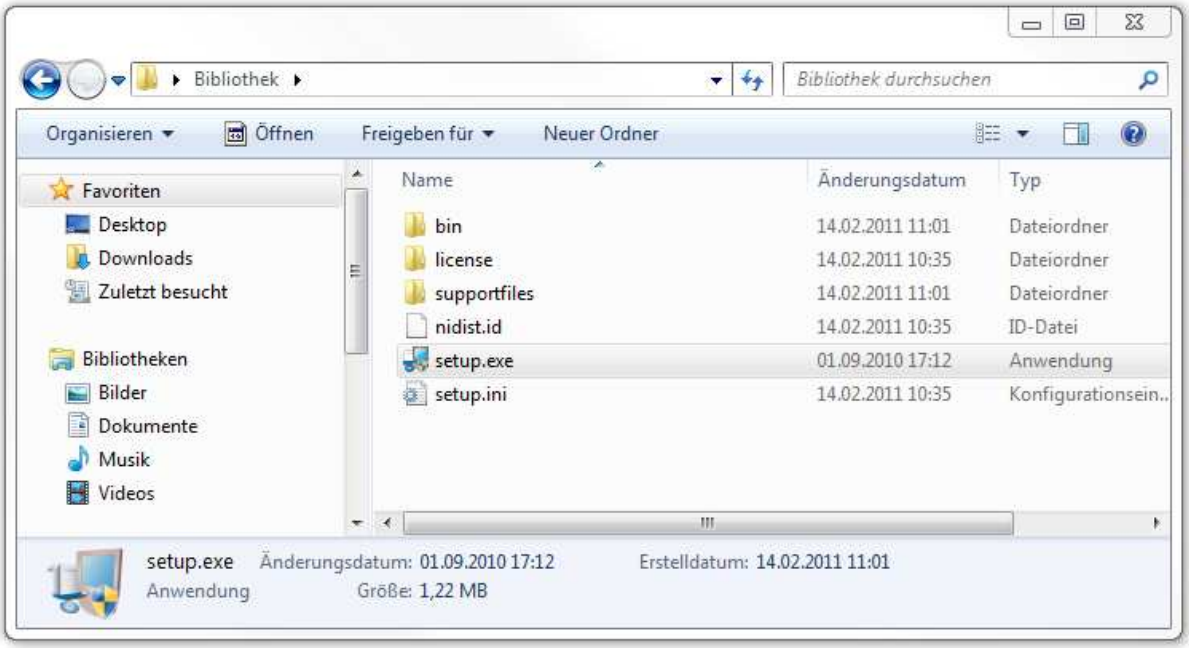


Abbildung 1: Inhalt des Bibliotheks-Verzeichnis

1.2.3 Installation von Spider32 Setup

Die Installationsdateien von Spider32 Setup befinden sich im Ordner Installationsdateien\Spider32 Setup. Zur Installation führen Sie die SETUP.EXE aus und befolgen die Anweisungen am Bildschirm.

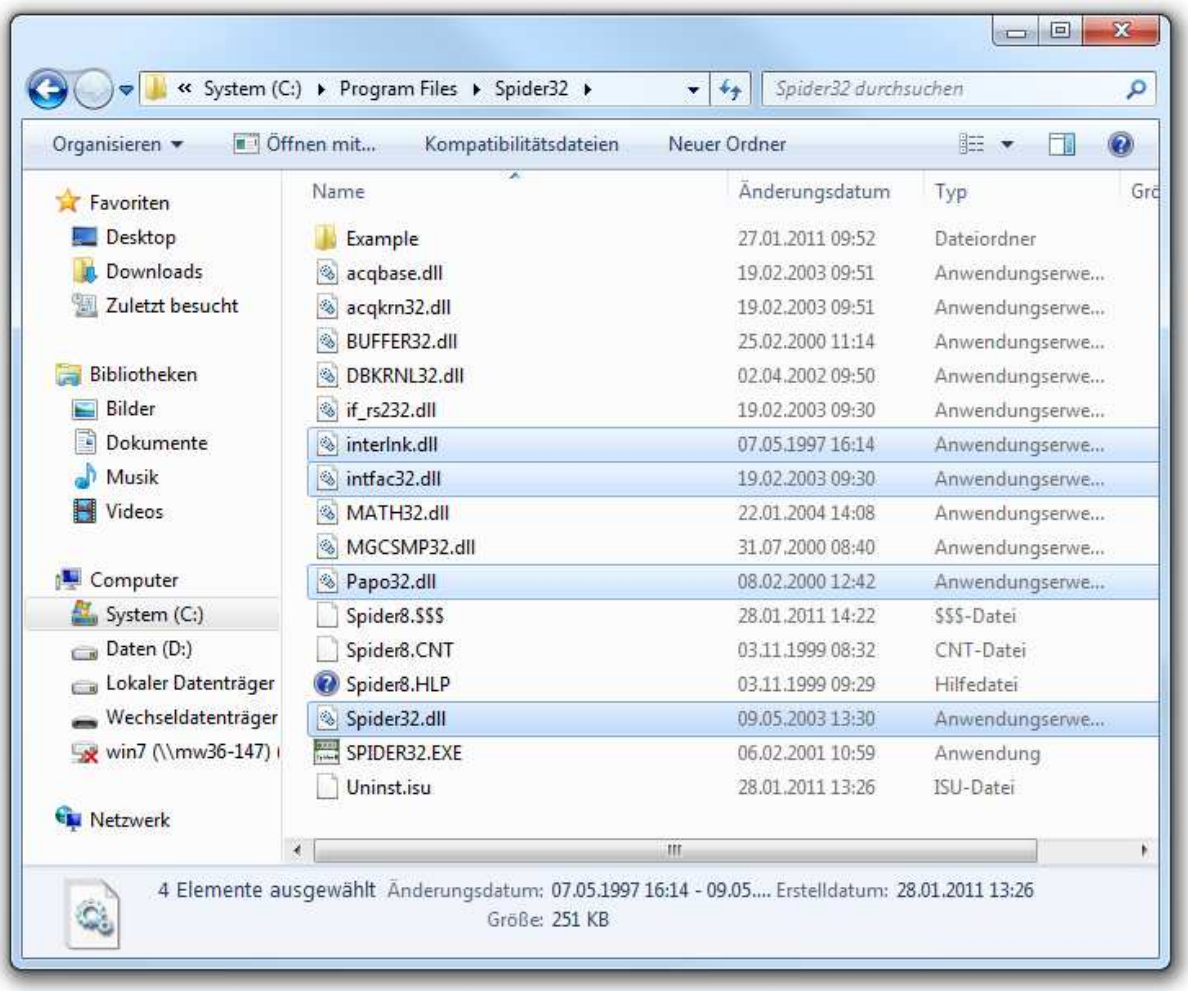


Abbildung 2: Das installierte Spider32 Setup und die markierten DLL-Dateien, die von den Bibliotheks-VIs benötigt werden

Die VIs der VI-Bibliothek erwarten die Spider32 Setup-Dateien im Ordner C:\program files\Spider32. Der Installationsordner ist prinzipiell frei wählbar, aber wenn nicht dieser Ordner gewählt wurde, sucht LabVIEW bei jedem ersten Aufrufen von jedem VI der Bibliothek nach den oben markierten Dateien. Die Suche kann beendet werden, indem Sie die aktuelle Position der Spider32.dll angeben, welche sich im Installationsverzeichnis von Spider32 Setup befindet. Wenn danach das VI gespeichert wird, ist eine erneute Eingabe nicht nötig und es startet in Zukunft keine Suche mehr.

2 Strukturvariablen

Vom Treiber werden durch bestimmte Funktionen oder VIs Strukturvariablen erzeugt, die nur im Arbeitsspeicher vorliegen. Diese Variablenstruktur enthält Messwerte, Kanaleinstellungen und Geräteeinstellungen. Diese werden im Arbeitsspeicher abgelegt, bis sie durch neue Werte ersetzt oder durch Aufrufen der Funktion S8_CloseDev.vi gelöscht werden. Dabei sind einige Werte im IDS-Code abgelegt. Diese lassen sich mit dem entsprechenden Convert VI und/oder Element in lesbare Werte übersetzten. Genaueres dazu kann der Beschreibung der VIs und Elemente entnommen werden. Die folgenden Tabellen listen alle möglichen Variablen auf und beschreiben ihren Inhalt.

Tabelle 1: Gerätestrukturvariablen

Variable	Bedeutung
NumChan	Enthält die Anzahl der Kanäle
MeasRate	Enthält die Messrate im IDS-Code
FilterT	Enthält den Filtertyp im IDS-Code
FilterF	Enthält die Filterfrequenz im IDS-Code
SerNum	Enthält die Seriennummer des Spider8
SWVers	Enthält die Versionnummer der Firmware des Spider8
Name	Enthält den Gerätenamen des Spider8
UserName	Enthält einen benutzerdefinierten Namen für den Spider8
Mode	Enthält den Operationsmodus im IDS-Code

Tabelle 2: Kanalstrukturvariablen

Variable	Bedeutung
ChanType	Enthält die Art des Kanals im IDS-Code
Name	Enthält einen benutzerdefinierten Namen für den Kanal
Unit	Enthält die Einheit in der gemessen wird
IsActive	Enthält den Aktivierungsstatus des Kanals
Mode	Enthält die Art des Sensors (Halbbrücke, etc.) im IDS-Code
MRange	Enthält den Messbereich im IDS-Code
Filter	Enthält die Filterauswahl im IDS-Code
NennVal	Enthält Nennwert des Kanals
MeasValue	Enthält den zuletzt gemessenen Messwert (nicht bei Kanal 8)
TareValue	Enthält den Tarawert
DigIO	Enthält den Wert der I/O-Buchse (nur bei Kanal 8)

3 Grundsätzliches Schema

Grundsätzlich benötigt der Treiber die in Abbildung 3 dargestellte Sequenz um wie vorgesehen zu funktionieren.

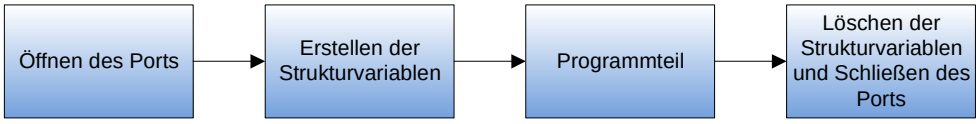


Abbildung 3: Die benötigte Treibersequenz

Dabei werden die Schritte „Öffnen des Ports“ und „Erstellen der Strukturvariablen“ in dieser Anleitung zu „Initialisieren des Spider8“ zusammengefasst. Danach folgt ein Mess- bzw. Steuerungsteil und eine Herunterfahrsequenz. Alle Funktionen und Sequenzen dieser Bibliothek verfügen über einen Fehlereingang und einen Fehlerausgang. Daher können sie über den Fehlercluster miteinander verbunden werden, um so den Programmablauf zu definieren.

3.1 Initialisieren des Spider8

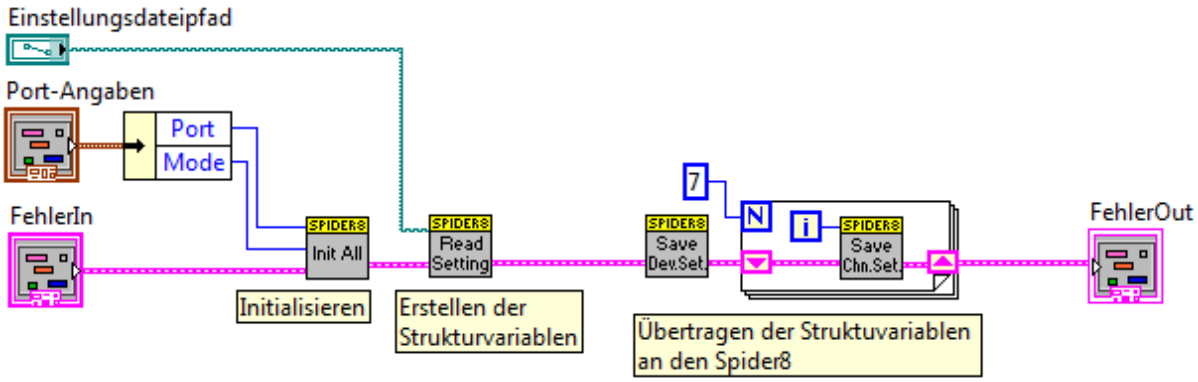


Abbildung 4: Blockdiagramm von S8_InitSequenz.vi

Die Initialisationssequenz muss folgende Aufgaben in dieser Reihenfolge erfüllen: Öffnen und Initialisieren des richtigen Ports, Laden der richtigen Geräte- und Kanaleinstellungen als Strukturvariablen in den Arbeitsspeicher und das Senden der Geräte- und Kanaleinstellungen zum Spider8. Diese Sequenz wird am Beispiel S8_InitSequenz.vi erläutert, welche sich in der VI-Bibliothek befindet und deren Blockdiagramm in Abbildung 3 dargestellt ist.

Die Funktion S8_InitAll.vi öffnet je nach Eingaben den richtigen Port im richtigen Modus. Welcher Port bzw. Modus richtig ist, hängt von der Art des Anschlusses ab und kann im technischen Datenblatt in der Bedienungsanleitung des Spider8 nachgelesen werden. Der Port ist hierbei der Anschluss an den PC und der Modus gibt die Art der Kommunikation bzw. die Baudrate an. Die Werte werden über ein Enum-Element an den Treiber weitergereicht, welches über das Kontextmenü leicht erstellt werden kann. Dazu geht man mit dem Mauszeiger über den Anschluss, für den man ein Enum-Element definieren möchte, und macht einen Rechtsklick. Der Anschluss sollte dabei blinken. Danach wählt man im Kontext-Menü „erstellen“ und dort was man erstellen will -z. B. eine Konstante- aus. Auch wenn die Angabe einer Baudrate bei Kommunikation über USB nicht notwendig ist, muss dennoch der Modus USB gewählt werden. Vorsicht: Obwohl diese Enum-Elemente gleich aussehen können, sind es unterschiedlich definierte Elemente. Wenn beide Anschlüsse mit demselben Element verbunden werden, kann die Funktion nicht richtig ausgeführt werden.

Die Geräte- bzw. Kanaleinstellungen lassen sich auf drei Arten in die Strukturvariablen im Arbeitsspeicher laden.

- Über eine Datei, die die gewünschten Einstellungen enthält
- Über Cluster mit den richtigen Einstellungen
- Durchs Herunterladen der aktuellen Einstellungen vom Gerät.

Die Verwendung einer Datei ist hierbei aber der komfortabelste Weg.

Um die Einstellungen über eine Datei zu laden, muss zunächst eine solche Datei erstellt werden. Dies geschieht mit dem Programm Spider32 Setup, welches von HBM erstellt wurde. Dort können nun die Geräte- bzw. Kanaleinstellungen vorgenommen werden. Danach können die Einstellungen über Datei/Speichern abgespeichert werden. Hier stehen zwei Formate zur Wahl: eine Binärdatei mit der Endung *.SP8 und eine ASCII-Datei mit der Endung *.txt. Die ASCII-Datei ist eine Textdatei, die alle Einstellung in englischer Sprache auflistet. Das ist zwar für den Benutzer gut verständlich, aber der Treiber benötigt die Binärdatei, um die Einstellungen richtig zu lesen.

Um nun in LabVIEW diese Einstellungen in den Arbeitsspeicher zu laden, wird, die Funktion S8_ReadSettings.vi ausgeführt. Dieses VI benötigt den Dateipfad zur der *.SP8-Datei mit den gewünschten Einstellungen.

Alternativ können die Einstellungen auch in Form von mehreren Clustern im LabVIEW Programm vorhanden sein. Dabei sind ein Cluster je Kanal und ein Cluster für die Geräteeinstellungen notwendig, welche dann einzeln über S8_SetChanSetting.vi bzw. S8_SetDevSettings.vi in den Arbeitsspeicher geladen werden. Die Cluster sind bereits vordefiniert in der Element-Bibliothek unter den Namen S8_ChanType.ctl bzw. S8_DevType.ctl vorhanden und lassen sich alternativ über das Kontextmenü erstellen. Nachteil dieser Methode ist, dass hier ungültige Einstellungskombinationen gewählt werden können, welche zu Fehlermeldungen in anderen VIs oder falschen Messwerten führen können. Außerdem kann mit S8_SetDevSettings.vi immer nur eine Geräteeinstellung in den Arbeitsspeicher geladen werden. Genaueres dazu steht in der VI-Beschreibung.

In beiden Fällen wird nicht mir dem Spider8 kommuniziert, sondern lediglich die Einstellungen als Strukturvariablen in den Arbeitsspeicher des PCs geschrieben, womit alle weiteren VIs darauf zugreifen können.

Um die Einstellungen an den Spider8 zu übertragen, sind die VIs S8_SaveChanSettings.vi bzw. S8_SaveDevSettings.vi notwendig. Die Einstellungen werden dabei aus dem Arbeitsspeicher geladen, womit eine weitere Eingabe nicht notwendig ist. Für S8_SaveChanSettings.vi ist allerdings eine Angabe des Kanals notwendig, für den neue Einstellungen übertragen werden sollen. Das VI muss also für jeden Kanal einzeln ausgeführt werden. Der Kanal 8 (I/O-Buchse) bildet hier eine Ausnahme, denn dieser muss nicht konfiguriert werden. Das VI S8_InitSequenz.vi speichert die Geräteeinstellungen und die Kanaleinstellungen für Kanal 0 und Kanal 1 in dieser Reihenfolge ab.

Danach ist der Spider8 vollständig initialisiert.

3.2 Messen oder Steuern

Um die Messwerte aus den Kanälen auszulesen gibt es drei verschiedene Alternativen. Die erste Möglichkeit ist das Messen eines Wertes je Kanal, was für eine kontinuierliche Messung mit z. B. einer Schleife mehrfach wiederholt werden kann. Die zweite Variante ist eine nichtkontinuierliche Messung, bei der eine festgelegte Anzahl von Messwerten mit einer bestimmten Messrate gemessen und angegeben wird. Die letzte Möglichkeit ist eine kontinuierliche Messung, bei der die Messergebnisse in Datenpaketen zu maximal 3000 Messwerten je Kanal vom Spider8 gelesen werden. Der Unterschied der zweiten und dritten Variante zur ersten Variante liegt darin, dass der Spider8 die Messrate steuert und nicht das LabVIEW-Programm, was sehr viel schnellere Messraten ermöglicht.

Bei jeder dieser drei Methoden ist es notwendig, dass die Kanäle vorher über S8_ACQSetup.vi zum Messen aktiviert werden. Dies erfordert einen 1D-Array mit numerischen Werten im long Integer Format, welcher die gewünschten Kanalnummern enthält und an das VI übergeben wird.

3.2.1 Messung eines Wertes

Diese Art der Messung ist in S8_Einzelmessung.vi realisiert worden und wird anhand dieses Beispiels erläutert. Das Blockdiagramm ist in Abbildung 4 dargestellt. Dabei wird ein Wert für jeden Kanal mit der Funktion S8_MeasOneVal.vi gemessen und in die Strukturvariablen im Arbeitsspeicher geladen.

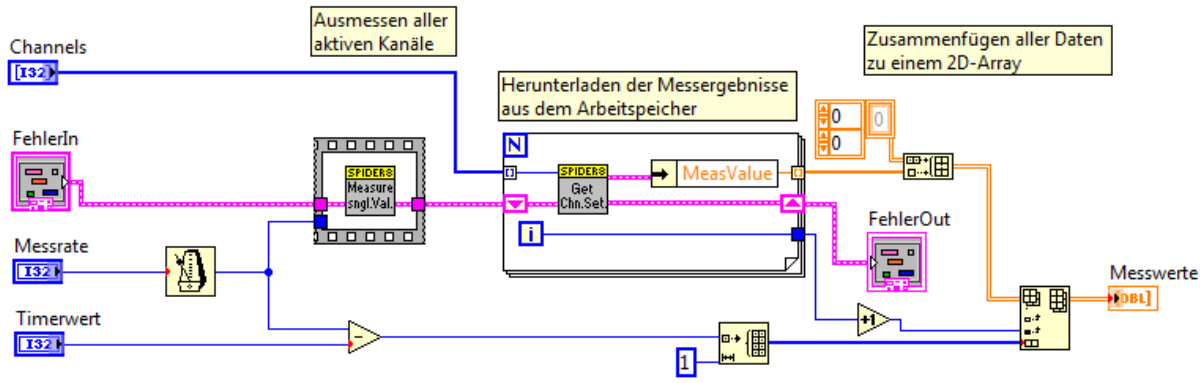


Abbildung 5: Das Blockdiagramm von S8_Einzelmessung.vi

In S8_Einzelmessung.vi wird vor der Messung bis zu einem Vielfachen der angegebenen Messrate gewartet, um im Falle einer kontinuierlichen Messung eine bestimmte Messrate einhalten zu können. Der Timerwert nach dem Warten wird daraufhin weitergegeben.

Die gemessenen Werte können daraufhin mit S8_GetChanSettings.vi gelesen werden. Allerdings muss das Auslesen für jeden Kanal einzeln erfolgen, was in S8_Einzelmessung.vi durch eine Schleife durchgeführt wird. Die Funktion gibt daraufhin einen Cluster mit dem vorher gemessenen Wert und den aktuellen Kanaleinstellungen aus, welcher noch aufgeschlüsselt werden muss. Die Messwerte werden dann zusammen mit den Timerwerten in ein 2D-Array geschrieben, bei dem jede Spalte den Wert eines Kanals enthält und die letzte Spalte den Timerwert enthalten.

Diese Methode zum Messen ist vorwiegend für Steueraufgaben geeignet, da hier jeder Messwert vor dem Messen des nächsten Werts weiterverarbeitet werden kann. Allerdings ist hier die maximale

Messrate durch aktuelle Prozessorlast und maximaler Kommunikationsgeschwindigkeit mit dem Spider8 begrenzt. Bei zu schnellen Messraten wird die Einhaltung der aktuellen Messrate zunehmend ungenauer, da es vorkommen kann, dass Durchgänge übersprungen werden.

3.2.2 Nichtkontinuierliche Messung

Hier wird eine festgelegte Anzahl Messwerte mit einer bestimmten Messrate gemessen und als ein 2D-Array herausgegeben. Das ist in S8_Messung.vi realisiert. In diesem VI ist kein Trigger definiert worden. Die Messung beginnt unmittelbar bei Ausführung. Das Blockdiagramm ist in Abbildung 5 dargestellt.

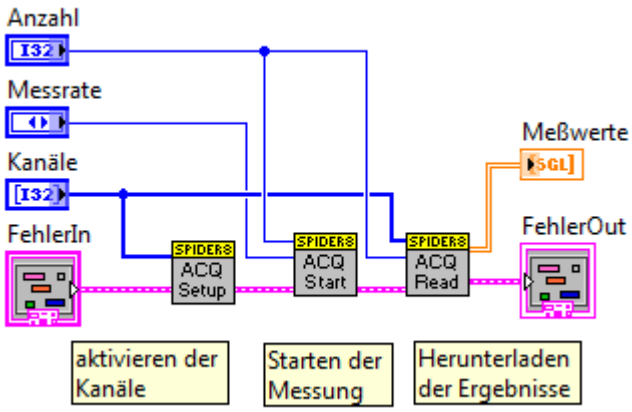


Abbildung 6: Blockdiagramm von S8_Messung.vi

Dazu muss die Messung mit S8_ACQStart.vi gestartet werden, was die Angabe einer Anzahl von Messwerten benötigt. Zusätzlich lassen sich Triggerbedingungen und eine Anzahl an Messwerten, die vor dem Auslösen des Triggers aufgenommen werden, definieren. Näheres dazu steht in der VI-Beschreibung. Alle Messwerte werden daraufhin auf dem Spider8 gespeichert. Vorsicht: Es werden alle aktiven Kanäle ausgelesen. Wenn mehr Kanäle aktiv sind als über den Anschluss „Channels“ übergeben werden, ist das Messwerte-Array am Ende ungeordnet und damit nur umständlich auswertbar.

Um sie nun vom Gerät herunterzuladen, wird mit S8_ACQRead.vi eine bestimmt Anzahl an Messwerten (maximal 3.000) vom Spider8 herausgelesen. Dazu wird ebenfalls ein Array mit den aktiven Kanälen benötigt. Genaueres kann der VI-Beschreibung entnommen werden.

Diese Methode ermöglich die maximale Messrate von 9600Hz, ist aber durch die maximale Anzahl an Messwerten von 3.000 und die fehlende Abbrechen-Option nur für sehr spezielle Aufgaben geeignet.

Im VI S8_Messung.vi werden die Kanäle mit aktiviert, sodass es beim Auslesen nicht zu einem ungeordneten Array kommen kann.

3.2.3 Kontinuierliche Messung über Datenpakete

Diese Variante ähnelt sehr der nichtkontinuierlichen Messung, weil die gleichen Funktionen verwenden werden, und ist als S8_kon_Messung.vi umgesetzt worden. Dieses VI lässt sich allerdings so, wie es in der Bibliothek vorliegt, nicht ausführen, denn es fehlt eine Abbruchbedingung der Schleife. Wenn das VI als SubVI arbeiten soll, ist die Ergänzung einer globalen Variablen nötig. Alternativ kann das VI auch als Kopiervorlage dienen.

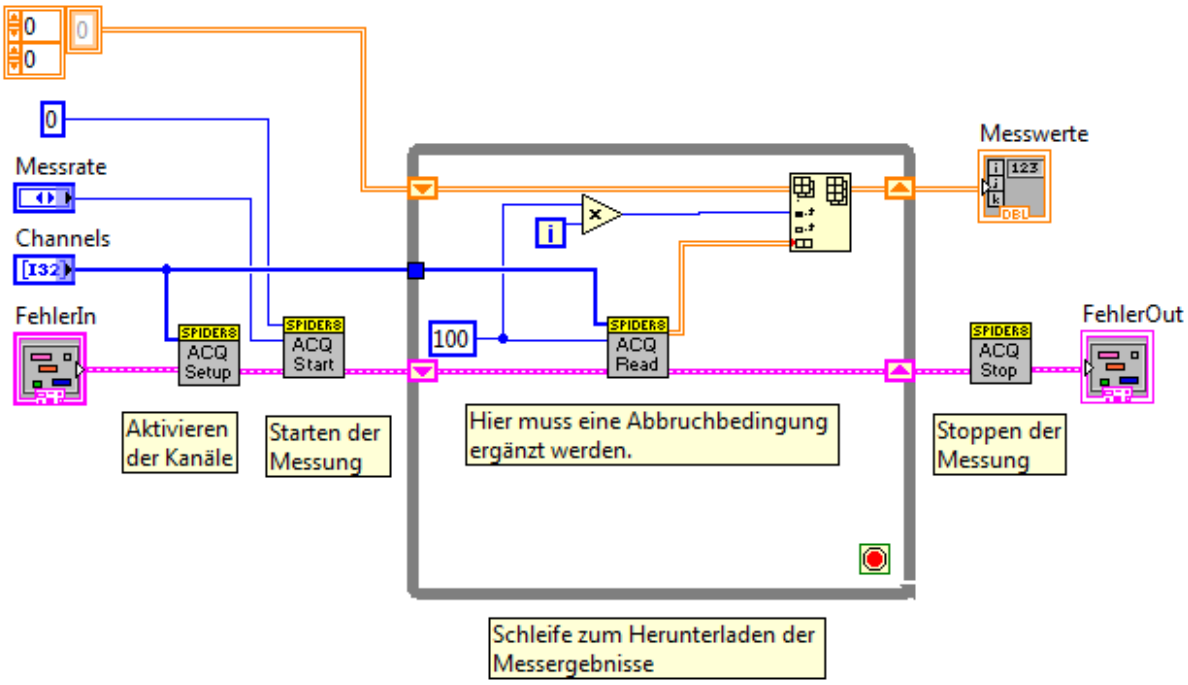


Abbildung 7: Das Blockdiagramm von S8_kon_Messung.vi

Der Beginn ist ganz ähnlich wie bei S8_Messung.vi. Allerdings wird beim Starten der Messung über S8_ACQStart.vi für die Anzahl an Messerwerte eine Null angegeben. Daraufhin misst der Spider8 solange, bis er über S8_ACQStop.vi abgebrochen wird.

Während der Messung wird nun das VI S8_ACQRead.vi mehrfach ausgeführt, da das VI die Daten nach dem FIFO-Prinzip herausliest und bereits herausgelesene Daten vom Spider8 gelöscht werden. Dies geschieht am besten über eine Schleife. Die Datenpakete müssen dann zusammengeführt werden, indem man das neue Datenpaket an die Ergebnisse der letzten Schleife anhängt.

Mit dieser Methode kann ebenfalls eine Messrate von 9600Hz erreicht werden, wobei hier die Anzahl der Messwerte prinzipiell nicht begrenzt ist, solange der Speicher des Spider8 nicht voll ist. Wenn aber eine längere Messung mit hoher Messrate durchgeführt werden soll, kann es passieren, dass die Daten nicht schnell genug herausgelesen werden. Ein weiterer Nachteil ist, dass diese Variante nicht ohne weiteres als SubVI vorbereitet werden kann, weil hierfür globale Variablen definiert werden müssten. Diese Methode muss also immer angepasst werden.

In S8_kon_Messung.vi werden die Kanäle automatisch aktiviert und kein Trigger definiert. Wie man einen Trigger definiert, kann der VI-Beschreibung des VIs S8_ACQStart.vi entnommen werden.

3.2.4 Lesen von I/O-Signalen

Der Spider8 verfügt zusätzlich zu den Messkanälen eine I/O-Buchse mit acht Inputs und weiteren acht Input/Outputs. Diese max. 16 möglichen Inputs können mit S8_MeasOneVal.vi ausgelesen werden, was in S8_IO_lesen.vi umgesetzt ist. Dazu muss vorher der Kanal 8, der für die I/O-Buchse steht, über S8_ACQ-Setup.vi zum Messen aktiviert werden. Dies wird nicht automatisch von S8_IO_lesen.vi durchgeführt. Das Blockdiagramm ist in Abbildung 7 dargestellt.

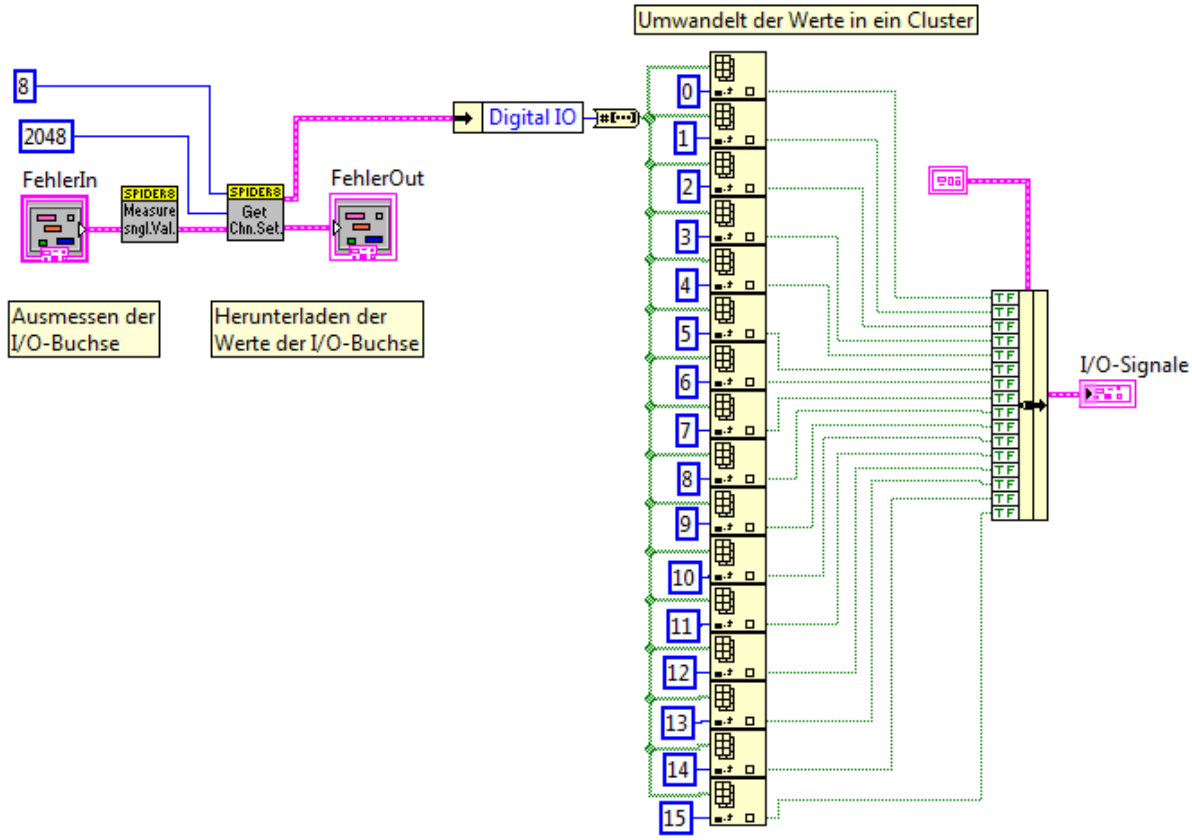


Abbildung 8: Das Messen der Kanäle in S8_IO_lesen.vi

Mit S8_GetChanSettings.vi können nun die Werte ausgelesen werden. Der Kanal 8 ist allerdings kein Messkanal und deshalb sind viele vordefinierte Werte, die für normale Messkanäle gelten, nicht vorhanden. Ein Herauslesen aller Werte über S8_GetChanSettings.vi würde zu Fehlermeldungen führen. Deswegen wird über den Eingang „What (All)“ die genaue Strukturvariable ausgewählt, die ausgelesen werden soll. Dafür muss in diesem Fall die Zahl 2048 im Long-Format eingesetzt werden. Welche Zahl für welche Strukturvariable steht, kann der VI-Beschreibung entnommen werden.

Der Cluster, der von S8_GetChanSettings.vi erzeugt wird, muss danach nach Digital IO aufgeschlüsselt werden. Der Wert entspricht einer 32bit Zahl im Long-Format. Diese Zahl muss nun über die Funktion „Zahl nach boolschen Array“ in ein boolsches Array umgewandelt werden. Die ersten acht Werte entsprechen nun den acht Input/Outputs und die Werte Nummer neun bis 16 entsprechen den acht Inputs. Die Restlichen 16 Werte haben keine Bedeutung.

In S8_IO_lesen.vi wird danach zum besseren Handling das Array in einen Cluster aus boolschen Werten umgewandelt.

3.2.5 Setzen von I/O-Signalen

Das Setzen von Signale funktioniert wie das Übertragen von Kanaleinstellungen. Erst wird der neue Wert in die Strukturvariablen im Arbeitspeicher geschrieben und danach zum Gerät übertragen. Beide Schritte werden von S8_IO_setzen.vi durchgeführt. Das Blockdiagramm ist in Abbildung 8 dargestellt.

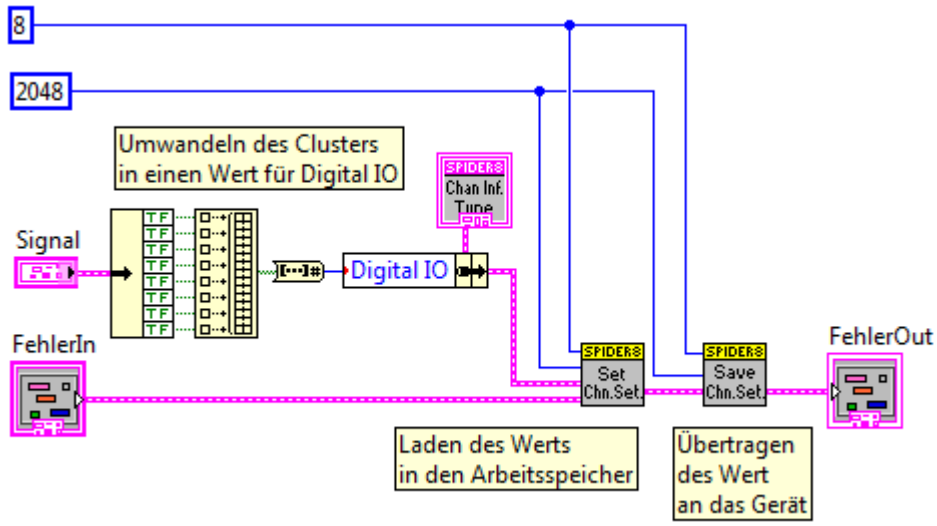


Abbildung 9: Das Blockdiagramm von S8_IO_setzen.vi

Für das Schreiben in den Arbeitsspeicher wird die Funktion S8_SetChanSettings.vi benötigt. Hier wird wieder ein Cluster hinein gegeben, der einen Platz für alle möglichen Kanaleinstellungen enthält. In diesem Cluster ist hier aber nur der Wert für Digital IO interessant, denn dort werden als 32Bit-Long Zahl die neuen Werte der Outputs gesetzt. Dabei stehen die ersten acht Bits für die Outputs und die restlichen Bits müssen mit True-Werten aufgefüllt werden, was LabVIEW automatisch beim umwandeln in den 32Bit-LongWert macht. Weil der Cluster wieder Einstellmöglichkeiten enthält, die nicht für die I/O-Buchse zutreffen, wird, wie beim Lesen der I/O-Signale, durch die Eingabe von 2048 bei „What (All)“ nur der Wert für Digital IO gesetzt.

Danach wird die Strukturvariable mit S8_SaveChanSettings.vi an das Gerät übertragen. Hier muss ebenfalls bei „What (All)“ die 2048 gesetzt werden.

3.3 Herunterfahren

Das Herunterfahren besteht aus zwei Schritten, die in beliebiger Reihenfolge durchgeführt werden können. Dies wird von S8_Herunterfahrsequenz.vi durchgeführt und das Blockdiagramm ist in Abbildung 9 dargestellt.

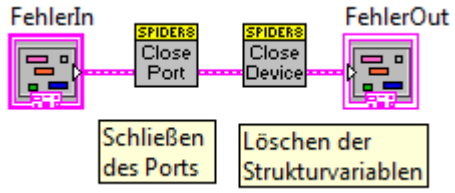


Abbildung 10: Blockdiagramm von S8_Herunterfahrsequenz.vi

Mit S8_ClosePort.vi wird die Verbindung mit dem Spider8 beendet. S8_CloseDevice.vi löscht die Strukturvariablen aus dem Arbeitspeicher. Ein sauberes Herunterfahren ist sehr wichtig für die Zuverlässigkeit der Programme. Wenn das Gerät nicht durch diese beiden VIs beendet wird, führt das zu Messfehlern und Fehlermeldungen beim Ausführen des nächsten Programms, obwohl es vielleicht gar nicht fehlerhaft ist. Diese beiden Funktionen sind die einzigen Funktionen, die auch ausgeführt werden, wenn am Fehlereingang ein Fehler vorliegt.

4 Erläuterung eines Programmbeispiels

Mit der nun vorliegenden Bibliothek sind mit wenigen Handgriffen und Programmierkenntnissen sowie grundlegenden Kenntnissen der Messtechnik Messprogramme erstellbar. Um dies zu verdeutlichen, werden hier ein simples Beispiel eines Messprogramms und sein Aufbau detailliert erläutert. Das „Messungsbeispiel.vi“ genannte Programm befindet sich ebenfalls in der VI-Bibliothek und kann von dort aus geladen und bei Bedarf modifiziert werden.

Generell wird die Sequenz durch den Fehlercluster gesteuert, der sich durch alle verwendeten Knoten zieht und im Blockdiagramm violett dargestellt ist. Es arbeitet also von links nach rechts die Knoten ab. Am linken Rand des Blockdiagramms werden alle benötigten Randbedingungen aufgelistet. Dabei handelt es sich um die Anzahl der Messwerte, die Kanäle, die ausgelesen werden sollen, der Pfad zu einer Einstellungsdatei mit Geräte und Kanaleinstellungen und einer Cluster-Konstanten, die die nötigen Angaben für eine Verbindung über USB enthält. Messwerteanzahl, Kanäle und Einstellungsdateipfad werden dabei über ein Bedienelement im Frontpanel eingegeben.

Mit der Einstellungsdatei, die z. B. Messrate, Sensortyp sowie Filtereinstellungen enthält, und den Verbindungsangaben öffnet S8_InitSequenz den Port und erstellt passende Strukturvariablen, die danach an das Gerät übertragen werden. S8_ACQSetup aktiviert alle im Kanäle-Array angegebenen Kanäle zum Messen. S8_Messung startet eine Messung mit der angegebenen Anzahl von Messwerten und lädt diese herunter. Danach wird die Messung automatisch beendet. Diese Messergebnisse werden in der Einheit mit der der Spider8 das Signal des Sensors empfängt, in einem 2D-Array ausgegeben und danach in einem Signalverlaufsdiagramm angezeigt. S8_Herunterfahrsequenz beendet die Verbindung und löscht die Strukturvariablen aus dem Arbeitsspeicher. Am Ende ist noch S8_Fehlerhandler ergänzt, das aus den Daten des Fehlercluster eine Fehlermeldung erstellt, wenn irgendwo ein Fehler aufgetreten ist. Danach ist jeder Datenfluss an einer Datensenke angekommen und alle Knoten wurden abgearbeitet, wodurch das Programm beendet wird. Das Frontpanal ist in Abbildung 10 und das Blockdiagramm in Abbildung 11 dargestellt.

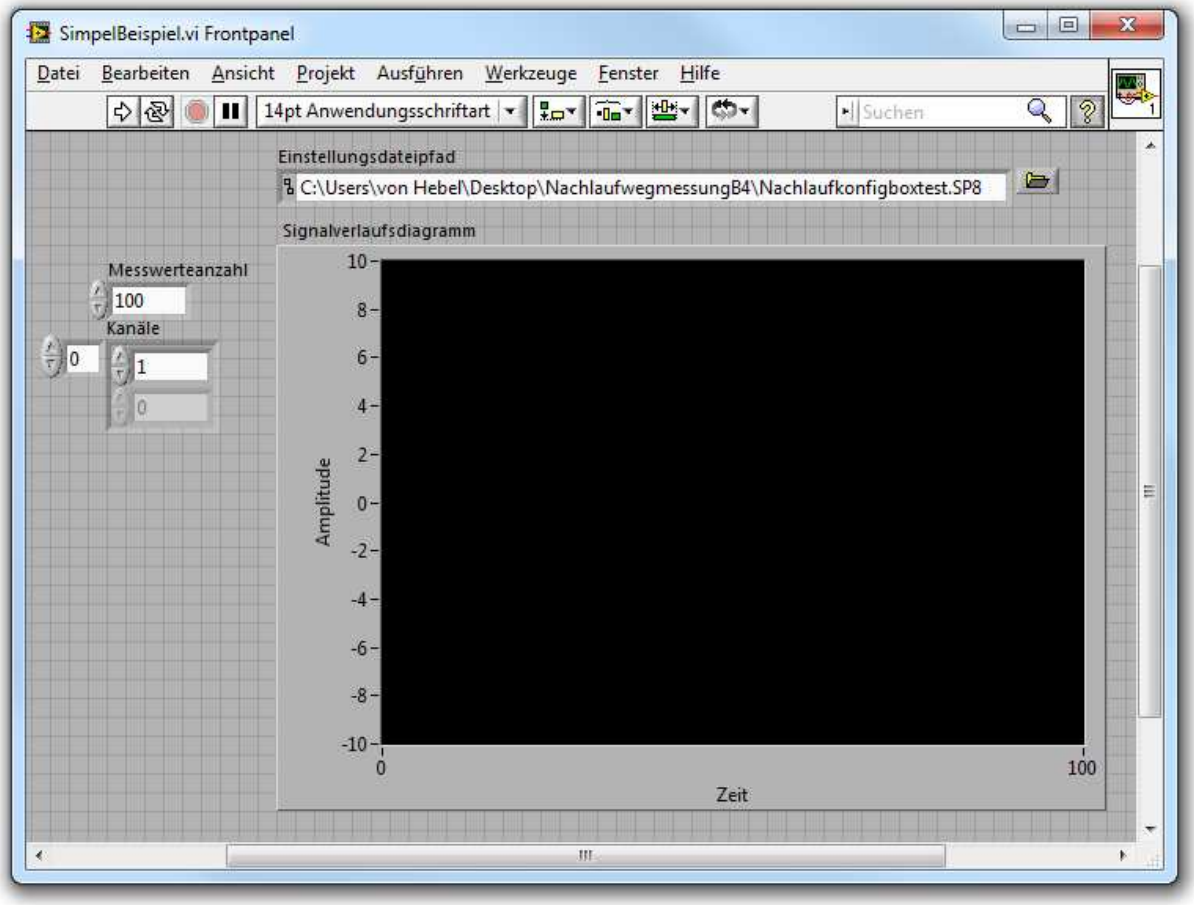


Abbildung 11:Frontpanel des vorgestellten Beispiels

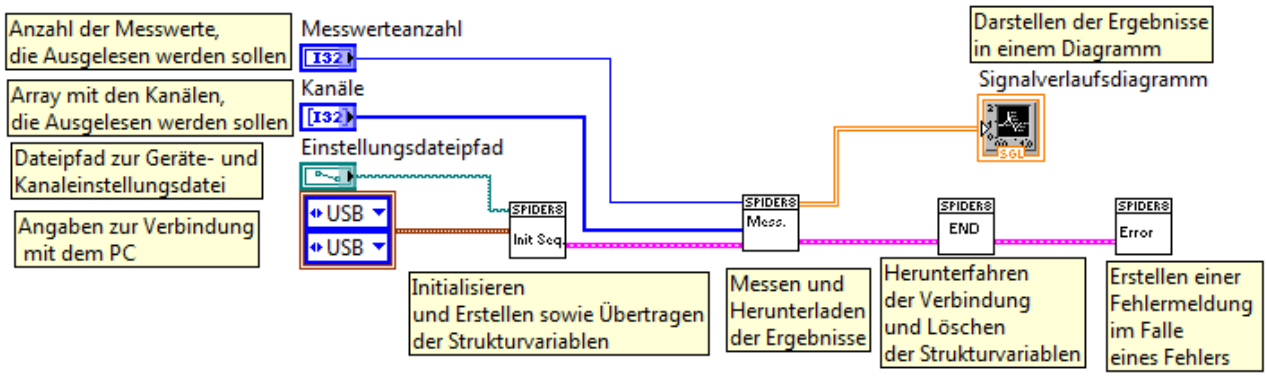


Abbildung 12: Blockdiagramm des vorgestellten Beispiels

5 Erläuterung aller Elemente und Funktionen

5.1 Elemente

In der Bibliothek sind diverse Elemente vordefiniert, sodass diese eine schnelle Erstellung von Konstanten, Anzeige- oder Bedienelementen ermöglichen.

ChanActive.ctl

Vordefiniertes Enum-Element für Werte der Strukturvariable IsActive.

ChanType.ctl

Vordefiniertes Enum-Element für Werte der Strukturvariable ChanType.

DevMode.ctl

Vordefiniertes Enum-Element für Werte der Strukturvariable Mode aus den Geräteeinstellungen.

FilterChar.ctl

Vordefiniertes Enum-Element für Werte der Strukturvariable FilterT.

FilterFreq.ctl

Vordefiniertes Enum-Element für Werte der Strukturvariable FilterF.

FilterSelect.ctl

Vordefiniertes Enum-Element für Werte der Strukturvariable Filter.

MeasRate.ctl

Vordefiniertes Enum-Element für Werte der Strukturvariable MeasRate.

RangeType.ctl

Vordefiniertes Enum-Element für Werte der Strukturvariable MRange.

S8_ChanType.ctl

Vordefinierter Cluster aus Enum, String- und numerischen Elementen für alle Strukturvariablen, die Kanaleinstellungen enthalten.

S8_DevType.ctl

Vordefinierter Cluster aus Enum, String- und numerischen Elementen für alle Strukturvariablen, die Geräteeinstellungen enthalten.

SensorType.ctl

Vordefiniertes Enum-Element für Werte der Strukturvariable Mode aus den Kanaleinstellungen.

TriggerEdge.ctl

Vordefiniertes Enum-Element für die Parameter TrgEdge der Funktion S8_ACQStart.vi.

TriggerMode.ctl

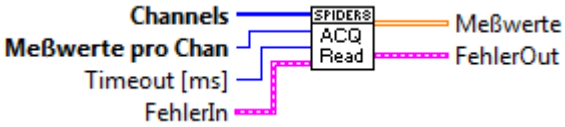
Vordefiniertes Enum-Element für die Parameter TrgMode der Funktion S8_ACQStart.vi.

5.2 Funktionen

Eigentlich sind alle Funktionen in der Spider8 Bibliothek VIs. Trotzdem wird hier zwischen Funktionen, die auf eine einzige Treiberfunktion zugreifen, Sequenzen, die mehrere Treiberfunktionen enthalten, und Convert-VI, die zur Übersetzung des IDS-Codes dienen, unterschieden

Die hier aufgelisteten Funktionen rufen eine Treiberfunktion auf und setzen die Parameter sowie geben die Ergebnisse in einer für LabVIEW verarbeitbaren Form aus. Grundsätzlich gibt jede Funktion in jedem Fall einen Rückgabewert aus, der den Fehlercode enthält. Dieser Wert beträgt 0, wenn kein Fehler aufgetreten ist. Das macht sich die Bibliothek zunutze, indem damit und mit dem Namen der Funktion ein Fehlercluster erstellt wird, der am Fehlerausgang ausgegeben wird. Zusätzlich verfügt jede Funktion und jede Sequenz über einen Fehlereingang. Wenn dort ein Fehlercluster, der einen Fehler enthält, anliegt, wird die Ausführung des VIs verhindert und der anliegende Fehler wird am Fehlerausgang weitergegeben. So können VIs durch den Fehlercluster gesteuert werden.

S8_ACQRead.vi



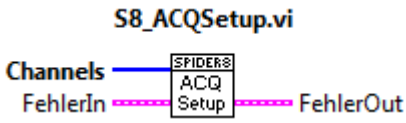
Dieses VI lädt die gewünschte Anzahl von Messwerten aus einer beliebigen Anzahl an Kanälen, löscht die Werte aus dem Speicher des Spider8 und gibt sie in einem 2D-Array heraus. Die Kanäle werden als 1D-Array über den Anschluss „Channels“ und die Anzahl der Messwerte als numerischer Wert im Long-Format über „Messwerte pro Chan“ an das VI übergeben

Bei einer Messung werden die zuerst gemessenen Werte im Speicher auch zuerst ausgelesen, sodass bei einer kontinuierlichen Messung die Werte ohne Laufzeit-Probleme ausgelesen werden können. Eine Messung wird dabei nicht unterbrochen und mit einem erneuten Ausführen von S8_ACQRead.vi kann das nächste Datenpaket aus Messwerten herunter geladen werden. Ein Datenpaket kann dabei max. 3000 Messwerte pro Kanal enthalten.

Wenn nicht genug Werte vorliegen, wartet die Funktion bis genügend Messwerte aufgenommen worden sind. Daher muss ein Timeout definiert werden, der die Funktion beendet, wenn die Funktion zulange auf Werte wartet. Dies ist ein numerischer Wert im Long-Format, der in Millisekunden am Anschluss „Timeout (ms)“ übergeben wird.

Es muss vorher eine Messung mit S8_ACQStart.vi gestartet werden. Der Treiber gibt alle Messwerte als 1D-Array aus, welches über LabVIEW-Array-Funktionen in ein passendes 2D-Array aus numerischen Daten im Double-Format umgewandelt und am Anschluss „Messwerte“ übergeben wird. Da alle aktiven Kanäle ausgemessen werden, muss ein Array mit allen aktiven Kanälen übergeben werden. Sonst ist die Umwandlung fehlerhaft und die Werte innerhalb des Messwerte-sind Arrays ungeordnet.

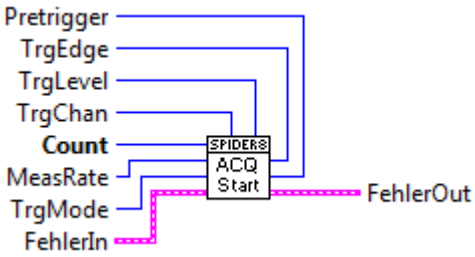
S8_ACQSetup.vi



Diese Funktion aktiviert eine beliebige Anzahl an Kanälen fürs Messen. Deren Nummern werden über ein 1D-Array aus numerischen Werten im Long-Format am Anschluss „Channels“ dem VI übergeben.

Es sollten vorher die korrekten Kanaleinstellungen über S8_SaveChanSettings.vi an das Gerät übertragen werden, da sonst die letzten verwendeten Werte bzw. die Standardwerte des Spider8 verwendet werden. Wenn die nicht stimmen, kann es in dieser oder in anderen Funktionen zu Fehlermeldungen oder Messfehlern kommen.

S8_ACQStart.vi



Damit wird eine Messung mit einer bestimmten und für alle Kanäle gleichen Messrate gestartet. Diese wird über den Anschluss „MeasRate“ an das VI übergeben. Die Eingabe sollte über das Enum-Element MeasRate.ctl erfolgen, da der Treiber diese Angabe im IDS-Code benötigt und das Element zusammen mit Convert MeasRate to Value den richtigen IDS-Code erzeugt. Welcher IDS-Code für welche Messrate steht, kann der Hilfe entnommen werden.

Es ist möglich eine vorher definierte Anzahl an Messwerten zu messen, wodurch die Messung im Anschluss automatisch beendet wird. Die Anzahl wird über den Anschluss „Count“ als numerische Daten im Long-Format eingegeben. Die maximale Anzahl an Messwerten beträgt dabei 20.000 Werte.

Alternativ kann auch eine kontinuierliche Messung begonnen werden, die über S8_ACQStop.vi beendet werden muss und die ein- oder mehrmals mit S8_ACQRead.vi ausgelesen wird, da sonst der Speicher im Spider8 voll geschrieben wird wodurch keine weiteren Messungen möglich sind. Dazu wird als Messwerteanzahl „0“ angegeben.

Zusätzlich lässt sich ein Trigger definieren, bei dessen Auslösen die Messung gestartet wird. Das Starten der Messung geschieht intern im Spider8 und wird nicht ohne weiteres an LabVIEW weiter gegeben. Dabei gibt es vier Möglichkeiten für einen Trigger. Die Auswahl wird über das Element TriggerMode.ctl an den Anschluss TrgMode übergeben.

Die erste Möglichkeit ist das Auslösen des Triggers über eine Funktion im VI. Dazu wird der Triggermodus auf TRG Cmd gestellt. Wenn nun die Funktion S8_Trigger.vi ausgeführt wird, beginnt die Messung.

Die Zweite Möglichkeit ist die Verwendung eines externen Triggers. Hier wird der TriggerMode „Extern“ gewählt. Wenn nun der Pin 14 mit der Masse der I/O-Buchse verbunden wird, startet die Messung.

Der TriggerMode „Intern“ vergleicht den aktuellen Wert eines Triggerkanals mit einem Pegel. Erreicht der Triggerkanal den Pegel, wird die Messung gestartet. Um zu verhindern, dass eine Messung zu früh beginnt, kann man auch eine Richtung definieren, aus der der Wert kommen muss, z. B. muss der Wert auf den Pegel ansteigen um die Messung zu starten. Der Kanal wird unter dem Anschluss „TrgChan“ als numerischer Wert im Long-Format, der Pegel als numerischer Wert im Long-Format an „TrgLevel“ und das Triggerverhalten über das Element TriggerEdge.ctl an „TrgEdge“ übergegeben. Die Optionen „MW > Triggerlevel“ und „MW < Triggerlevel“ starten dabei die Messung bei einem Messwert von größer bzw. kleiner gleich dem Pegel und die Optionen „positive Flanke“ und „negative Flanke“ starten die Messung, wenn der Wert bis zum Pegel angestiegen bzw. gefallen ist.

Zusätzlich können max. 200 Werte vor dem Auslösen des Triggers mit aufgenommen werden indem die gewünschte Anzahl an Werten beim Anschluss „Pretrigger“ angegeben wird.

S8_ACQStop.vi

Dieses VI stoppt eine kontinuierliche Messung und löscht alle auf dem Spider8 befindlichen Messwerte. Alle nicht über S8_ACQRead.vi aufgenommenen Werte sind danach verloren.

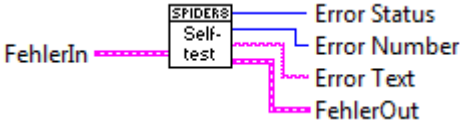
S8_CloseDevice.vi

Dieses VI löscht alle Strukturvariablen, die sich noch im Arbeitsspeicher befinden.

S8_ClosePort

Der aktuell offene Port wird geschlossen.

S8_DoSelfTest.vi



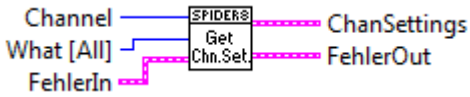
Hiermit startet das Gerät eine Selbstdiagnose und die Ergebnisse werden in Form eines Fehlerstatus, Fehlercodes und eines Fehlerstrings ausgegeben.

S8_DoTara.vi

Dieses VI führt eine Tarierung durch. Dazu müssen zuerst die zu tarierenden Kanäle mittels S8_TareSetup.vi zum Tarieren aktiviert werden.

Bei einer Tarierung wird der aktuell anliegende Wert als Tarawert in die Kanaleinstellungen eingetragen und dieser Wert wird von jedem gemessenen Wert abgezogen. Es wird also praktisch der aktuell anliegende Wert zum Nullpunkt des Sensors gesetzt.

S8_GetChanSettings.vi



Diese Funktion liest die aktuellen Werte der Strukturvariablen aus, die die Kanaleinstellungen für einen bestimmten Kanal enthalten. Der Kanalnummer wird als numerischer Wert im Long-Format am Anschluss „Channel“ übergeben.

Es können alle oder nur bestimmte Werte ausgelesen werden. Wenn nur ein bestimmter Wert benötigt wird, kann unter dem Anschluss „What (All)“ als numerischer Wert im Long-Format der Wert ausgewählt werden, der der Strukturvariable zugeordnet ist. Welcher Wert für welche Variable steht kann folgender Tabelle entnommen werden. Wenn mehr als ein Wert ausgelesen werden soll, können die Werte für What (All) addiert werden, die dem zu ändernden Werten entsprechen. Soll z.B. Der Kanalname und der Tarawert geändert werden ist 520 -8 für den Namen und 512 für den Tarawert- anzugeben.

Tabelle 3: What (All)-Werte für S8_GetChanSettings.vi

What (All)	Strukturvariable	What (All)	Strukturvariable
1	NennVal	64	MRange
2	ChanType	128	Unit
4	SerNum	256	MeasValue
8	Name	512	TareValue
16	IsActive	1024	Filter
32	Mode	2048	DigIO

Die Werte für „MeasValue“ und „DigIO“ müssen vorher mit S8_MeasOneVal.vi gemessen werden. Außerdem muss für den Wert „DigIO“ immer 2048 am Anschluss „What (All)“ gesetzt werden. Dies ist der einzige Weg die I/O-Buchse auszulesen, deren Werte als numerische Daten im Long-Format abgerufen werden. Dabei stehen die ersten 16Bit für die Inputs bzw. Outputs.

Die Kanaleinstellungen werden als ein Cluster aus allen möglichen Strukturvariablen herausgegeben, wobei Werte, die nicht abgefragt werden, leer bleiben. Daher muss der Cluster nach den benötigten Werten aufgeschlüsselt werden. Es findet keine Kommunikation mit dem Gerät statt, sondern nur mit dem Arbeitsspeicher.

S8_GetDevSettings.vi

Diese Funktion funktioniert ähnlich wie S8_GetChanSettings.vi. Es werden die aktuellen Werte aller Strukturvariablen ausgelesen, die Geräteeinstellungen enthalten. Auch hier kann durch Eingabe eines numerischen Werts im Long-Format beim Anschluss „What (All)“ auch nur bestimmte Strukturvariablen ausgelesen werden. Die gültigen Werte für „What (All)“ sind folgender Tabelle zu entnehmen. Wenn mehrere Werte ausgelesen werden sollen, können die Werte für What (All) addiert werden. Wenn z.B. sowohl die Filterfrequenz als auch der Filtertyp ausgelesen werden sollen ist der Wert 12 -4 für Filtertyp und 8 für Filterfrequenz- an die Funktion zu übergeben.

Tabelle 4: What (All)-Werte für S8_GetDevSettings.vi

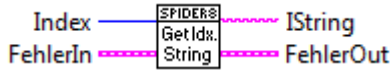
What (All)	Strukturvariable
1	NumChar
2	MeasRate
4	FilterT
8	FilterF
16	SerNum
32	SWVers
64	DevName
128	Mode

Es wird in jedem Fall ein Cluster aus allen Strukturvariablen herausgegeben, der für den jeweiligen gesuchten Wert aufgeschlüsselt werden muss. Es findet keine Kommunikation mit dem Gerät statt, sondern nur mit dem Arbeitsspeicher.

S8_GetErrorStatus.vi

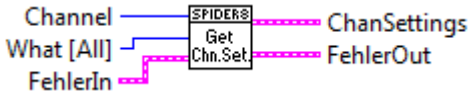
Dieses VI liest den aktuellen Fehlerstatus des Geräts aus.

S8_GetIndexString.vi



Viele Werte werden im IDS-Code zum Gerät übertragen oder zurückgegeben. Zum Beispiel werden alle Baud-Raten in diesem Code übertragen. Die Funktion S8_GetIndexString.vi wandelt den IDS-Code in einen lesbaren String um. Vorsicht: Dies ist nur bedingt als Fehlerhandler geeignet, da nur Fehler, die im IDS-Code sind, umgewandelt werden können. In der Praxis ist dieses Programm nur selten nützlich, da Werte im IDS-Code mit Ausnahme von Fehlermeldungen bereits von den VIs in der Bibliothek automatisch umgewandelt werden. Dazu wurden innerhalb der Funktions-VIs die Convert-VIs, Enum-Elemente und Case-Strukturen verwendet.

S8_GetPsbleChanSettings.vi



Dieses VI gibt alle für den angegebenen Kanal möglichen Werte heraus, die für die Strukturvariablen „ChanType“, „IsActive“, „Mode“, „Mrange“, „Unit“ und „Filter“ gesetzt werden können. Die Nummer des Kanals wird als numerischer Wert im Long-Format am Anschluss „Channel“ angegeben.

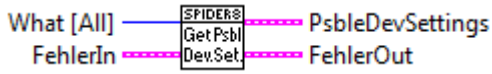
Es können auch nur für bestimmte Strukturvariablen die möglichen Parameter ermittelt werden, indem am Anschluss „What (All)“ ein numerischer Wert im Long-Format übergeben wird. Die gültigen Werte für „What (All)“ können folgender Tabelle entnommen werden. Um mehrere Strukturvariablen zu bearbeiten, können die Werte für What (All) addiert werden. So ist z.B. der Wert 66 -2 für Kanaltyp und 64 für Messbereich- an den Treiber zu übergeben, wenn der Kanaltyp und der Messbereich bearbeitetwerden soll

Tabelle 5: What (All)-Werte für S8_GetPsbleChanSettings.vi

What (All)	Strukturvariable
2	ChanType
16	IsActive
32	Mode
64	MRange
128	Unit
512	Filter

Als Ergebnis wird in jedem Fall ein Cluster aus allen in der Tabelle angegebenen Strukturvariablen ausgegeben, der aufgeschlüsselt werden kann.

S8_GetPsbleDevSettings.vi



Die Funktion funktioniert für Geräteeinstellungen genauso wie die Funktion S8_GetPsbleChanSettings.vi für Kanaleinstellungen. Es werden für die Strukturvariablen „NumChan“, „MeasRate“, „FilterT“ und „FilterF“ die möglichen Einstellungen herausgegeben.

Wenn ein numerischer Wert im Long-Format am Anschluss „What (All)“ gesetzt wird, werden nur Werte für die mit diesem „What (All)“-Wert verknüpften Strukturvariablen ermittelt. Mögliche Werte für „What All)“ können folgender Tabelle entnommen werden. Wenn die möglichen Werte für mehrere bestimmte Strukturvariablen bestimmt werden solle, können die Werte für What (All) addiert werden. So ist z.B. der Wert 12 -4 für Filtertyp und 8 für Filterfrequenz- zu übergeben, wenn sowohl Werte für die Filterfrequenz als auch für den Filtertyp ermittelt werden soll.

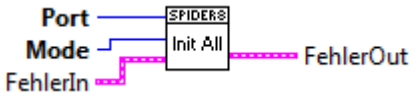
Tabelle 6: What (All)-Werte für S8_GetPsbleDevSettings.vi

What (All)	Strukturvariable
1	NumChan
2	MeasRate
4	FilterT
8	FilterF

S8_GetVersion.vi

Dieses VI gibt die Version der Firmware auf dem Spider8 als Daten im String-Format am Anschluss „Versionsstring“ aus.

S8_InitAll.vi



Dieses VI initialisiert die Verbindung mit dem Spider8. Dazu muss am Anschluss „Port“ die Nummer, die dem Anschluss am Computer zugeordnet ist, als numerischer Wert im Long-Format übergeben werden. Das geht am einfachsten durch das Kontext-Menü „Erstellen“, welches sich öffnet, wenn mit der rechten Maustaste auf den Anschluss geklickt wird.

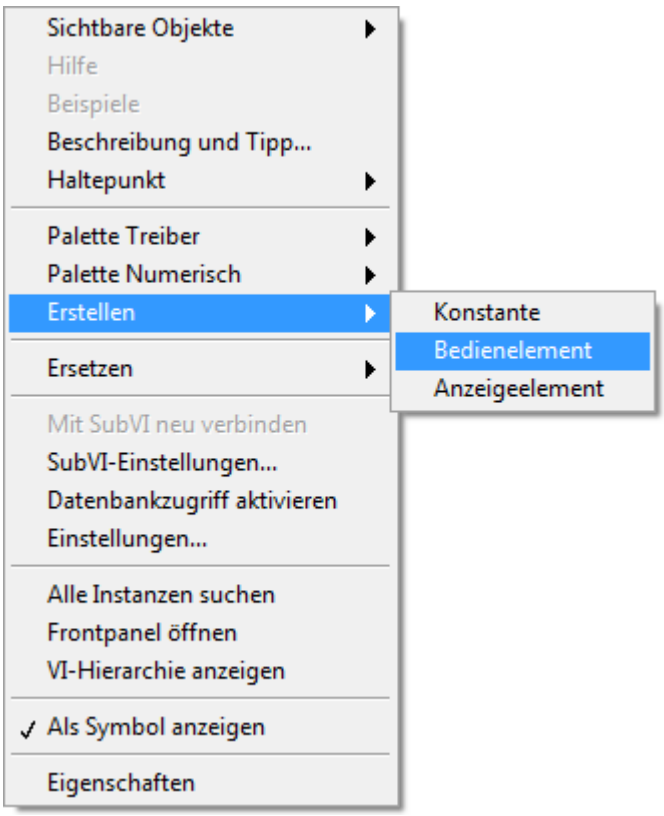
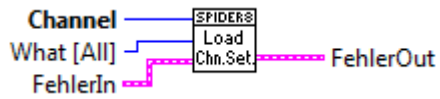


Abbildung 13: Das Kontextmenü

Hier kann nun ein Bedienelement oder eine Konstante erzeugt werden. In beiden Fällen handelt es sich um ein Enum-Element, in dem die Objekte automatisch passend definiert werden.

Am Anschluss „Mode“ muss die Baud-Rate bzw. der Kommunikationsmodus ausgewählt werden. Auch hier wird ein numerischer Wert im Long-Format angegeben, der für die entsprechende Auswahl steht, und eine Erstellung eines passenden Elements ist wie es auch beim „Port“-Anschluss erstellen über das Kontext-Menü „Erstellen“ möglich ist.

S8_LoadChanSettings.vi



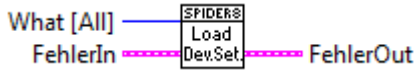
Die aktuellen Kanaleinstellungen eines Kanals werden aus dem Spider8 herausgelesen und in die Strukturvariablen im Arbeitsspeicher abgelegt. Der Kanal wird als numerischer Wert im Long-Format am Anschluss „Channel“ angegeben.

Über die Angabe eines numerischen Werts am Anschluss „What (All)“ ist es möglich nur bestimmte Kanaleinstellung abzurufen. Welche Werte für „What (All)“ gesetzt werden können, kann folgender Tabelle entnommen werden. Wenn mehrere dieser Werte herunter geladen werden sollen, muss die Summe der Einzelwerte aus dieser Tabelle als Wert für What (All) an die Funktion übergeben werden. So ist z.B. 520 -8 für Name und 512 für den Tarawert- an die Funktion zu übergeben, wenn der Kanalname und der Tarawert herunter geladen werden soll.

Tabelle 7: What (All)-Werte für S8_LoadChanSettings.vi

What (All)	Strukturvariable	What	Strukturvariable
1	NennVal	32	Mode
2	ChanType	64	MRange
4	SerNum	128	Unit
8	Name	512	TareValue
16	IsActive	1024	Filter

S8_LoadDevSettings.vi



Diese Funktion lädt die aktuellen Geräteeinstellungen vom Spider8 und speichert sie in die Strukturvariablen im Arbeitspeicher. Durch Angabe eines numerischen Wertes im Long-Format am Anschluss „What (All)“, kann auch nur bestimmte Einstellungen abgefragt werden. Gültige Werte für „What (All)“ stehen in folgender Tabelle. Wenn mehrere Strukturvariablen herunter geladen werden sollen, ist die Summe der einzelnen What (All)-Werte an die Funktion zu übertragen. So ist z.B. der Wert 12 -4 für den Filtertyp und 8 für die Filterfrequenz- zu übertragen, wenn sowohl der Filtertyp als auch die Filterfrequenz bestimmt werden sollen.

Tabelle 8: What(All)-Werte für S8_LoadDevSettings.vi

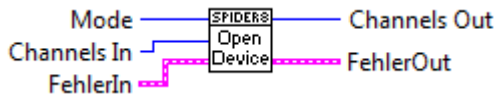
What (All)	Strukturvariable
1	NumChar
2	MeasRate
4	FilterT
8	FilterF
16	SerNum
32	SWVers
64	DevName
128	Mode

S8_MeasOneVal.vi

Es wird ein aktueller Messwert für jeden zum Messen aktivierten Kanal gemessen und in die Strukturvariablen abgespeichert. Mit S8_GetChanSettings.vi kann der Wert über die Strukturvariable „MeasValue“ herausgelesen werden.

Die Daten der I/O-Buchse werden ebenfalls mit diesem VI ausgelesen und mit S8_GetChanSettings.vi und der Strukturvariable DigIO herausgelesen.

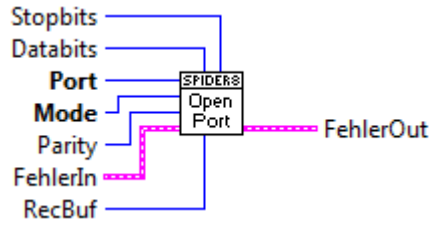
S8_OpenDevice.vi



Diese Funktion liest die aktuelle Konfiguration und die Einstellungen aus und speichert sie in den Strukturvariablen. Dazu muss am Anschluss „Mode“ über das Element DevMode.ctl angegeben werden, dass das Gerät online ist. Zusätzlich wird am Anschluss „Channels Out“ die Anzahl an gefundenen Kanälen angegeben. Dies sollten zehn Kanäle sein.

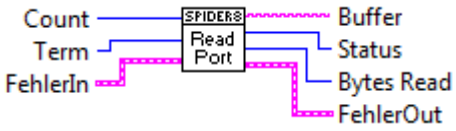
Wenn am Anschluss „Mode“ offline gewählt wurde, werden Strukturvariablen erzeugt, die zu der am Anschluss „Channels In“ im Long-Format angegebenen Anzahl an Kanale passt.

S8_OpenPort.vi



Dieses VI kann S8_InitAll.vi ersetzen, wenn eine weitergehende Konfiguration des Ports notwendig ist. Dies ist aber nur in Ausnahmefällen sinnvoll und S8_InitAll.vi sollte daher bevorzugt werden.

S8_ReadPort.vi

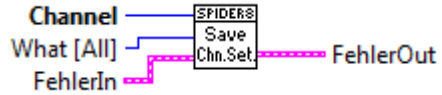


Wenn die Funktion S8_WritePort.vi verwendet wird, kann mit dieser Funktion die Antwort des Spider8 ausgelesen werden.

S8_ReadSettings.vi

Es wird eine Datei, die die Kanal und Geräteeinstellungen enthält, geladen und die Werte in die Strukturvariablen im Arbeitsspeicher abgelegt. Dazu wird der vollständige Pfad der Datei am Anschluss „FileName“ dem VI übergeben. Die Datei muss dabei als Binär-Datei mit der Endung *.SP8 vorliegen. Diese Datei kann entweder mit dem Programm Spider32 Setup oder über S8_WriteSettings.vi erstellt werden. Es findet aber keine Kommunikation mit dem Gerät statt. Um die Einstellungen an den Spider8 zu übertragen, werden die Funktionen S8_SaveChanSettings.vi und S8_SaveDevSettings.vi benötig.

S8_SaveChanSettings.vi



Die Werte der Strukturvariablen mit den Kanaleinstellungen für einen Kanal werden an das Gerät übertragen. Der Kanal wird dabei am Anschluss „Channel“ als numerischer Wert im Long-Format angegeben.

Es ist möglich, nur bestimmte Strukturvariable zu übermitteln, indem ein numerischer Wert am Anschluss „What (All)“ gesetzt wird. Gültige Werte für „What (All)“ stehen in folgender Tabelle. Die Werte der nicht in der Tabelle aufgeführten Strukturvariablen ergeben sich aus den aufgeführten Werten. So ergibt sich z. B. der Wert unter „NennVal“ durch eine Eingabe von „MRange“. Wenn mehrere Werte übertragen werden sollen, ist die Summe der einzelnen Werte für What (All) zu übertragen. So ist z. B. der Wert 520 -8 für den Namen und 512 für den Tarawert- zu übergeben, wenn sowohl der Kanalname als auch der Tarawert übertragen werden soll.

Tabelle 9: What (All)-Werte für S8_SaveChanSettings.vi

What (All)	Strukturvariable	What (All)	Strukturvariable
2	ChanType	64	MRange
8	Name	512	TareValue
16	IsActive	1024	Filter
32	Mode	2048	DigIO

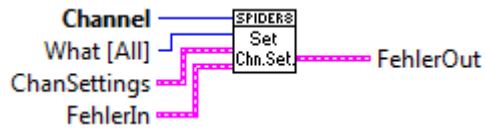
S8_SaveDevSettings.vi

Die Werte der Strukturvariablen mit den Geräteeinstellungen werden an das Gerät übertragen. Wenn nur bestimmte Werte übertragen werden sollen, muss am Anschluss „What (All)“ der entsprechende numerische Wert im Long-Format gesetzt werden. Eine Liste mit den möglichen Werten für „What (All)“ ist folgender Tabelle zu entnehmen. Für mehrere Werte ist die Summe der Einzelwerte zu übergeben. So muss der Wert 12 -4 für den Filtertyp und 8 für die Filterfrequenz- an die Funktion übergeben werden, wenn sowohl der Filtertyp als auch die Filterfrequenz übertragen werden soll.

Tabelle 10: What (All)-Werte für S8_SaveDevSettings.vi

What (All)	Strukturvariable
2	MeasRate
4	FilterT
8	FilterF
64	DevName

S8_SetChanSettings.vi

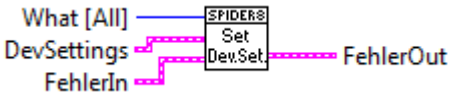


Es wird ein Cluster entsprechend dem Element ChanType.ctl mit neuen Werten für die Strukturvariablen eingelesen und in den Arbeitsspeicher abgelegt. Dabei können auch nur die Werte für bestimmte Strukturvariablen übertragen werden, indem man beim Anschluss „What (All)“ für die Strukturvariablen den zugeordneten numerischen Wert im Long-Format angibt. Eine Liste mit passenden Werten kann folgender Tabelle entnommen werden. Wenn mehrere Werte übertragen werden sollen, ist die Summe der Einzelwerte zu übergeben. So ist z.B. der Wert 520 -8 für den Namen und 512 für den Tarawert- an die Funktion zu übergeben, wenn sowohl der Kanalname als auch der Tarawert geändert werden soll.

Tabelle 11: What (All)-Werte für S8_SetChanSettings.vi

What (All)	Strukturvariable	What (All)	Strukturvariable
2	ChanType	64	MRange
8	Name	512	TareValue
16	IsActive	1024	Filter
32	Mode	2048	DigIO

S8_SetDevSettings.vi



Dieses VI ändert den Wert einer Strukturvariablen im Arbeitsspeicher. Dazu wird am Anschluss „DevSettings“ das Element DevType.ctl angebunden, in dem die zu ändernden Werte eingegeben werden.

Über den Anschuss „What (All)“ muss der numerische Wert im Long-Format übergeben werden, der für die zu ändernde Variable steht. Es sind nicht alle Wertekombinationen, die in das Element DevType.ctl eingebbar sind, an das Gerät übertragbar. So ist z. B. bei einer Messrate von 200Hz keine Filterfrequenz von 40Hz möglich. Deswegen wird eine Übergabe von mehr als einem Wert gleichzeitig an die Strukturvariablen in dieser Funktion nicht unterstützt. Eine Angabe am Anschluss von „What (All)“ ist notwendig. Wenn mehr als ein Wert geändert werden soll, muss das VI mehrfach durchgeführt werden. Die restlichen Strukturvariablen werden dann automatisch mit einer möglichen Wertekombination gefüllt, die möglichst genau den alten Werten entsprechen.

Mögliche Kombinationen können der Hilfe-Datei entnommen werden, aber es wird empfohlen, die Geräteeinstellungen über Spider32 Setup vorzunehmen und in einer Datei mit der Funktion S8_ReadSettings.vi in die Strukturvariablen zu übertragen. Mögliche Werte für „What (All)“ können folgender Tabelle entnommen werden.

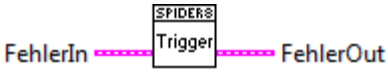
Tabelle 12: What (All)-Werte für S8_SetDevSettings.vi

What (All)	Strukturvariable
2	MeasRate
4	FilterT
8	FilterF
64	DevName

S8_TareSetup.vi

Aktiviert einen oder mehrere Kanäle zum Tarieren. Die Kanäle werden als 1D-Array am Anschluss „Channels“ zum VI übertragen.

S8_Trigger.vi



Löst einen Cmd Trigger aus.

S8_WritePort

Mit diesem VI können Befehlstrings direkt an den Spider8 übertragen werden. Diese müssen als String am Anschluss „Buffer“ an das VI übergeben werden. Die möglichen Strings und ihre Syntax können der S8_Command_Help Hilfedatei entnommen werden. Um einen Antwortstring zu erhalten, muss S8_ReadPort.vi ausgeführt werden.

S8_WriteSettings.vi

Speichert die aktuellen Strukturvariablen entweder in einer ASCII-Datei oder in einer Binär-Datei. Der Speicherort wird als Pfad am Anschluss „FileName“ und der Datei-Typ als numerischer Wert im Long-Format am Anschluss „FileType“ übergeben. „0“ steht für eine Binärdatei und „1“ für eine ASCII-Datei. Die Binär-Datei kann danach dazu verwendet werden die Einstellungen mittels S8_ReadSettings.vi wieder zu laden, während die ASCII-Datei für den Benutzer lesbar ist.

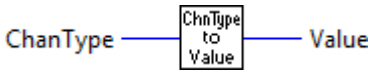
5.3 Convert

Innerhalb der Funktionen werden diese VIs verwendet, um aus den vordefinierten Elementen für den Treiber passende Werte im IDS-Code zu erzeugen bzw. Werte im IDS-Code in Werte für die vordefinierten Elemente umzuwandeln. Wenn mit den Funktionen und VIs in der Bibliothek gearbeitet wird, ist eine Verwendung dieser VIs so gut wie nie erforderlich, da diese bereits in den Funktionen benutzt wurden.

Convert ChanActive to Value

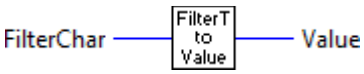
Wandelt die Daten des Enum-Elements ChanActive.ctl in IDS-Daten für den Treiber um.

Convert ChanType to Value



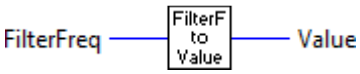
Wandelt die Daten des Enum-Elements ChanType.ctl in IDS-Daten für den Treiber um.

Convert FilterChar to Value



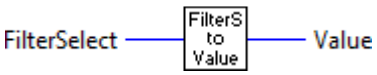
Wandelt die Daten des Enum-Elements FilterChar.ctl in IDS-Daten für den Treiber um.

Convert FilterFreq to Value



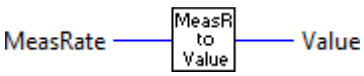
Wandelt die Daten des Enum-Elements FilterFreq.ctl in IDS-Daten für den Treiber um.

Convert Filter Select to Value



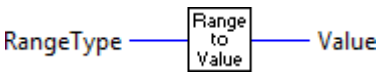
Wandelt die Daten des Enum-Elements FilterSelect.ctl in IDS-Daten für den Treiber um.

Convert MeasRate to Value



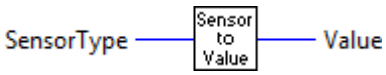
Wandelt die Daten des Enum-Elements MeasRate.ctl in IDS-Daten für den Treiber um.

Convert Range to Value



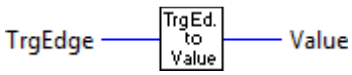
Wandelt die Daten des Enum-Elements RangeType.ctl in IDS-Daten für den Treiber um.

Convert SensorType to Value



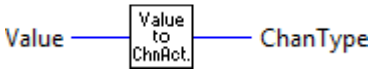
Wandelt die Daten des Enum-Elements SensorType.ctl in IDS-Daten für den Treiber um.

Convert TrgEdge to Value



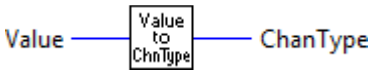
Wandelt die Daten des Enum-Elements TrgEdge.ctl in IDS-Daten für den Treiber um.

Convert Value to ChanActive



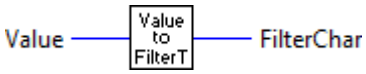
Wandelt IDS-Daten in Daten für das Enum-Element ChanActive.ctl um.

Convert Value to ChanType



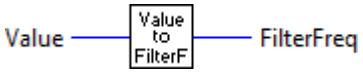
Wandelt IDS-Daten in Daten für das Enum-Element ChanType.ctl um.

Convert Value to FilterChar



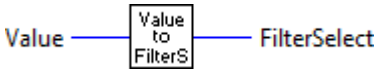
Wandelt IDS-Daten in Daten für das Enum-Element FilterChar.ctl um.

Convert Value to FilterFreq



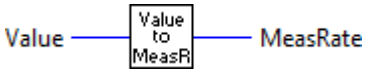
Wandelt IDS Daten in Daten für das Enum-Element FilterFreq.ctl um.

Convert Value to FilterSelect



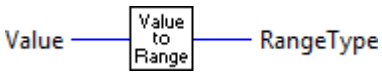
Wandelt IDS-Daten in Daten für das Enum-Element FilterSelect.ctl um.

Convert Value to MeasRate



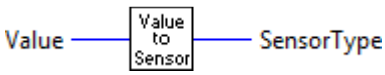
Wandelt IDS-Daten in Daten für das Enum Element MeasRate.ctl um.

Convert Value to Range



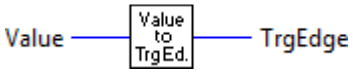
Wandelt IDS-Daten in Daten für das Enum-Element RangeType.ctl um.

Convert Value to SensorType



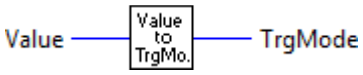
Wandelt IDS-Daten in Daten für das Enum-Element SensorType.ctl um.

Convert Value to TrgEdge



Wandel IDS-Daten in Daten für das Enum-Element TriggerEdge.ctl um.

Convert Value to TrgMode

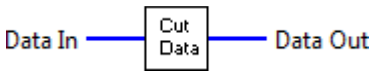


Wandelt IDS-Daten in Daten für das Enum-Element TriggerMode.ctl um.

5.4 VIs

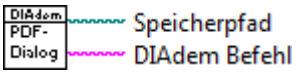
Diese VIs fassen mehrere Funktionen zu Sequenzen zusammen oder erleichtern auf andere Art und Weise die Programmerstellung in LabVIEW. In den VIs, die Treiberfunktionen aufrufen, werden ebenfalls ein Fehlercluster und Fehleraus- sowie -eingänge verwendet, die kompatibel mit dem Fehlercluster der Treiberfunktionen sind.

Cut Data.vi



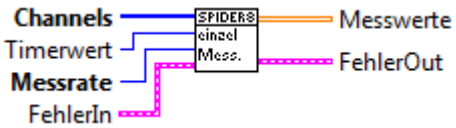
Wenn während einer kontinuierlichen Messung ein Fehler auftritt, kann es passieren, dass die letzten Werte durch Nullen ersetzt werden. Dieses VI löscht diese Nullen aus einem 1D-Array und passt die Größe an.

DIAdem_Speicherpfad



Öffnet einen Dateimonitor, in dem ein Dateipfad für eine PDF-Datei ausgewählt werden kann, und wandelt diesen Dateipfad in einen Befehl für Diadem zur Erstellung eines PDFs um. Der Befehl wird als String am Anschluss „DIAdem Befehl“ und der Dateipfad am Anschluss „Speicherpfad“ augegeben. Um DIAdem-Befehle aus LabVIEW heraus in DIAdem zu benutzen, ist das DIAdem Connectivity Toolkit erforderlich.

S8_Einzelmessung



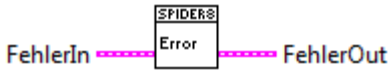
Dieses VI misst für eine beliebige Anzahl Kanäle den aktuellen Wert sowie den Timerwert in ms und gibt diese Werte aus. Die Kanäle werden als 1D-Array aus numerischen Werten im Long-Format am Anschluss „Channels“ angegeben. Der Timerwert des Starts der Messung muss am Anschluss „Timerwert“ im Long-Format übergeben werden. Dabei kann für kontinuierliche Messungen eine Messrate angegeben werden. Das VI wird dazu in einer Schleife mehrfach ausgeführt. Das Einhalten der Messrate übernimmt das Programm selbst. Die maximale Messrate ist allerdings durch aktuelle Prozessorlast und maximale Kommunikationsgeschwindigkeit über USB begrenzt und kann bei schnellen Messraten schwanken.

Außerdem erzeugt das Programm pro Schleifendurchgang nur einen aktuellen Satz von Werten. Das Speichern der vorherigen Werte muss außerhalb des VIs sichergestellt werden.

Die Messwerte werden als 2D-Array herausgegeben, wobei die Spalten für die Kanäle stehen und die letzte Spalte den Zeitstempel in ms enthält.

Die zu messenden Kanäle müssen vorher über S8_ACQSetup.vi zum Messen aktiviert werden.

S8_Fehlerhandler

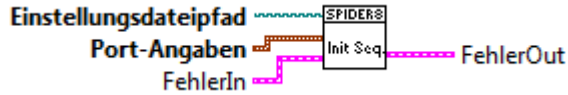


Erstellt aus einem Fehlercluster eine Fehlermeldung in englischer Sprache.

S8_Herunterfahrsequenz

Dieses VI beendet die Kommunikation mit dem Spider8 und gibt den Speicher wieder frei.

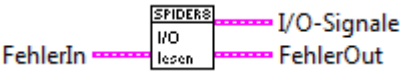
S8_InitSequenz



Dieses VI führt alle Schritte aus, die zum Initialisieren des Spider8 notwendig sind. Dazu ist eine *.SP8-Datei mit den Geräte- und Kanaleinstellungen notwendig, deren Pfad am Anschluss „Einstellungsdateipfad“ übergeben wird. Außerdem werden Angaben zum Port benötigt. Alle möglichen Angaben lassen sich dabei über ein Enum-Element auswählen.

Die Einstellungsdatei lässt sich entweder über S8_WriteSettings.vi oder über Spider32 Setup erstellen.

S8_IO_lesen



Dieses VI liest den Kanal 8 (I/O-Buchse) aus und wandelt die Signale in einen Cluster aus boolschen Werten um, der am „Anschluss I/O-Signale“ herausgegeben wird. Davor muss mit S8_ACQSetup.vi der Kanal 8 fürs Messen aktiviert werden.

S8_IO_setzen

Dieses VI setzt die Ausgänge des Kanals 8 (I/O-Buchse) gemäß eines Eingangsclusters aus boolschen Werten, der am Anschluss „Signal“ übergeben wird.

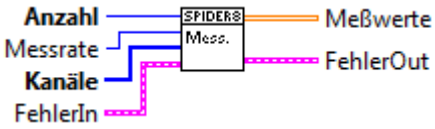
S8_kon_Messung

Dieses VI startet eine kontinuierliche Messung mit konstanter Messrate und ruft die Daten über Datenpakete von je 100 Werten ab. Um die Messung zu beenden ist die Verwendung einer globalen Variablen nötig, die als zusätzliche Abbruch-Bedingung in die Schleife eingefügt werden muss. Ansonsten wird die Schleife nur durch einen Fehler beendet.

Die zu messenden Kanäle werden als 1D-Array am Anschluss „Channels“ und die Messrate über das Element MeasRate.ctl am Anschluss „Messrate“ übergeben. Die Messwerte werden als 2D-Array im Double-Format am Anschluss „Messwerte“ herausgegeben. Dabei bilden die Spalten die Kanäle ab.

In diesem VI ist kein Trigger definiert. Die Messung beginnt unmittelbar nach Aufruf des Programms.

S8_Messung



Dieses VI misst eine vorgegebene Anzahl an Messwerten aus einem oder mehreren Kanälen und gibt die Werte als 2D-Array aus. Dabei sind die Spalten des Arrays die Kanäle.

Die Anzahl der Messwerte pro Kanal muss als numerischer Wert im Long-Format am Anschluss „Anzahl“ übergeben werden. Analog werden die Kanäle am Anschluss „Kanäle“ als 1D-Array im Long-Format und die Messrate über das Element MeasRate.ctl am Anschluss „Messrate“ verbunden. Wenn keine Messrate angegeben wird, wird mit den aktuellen Geräteeinstellungen gemessen. Die Messwerte können dann als 2D-Array am Anschluss „Messwerte“ herausgelesen werden.

In diesem VI ist kein Trigger definiert. Die Messung beginnt unmittelbar nach Aufruf des Programms.

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences
Department Maschinenbau und Produktion

Formblatt **Erklärung zur selbständigen Bearbeitung einer ausgeführten Bachelorthesis**

Zur Erläuterung des Zwecks dieses Blattes:

§ 16 Abs. 5 der APSO-TI-BM lautet:

„Zusammen mit der Thesis ist eine schriftliche Erklärung abzugeben, aus der hervorgeht, dass die Arbeit – bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit (§18 Absatz 1) – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Dieses Blatt mit der folgenden Erklärung ist nach Fertigstellung der Arbeit durch jede/n Kandidat/en/in auszufüllen und jeweils mit **Originalunterschrift** (keine Ablichtungen !) **als letztes Blatt des als Prüfungsexemplar der Bachelorthesis gekennzeichneten Exemplars einzubinden**.

Eine unrichtig abgegebene Erklärung kann - auch nachträglich - zur Ungültigkeit der Bachelor-Abschlusses führen.

Erklärung

Hiermit versichere ich,

Name: von Hebel Vorname: Markus

daß ich die vorliegende Bachelorthesis – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema

Erstellung einer an der Messung des Nachlaufs einer hydraulischen Presse erprobten Treiber und Programmdatenbank mit Programmieranleitung

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der Bachelorthesis ist erfolgt durch

_____

_____

Ort	Datum	Unterschrift im Original