Lehrstuhl für Umformtechnik

Masterarbeit

Implementierung eines Low-Cost Human-Machine Interfaces in ein Cyber Physical Production System

Florian Messner, BSc

Mai 2021

MONTANUNIVERSITÄT LEOBEN
www.unileoben.ac.at

EIDESSTATTLICHE ERKLÄRUNG

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

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

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

Datum 02.06.2021

Florian Messner

Unterschrift Verfasser/in
Florian Messner

Danksagung

An dieser Stelle möchte ich all jenen Personen, die mich beim Verfassen dieser Arbeit und im Verlauf meines Studiums unterstützt haben, meinen Dank aussprechen.

Zuerst bedanken will ich mich bei meinen Betreuern Dipl.-Ing. Marcel Sorger und Univ.-Prof. Dipl.-Ing. Dr.techn. Martin Stockinger. Die Hilfsbereitschaft, Ratschläge und fachlichen Rückmeldungen haben mich stets motiviert und tragen großen Anteil an der Verfassung dieser Masterarbeit. Besonders in Zeiten wie diesen weiß ich die tolle Betreuung und Unterstützung sehr zu schätzen.

Ebenso gilt mein Dank Dipl.-Ing. Dipl.-Ing. Benjamin Ralph für die Unterstützung und Anregungen, sowie dem gesamten Team des Lehrstuhls für Umformtechnik, das mir bei Fragen immer sehr freundlich weitergeholfen hat.

Nicht zuletzt herzlich danken möchte ich meinen Eltern für die Unterstützung während des gesamten Studiums. Ebenso bedanke ich mich bei meiner Familie, meinen Freunden und Studienkollegen, die mich stets ermutigt und dazu beigetragen haben, dass ich sehr schöne Jahre in Leoben verbringen durfte.

Abstract

In the era of digitalization, more and more production facilities are being modernized and networked, mostly driven by large industrial concerns. With Industry 4.0, fully networked smart factories are emerging, which results in a deconstruction of the classic automation pyramid through the use of Cyber Physical Production Systems (CPPS). The implementation of new digitalization technologies based on the fourth industrial revolution is mainly driven by large industrial companies due to the usually high initial investment costs and the required know-how. Low-Cost Intelligent Automation (LCIA) aims to benefit of these technologies on a smaller scale, which makes this concept interesting for small and medium-sized enterprises (SMEs) with fewer financial resources and specialized personnel. Regardless of the size of a company, the involvement of skilled personnel at shop-floor level is essential for a successful digital transformation, hence Human-Machine Interfaces (HMI) play a central role in this development.

This thesis describes the implementation of such an HMI using the example of a rolling mill. A proof of concept for LCIA implementation is demonstrated by using the Open Source programming language Python for development and inexpensive but appropriate hardware components. The raw data generated by the existing CPPS is automatically processed and filtered to create high quality data sets. A Graphical User Interface (GUI) allows the user to monitor the time history of important process parameters. This enables the machine operator to visualize, assess and, if necessary, react to occurring abnormal values, systematic errors and set machine parameters for each test performed.

For the implementation at the Chair of Metal Forming, possible concepts were compared and evaluated in the initial stage. The interface was realized by applying a touch screen, which is operated by a single board computer, integrated into the chair's internal network. This concept should subsequently support the long-term establishment of new digitalization technologies through effective and employee-centered visualization and highlight opportunities for efficient and effective digital transformation in SMEs.

Kurzfassung

In Zeiten der Digitalisierung werden, meist angetrieben von großen Industriekonzernen, immer mehr Produktionsanlagen modernisiert und vernetzt. Mit der Industrie 4.0 entstehen voll vernetzte Smart Factories, welche durch den Einsatz von Cyber Physical Production Systems (CPPS) eine Auflösung der klassischen Automatisierungspyramide zur Folge haben. Die Implementierung neuer Digitalisierungstechnologien auf Basis der vierten industriellen Revolution wird aufgrund der meist hohen initialen Investitionskosten und benötigtem Know-how mehrheitlich von großindustriellen Unternehmen vorangetrieben. Low-Cost Intelligent Automation (LCIA) verfolgt das Ziel, die Vorteile dieser Technologien auch im kleineren Maßstab zu nutzen, was dieses Konzept auch für kleine und mittlere Unternehmen (KMUs) mit geringeren finanziellen Mitteln und Fachpersonal interessant macht. Unabhängig von der Größe eines Unternehmens ist die Einbindung von Fachpersonal auf shop-floor Ebene für eine erfolgreiche digitale Transformation unerlässlich, weswegen Human-Machine Interfaces (HMI) eine zentrale Rolle in dieser Entwicklung einnehmen.

Diese Arbeit beschreibt am Beispiel eines Walzwerkes die Implementierung eines solchen HMI. Durch die Entwicklung mit der Open Source-Programmiersprache Python sowie dem Einsatz günstiger, aber geeigneter Hardwarekomponenten wird die Machbarkeit von LCIA Konzepten nachgewiesen. Die vom vorhandenen CPPS generierten Rohdaten werden zur Schaffung qualitativ hochwertiger Datensätze automatisiert aufbereitet und gefiltert. Eine Graphical User Interface (GUI) ermöglicht dem Anwender die Darstellung des zeitlichen Verlaufs relevanter Prozesskenngrößen. Dadurch ist der Maschinenbediener in der Lage, auftretende Extremwerte, systematische Fehler und eingestellte Maschinenparameter für jeden durchgeführten Versuch zu visualisieren, beurteilen und gegebenenfalls darauf zu reagieren.

Für die Implementierung am Lehrstuhl für Umformtechnik wurden in erster Instanz mögliche Konzepte verglichen und bewertet. Realisiert wurde die HMI-Schnittstelle mittels eines Touchscreens, welcher von einem in das lehrstuhlinterne Netzwerk eingebundenen Einplatinencomputer betrieben wird. Dieses Konzept soll in weiterer Folge die langfristige Etablierung neuer Digitalisierungstechnologien durch effektive und mitarbeiterzentrierte Visualisierung unterstützen und Möglichkeiten der effizienten und effektiven digitalen Transformation in KMUs aufzeigen.

Inhaltsverzeichnis

Abkürzungsverzeichnis

Zeichen	Einheit	Beschreibung
A	[-]	Auslaufpunkt
α_0	[°]	Walzwinkel
b	[m]	Walzgutbreite
b_0	[m]	Anfangsbreite
b_1	[m]	Endbreite
C	$[kN/mm]$	Gerüstmodul
E	[-]	Einlaufpunkt
F	[N]	Walzkraft
ε_h	[-]	Bezogene Formänderung
F_N	[N]	Normalkraft
F_{Reib}	[N]	Reibkraft
$F_{Rück}$	[N]	Rückstoßende Kraft
F_{Einzug}	[N]	Einziehende Kraft
h_0	[m]	Einlaufhöhe des Walzgutes
h_1	[m]	Auslaufhöhe des Walzgutes
Δ_h	[m]	Stichabnahme
k_f	$[N/mm^2]$	Fließspannung
k_{fm}	$[N/mm^2]$	Mittlere Fließspannung
k_w	$[N/mm^2]$	Umformwiderstand
k_{wm}	$[N/mm^2]$	Mittlerer Umformwiderstand
l_0	[m]	Anfangslänge
l_1	[m]	Endlänge
l_d	[m]	Gedrückte Länge
μ_R	[-]	Reibkoeffizient
μ	[-]	Erwartungswert der Normalverteilung
Q_f	[-]	Geometriefaktor
R	[m]	Walzenradius
s_0	[m]	Anfangswalzspalthöhe
s_1	[m]	Endwalzspalthöhe

σ_z	$[N/mm^2]$	Normalspannung in Z-Richtung
σ	[-]	Standardabweichung der Normalverteilung
u_0	[m/s]	Einlaufgeschwindigkeit des Walzgutes
u_1	[m/s]	Auslaufgeschwindigkeit des Walzgutes
V	$[m^3]$	Volumen
v_u	[m/s]	Umfangsgeschwindigkeit der Walze
φ	[-]	Umformgrad

AI	Artificial Intelligence
BCI	Brain Computer Interface
CPS	Cyber Physical System
CPPS	Cyber Physical Production System
CSV	Comma-Separated Value
DT	Digital Twin
DS	Digital Shadow
DT-CPPS	Digital Twin based Cyper Physical Production System
ERP	Enterprise Resource Planning
GUI	Graphical User Interface
HMI	Human-Machine Interaction
HRC	Human-Robot Collaboration
IoT	Internet of Things
KMU	Kleine und mittlere Unternehmen
KMD	Kraftmessdose
LCIA	Low-Cost Intelligent Automation
LVDT	Linear Variable Differential Transformer
MES	Manufacturing Execution System
nexDT	Next Generation Digital Twin
SBC	Single Board Computer
SCADA	Supervisory Control and Data Acquisition
SME	Small and medium-sized enterprises
SPS	Steuerprogrammierbare Speicherung
TUI	Touch User Interface

Kapitel 1

Einleitung

Industrie 4.0 eröffnet durch die Verbindung von Produkten und Dienstleistungen mit entsprechender Software und Hardware und die dadurch ermöglichte Vernetzung neue industrielle Perspektiven. In den Regierungsprogrammen der technologisch führenden Länder hat die fortschreitende Digitalisierung und die Adaption an neue industrielle Voraussetzungen längst höchste Wichtigkeit. [1, 2]

Mit der steigenden Leistungsfähigkeit des Internets und der Entwicklung kleinster Elektronikkomponenten ermöglichen vernetze Mikrocomputer den Zusammenschluss der physischen und der virtuellen Welt (Cyberspace) in der Form von Cyber Physical Systems (CPS). Durch das 2012 eingeführte Internetprotokoll IPv62 stehen ausreichend Internetadressen zur Verfügung um intelligenten Produkten und Objekten die Möglichkeit der vollständigen Vernetzung zu geben und damit das Internet of Things (IoT) zu schaffen. [3]
Das IoT verlangt die Kombination aus Sensorik-, Computertechnik- und Netzwerktechnologien. Die Voraussetzungen für leistungsstarke Systeme und Netzwerke wurden durch rasante Entwicklungen, vor allem in der Halbleitertechnik, geschaffen. Unternehmen können durch zunehmend sicherere, zuverlässigere und schnellere drahtlose Netzwerke den Datenaustausch innerhalb und außerhalb der Produktionsstätten effektiv integrieren. [3, 4]

Die Hochschulforschung kann mit Pilotprojekten wie Smart Factories oder einzelnen hochtechnologischen Anlagen einen großen Beitrag zu diesem Fortschritt leisten. An der Montanuniversität Leoben, am Lehrstuhl für Umformtechnik, beschäftigt man sich mit der Digitalisierung konventioneller Anlagen. Durch die Digitalisierung dieser Anlagen entstehen CPS für den Einsatz im fertigungstechnischen Umfeld, welche unter dem Begriff Cyber Physical Production System (CPPS) zusammengefasst werden können. Diese CPPS umfassen die Erfassung prozessrelevanter Messgrößen sowie die Verarbeitung, welche durch die horizontale und vertikale Datenintegration eine datenbasierte Entscheidungshilfe unterstützt. [5]

Im Zuge der Industrie 4.0 wird viel über die Rolle, welche der Mensch in einer digitalisierten Arbeitsumgebung einnimmt, diskutiert. Während das Hauptaugenmerk heute auf der Entwicklung von Smart Factories und möglichst automatisierter Systeme liegt, soll die human-zentrierte Digitalisierung die menschliche Intelligenz wieder stärker einbinden. Durch Weiterentwicklungen in der Robotik und im Bereich der Artificial Intelligence (AI) sind intelligentere Roboter zu vertretbaren Kosten verfügbar. Aus diesem Grund wird die Industrie 5.0 als eine Zusammenarbeit von Mensch und Roboter, auch auf shop-floor Ebene, beschrieben. Zur Umsetzung werden neue Technologien, beispielsweise intelligente Sensoren, die zur Entlastung der Netzwerke Datenmengen durch Datenaufbereitung reduzieren, benötigt. Der Mensch übernimmt in dieser Zusammenarbeit, neben kreativen Prozessen, Aufgaben wie die Programmierung, das Training oder die Wartung von Robotern. [6, 7]

Kapitel 2

Grundlagen

In folgendem Kapitel werden die wichtigsten Grundlagen dieser Arbeit erläutert. Dadurch soll die Relevanz bezüglich Automatisierung, Datenaufbereitung, Digitalisierung und Mensch-Maschine Interaktionen aufgezeigt werden.

2.1 Grundgrößen Umformtechnik

Umformtechnik ist ein Teilgebiet der Fertigungstechnik. Nach DIN 8580 wird Umformen als gezielte, plastische Änderung der Form eines geometrisch festen Körpers, wobei sowohl Masse als auch Stoffzusammenhalt erhalten bleiben, beschrieben. Dabei sind von außen wirkende Kräfte für eine Verschiebung und Umlagerung der Stoffteilchen verantwortlich. [8, 9]
Abbildung 2.1 zeigt die Einteilung der Umformverfahren nach DIN 8582 gemäß des beim Umformern vorherrschenden Spannungszustands.

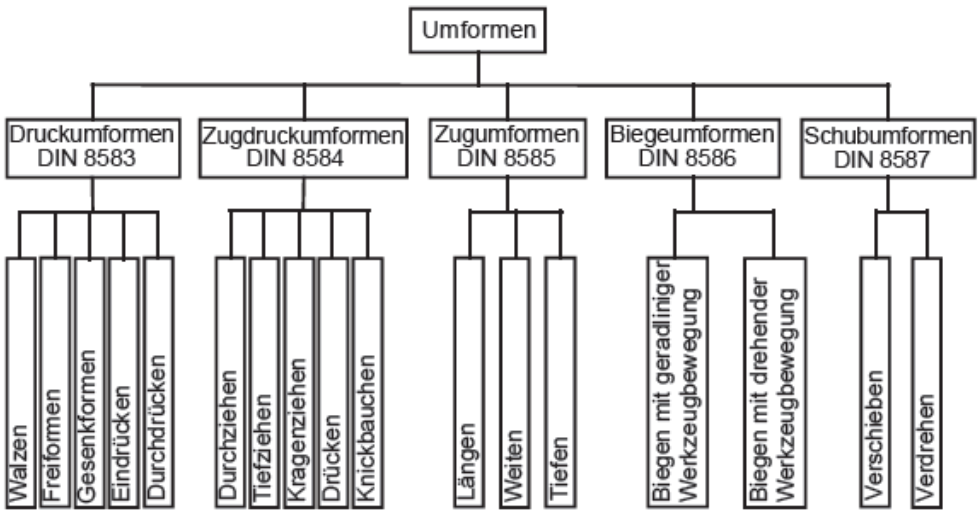


Abbildung 2.1: Einteilung der Umformverfahren nach Spannungszustand [9]

Beim Umformen bleibt das Volumen des plastisch verformten Körpers unverändert. Diese Volumenkonstanz kann für quaderförmige Werkstücke wie folgt beschrieben werden: [10]

$$V = l_0 \cdot h_0 \cdot b_0 = l_1 \cdot h_1 \cdot b_1 = const. \tag{2.1}$$

Der Index 0 wird für Größen vor dem Umformvorgang verwendet, mit dem Index 1 handelt es sich um Größen nach dem Umformvorgang. Die Seitenverhältnisse des Werkstücks h_1/h_0, l_1/l_0 und b_1/b_0 werden Formänderungsverhältnisse genannt. Die bezogene Formänderung (Dehnung) ist das Verhältnis der absoluten Formänderung zum Ausgangsmaß. Für die Werkstückhöhe h gilt beispielsweise: [11]

$$\varepsilon_h = \frac{\Delta h}{h_0} = \frac{h_1 - h_0}{h_0} \tag{2.2}$$

Durch Umsetzen und Logarithmieren der Gleichung 2.1 erhält man: [10]

$$\ln \frac{l_1}{l_0} + \ln \frac{h_1}{h_0} + \ln \frac{b_1}{b_0} = 0 \tag{2.3}$$

Diese Gleichung kann auch geschrieben werden als:

$$\varphi_l + \varphi_h + \varphi_b = 0 \tag{2.4}$$

wobei φ als Umformgrad oder auch als logarithmische Formänderung bezeichnet wird. Der Umformgrad nimmt in umformtechnischen Berechnungen, beispielsweise für den Kraft- und Arbeitsbedarf, eine zentrale Rolle ein. Die einzelnen Beiträge werden oft als Stauchungs-, Breitungs- und Längungsgrad bezeichnet. Die Vorteile der Verwendung des Umformgrads als Kenngröße sind einerseits die Beurteilung der Verformungsrichtung am Vorzeichen des Wertes, andererseits kann bei stufenweiser Umformung durch Addition der Einzelbeiträge die Gesamtumformung berechnet werden. [8, 11, 12]

2.2 Walzen

Nach DIN 8583 wird Walzen den Druckumformverfahren zugeordnet (Abbildung 2.1). Meist gegenüberliegende, sich gegenläufig drehende Walzen bringen Druckspannungen in den Werkstoff ein, wodurch dieser plastisch zu fließen beginnt. Typisch ist die inkrementelle Umformung, also eine schrittweise Umformung des Walzgutes. [10, 11]

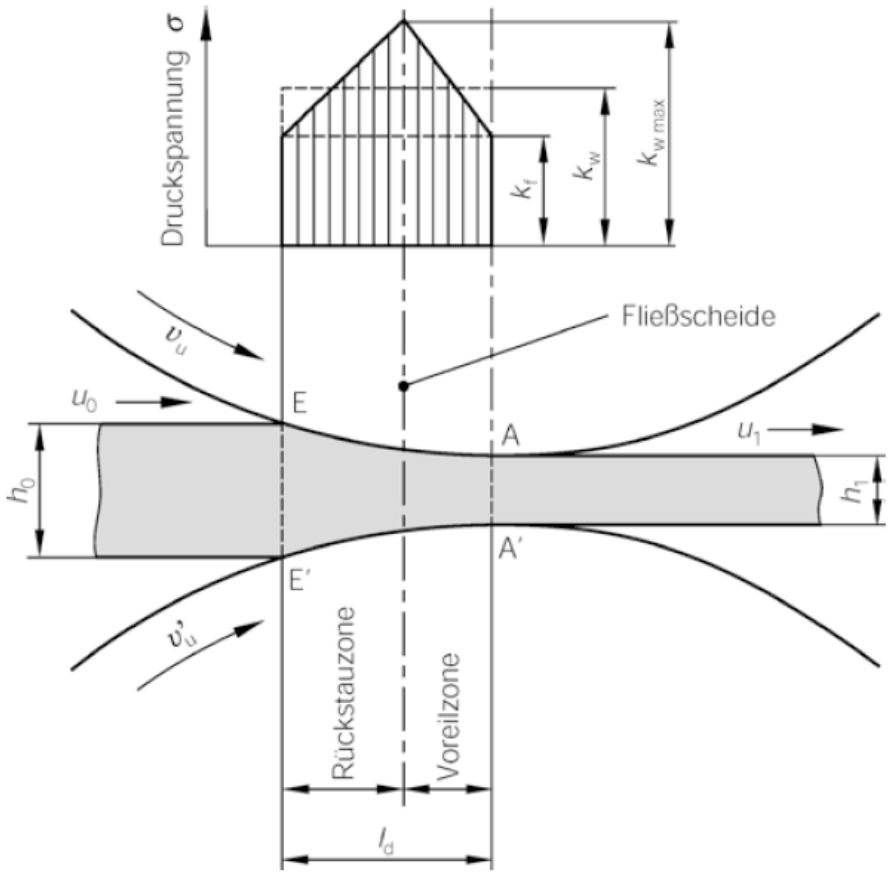


Abbildung 2.2: Geometrische Verhältnisse beim Längswalzen von Flachprofilen [13]

In Abbildung 2.2 sind die wichtigsten geometrischen Größen des Längswalzprozesses dargestellt. Das Walzgut mit der Einlaufdicke h_0 und Breite b_0 verlässt den Walzspalt mit den Abmessungen h_1 und b_1. Die dafür notwendige Kraft wird allgemein als Walzkraft F bezeichnet. Analytisch wird diese durch Integration aller vertikalen Komponenten der im Walzspalt wirkenden Kräfte ermittelt. [13]

$$F = \int_0^{l_d} b \cdot \sigma_z(x) dx \tag{2.5}$$

In der Praxis reicht es meist den genauen Druckverlauf durch einen mittleren Umformwiderstand k_{wm} zu ersetzen. Der Umformwiderstand k_w berücksichtigt neben der werkstoffabhängigen Fließspannung k_f auch den Umformwirkungsgrad, abhängig von Eigenschaften und Geometrie der Umformzone. [11]

$$k_{wm} = k_{fm} \cdot Q_f \text{ oder } k_w = k_f \cdot Q_f \tag{2.6}$$

Damit kann die Kraft wie folgt ermittelt werden: [11]

$$F = b \cdot l_d \cdot k_{wm} \tag{2.7}$$

l_d bezeichnet die gedrückte Länge und kann bei Kenntnis der Walzengeometrie (Radius R) und der Stichabnahme Δh nach

$$l_d = \sqrt{R \cdot \Delta h} \tag{2.8}$$

ermittelt werden. [11, 8]
Grundlegende Voraussetzung für den Walzenprozess ist der Einzug des Walzgutes in den Walzspalt (siehe Abbildung 2.3). Der notwendige Vorschub wird durch die Reibung zwischen den rotierenden Walzen und dem Walzgut erzeugt. Die Reibkraft F_R lässt sich mittels dem Reibungskoeffizienten μ_R und der Normalkraft F_N ermitteln. [11]

$$F_{Reib} = \mu_R \cdot F_N \tag{2.9}$$

Da gelten muss $F_{Einzug} \geqslant F_{Rück}$ kann folgender Zusammenhang beschrieben und vereinfacht werden: [11]

$$\mu \cdot F_N \cdot \cos \alpha_0 \geqslant F_N \cdot \sin \alpha_0 \tag{2.10}$$

$$\mu \geqslant \tan \alpha_0 \tag{2.11}$$

In der Umformtechnik wird dieser Zusammenhang als Greifbedingung bezeichnet. [11]

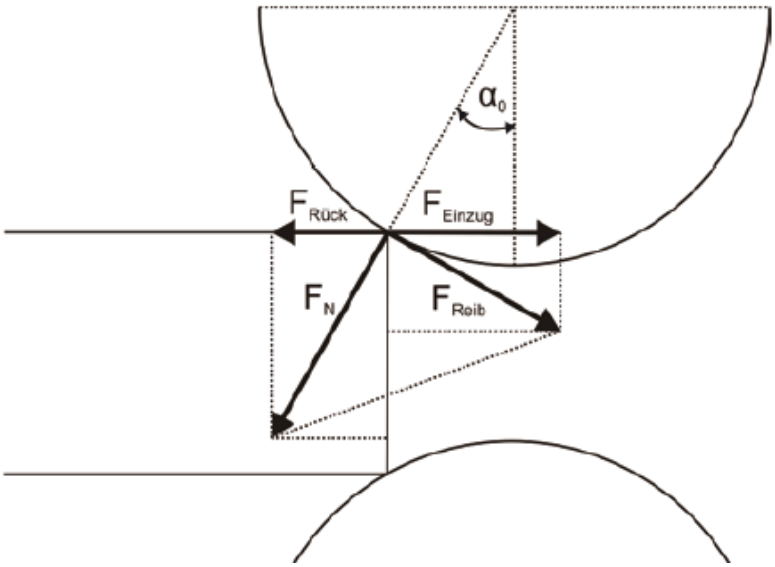


Abbildung 2.3: Wirkende Kräfte während des Walzguteinzugs [11]

Das Walzspaltdiagramm Abbildung 2.4 stellt den Zusammenhang zwischen auftretender Walzkraft und Walzspalthöhe dar. Durch die elastische Auffederung des Walzgerüstes nimmt die Walzspalthöhe und in Folge auch die auslaufende Walzgutdicke zu. Der Schnittpunkt der Werkstoffkennlinie des Walzguts (rechte Kennlinie in Abbildung 2.4) mit der Gerüstkennlinie wird Arbeitspunkt genannt. Im diesem Punkt lassen sich die Walzkraft F_1, die Walzspalthöhe s_1 und die Auslaufdicke des Walzgutes h_1 ablesen. Der

Gerüstmodul C, die Steigung der Gerüstkennlinie, wird für die rechnerische Ermittlung der tatsächlichen Endhöhe des Walzguts h_1 benötigt. [11]

$$h_1 = s_0 + \frac{F}{C} \tag{2.12}$$

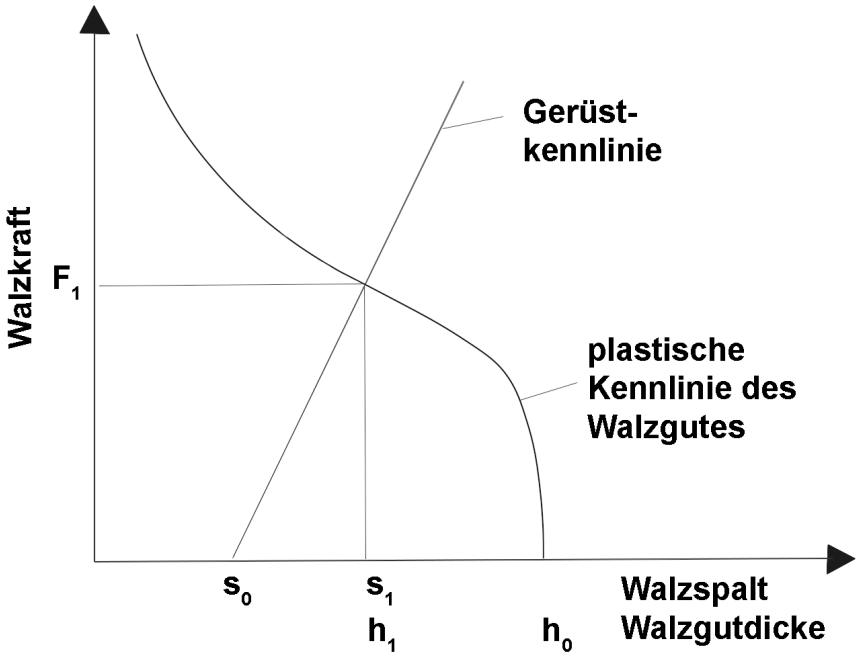


Abbildung 2.4: Schematische Darstellung Walzspaltdiagramm [12]

2.3 Automatisierungspyramide

Die Automatisierungspyramide mit ihrer hierarchischen Struktur kann als Grundlage der vierten industriellen Revolution betrachtet werden. Nach dem IEC-62264 Standard besteht diese Pyramide aus fünf verschiedenen Ebenen. Auch wenn durch die Einführung von CPS und CPPS (siehe Kapitel 2.4) eine Dezentralisierung und damit die Auflösung der klassischen Automatisierungspyramide beobachtet werden kann, ist sie nach wie vor die Basis vieler Produktionssysteme, weshalb die Ebenen der Automatisierungspyramide nachfolgend beschrieben werden. [14]
Die fünf Hierarchieebenen sind in Abbildung 2.5 dargestellt und wie folgt definiert:

- Ebene 0 - Feldebene:
  Die Feldebene umfasst den eigentlichen Produktionsbereich. Die Ebene beinhaltet die auf den jeweiligen Produktionsprozess abgestimmten Sensoren und Aktoren. [15]

- Ebene 1 - Steuerungsebene:
  In dieser Ebene werden meist mittels speicherprogrammierbaren Steuerungen (SPS) die in der Feldebene erfassten Sensordaten verarbeitet und resultierende Ausgangssignale zur Steuerung der Aktoren in der Feldebene erzeugt. [15]

- Ebene 2 - (Prozess-)Leitebene:
  In dieser Ebene befindet sich die Prozesskontrolle. Daten von beispielsweise verschiedenen SPS werden zusammengetragen und zur Datenerhebung, Überwachung und Prozesssteuerung, im sogenannten Supervisory Control and Data Acquisition (SCADA)-System bearbeitet. Die Ebene kann als Steuerung der Maschinensysteme in Form einer Mensch-Maschine-Schnittstelle betrachtet werden. [15]

- Ebene 3 - Betriebsebene:
  Hier passiert die Produktionsplanung und Steuerung. Typischerweise ist ein Manufacturing Execution System (MES) für die Steuerung, Lenkung und Kontrolle der Produktion verantwortlich. Dieses System stellt die Verbindung zwischen der Maschinensteuerung und der Unternehmensebene dar, und übermittelt erfasste Daten an das Enterprise Resource Planning (ERP)-System. [15]

- Ebene 4 - Unternehmensebene:
  Die oberste Ebene der Pyramide wird vom ERP-System gebildet. Hier wird die Grobplanung der Produktion, auch bezüglich Auftragsabwicklung, vom unternehmerischen Standpunkt aus durchgeführt. [15]

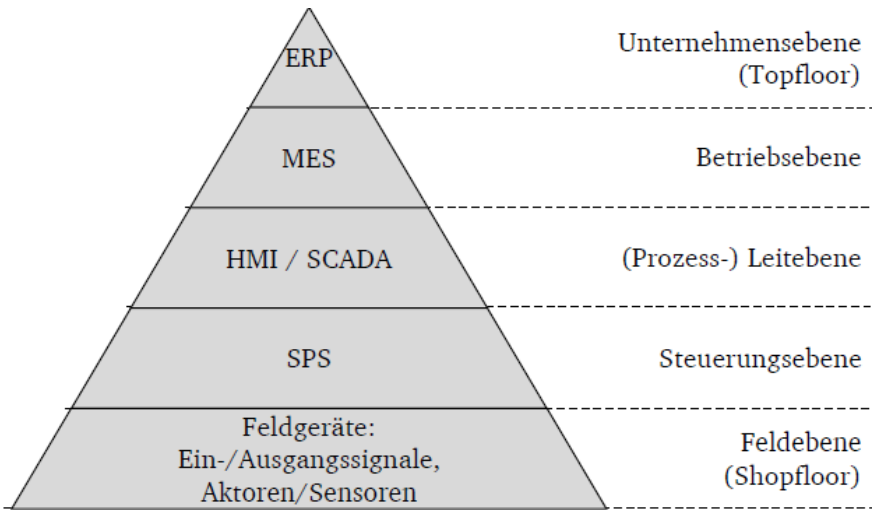


Abbildung 2.5: Automatisierungspyramide nach Siepmann [16]

Durch Fortschritte, begründet durch die Schaffung von CPS/CPPS wird der hierarchische Ansatz der Automatisierungspyramide zunehmend hinterfragt. Im Sinne der umfangreichen Vernetzung müssen die starren Ebenen durchlässiger gestaltet werden, um

den horizontalen und vertikalen Datenaustausch von Geräten und Systemen innerhalb und außerhalb der Unternehmensstruktur zu ermöglichen. Zukünftig wird eine dezentrale, ebene Vernetzungsstruktur notwendig werden, was jedoch nicht das zwangsläufige Verschwinden von etablierten Systemen in den Ebenen der Automationspyramide bedeutet. [14, 15, 17]

2.4 Cyber Physical Systems

Die Digitalisierung wird durch Entwicklungen im Hardwarebereich ermöglicht. Fortschritte in der Halbleitertechnik lassen, bei sinkenden Kosten, den Leistungszuwachs exponentiell steigen. Diese Beobachtung, bezeichnet als Mooresches Gesetz, prognostiziert auch für die nächsten Jahrzehnte eine Verdoppelung der Hardwareleistung alle 1,5 Jahre. Durch eine derart rasante Entwicklung steigt die Leistungsfähigkeit aller Systeme und immer komplexere Aufgabenstellungen werden lösbar. [4]
Ein Cyber Physical System (CPS) ist ein System, welches durch die nahtlose Integration von Berechnungsalgorithmen mit physischen Komponenten definiert wird. Sogenannte Embedded Systems überwachen und steuern mittels Sensoren und Aktoren physikalische Vorgänge. Die Kommunikation läuft dabei über entsprechende digitale Netze, auch Cyberspace genannt. Um die Vorteile eines CPS bestmöglich zu nutzen, ist die wissenschaftliche Entwicklung hinsichtlich Methodik, Technologie und Kosten von großer Bedeutung. [4, 18]
Mit dem Ziel, Zeiten für Entwicklung und Implementierung möglichst gering zu halten, werden neue Ansätze bezüglich des Aufbaus von CPS notwendig. Im Bereich Netzwerkkommunikation wurden beispielsweise die Interfaces zwischen verschiedenen Ebenen standardisiert. Die entstandene Modularität ermöglichte eine spezialisiertere Weiterentwicklung einzelner Bereiche. Dadurch sind heterogene Gesamtsysteme nach dem Plug-and-Play-Prinzip aufbaubar, wodurch neue Technologien schneller zu etablieren sind. Auch im Bereich der CPS sind Standardisierungen unabdingbare Voraussetzung die wissenschaftliche und technische Grundlage zu schaffen, um die Integration und Interaktion bestmöglich zu nutzen. [19]
In Abbildung 2.6 ist der grundsätzliche Aufbau eines CPS dargestellt. Die Kommunikation nach außen erfolgt entweder direkt zu anderen Systemen oder mit Menschen.
Eng mit dem Begriff CPS verbunden ist das Cyber Physical Production System (CPPS). Ein CPPS ist ein CPS, welches besonders auf die industriellen und fertigungstechnischen Anforderungen in der Produktion ausgelegt ist. CPPS setzen sich aus autonomen und kooperativen Elementen und Subsystemen zusammen. Diese sind in der Lage, der jeweiligen Situation angepasst, mit allen Produktionsebenen zu interagieren und auf Analytik und Simulationsdaten zuzugreifen und brechen so mit der klassischen Automationspyramide. CPPS können selbst Daten erfassen und verarbeiten, bestimmte Aufgaben steuern

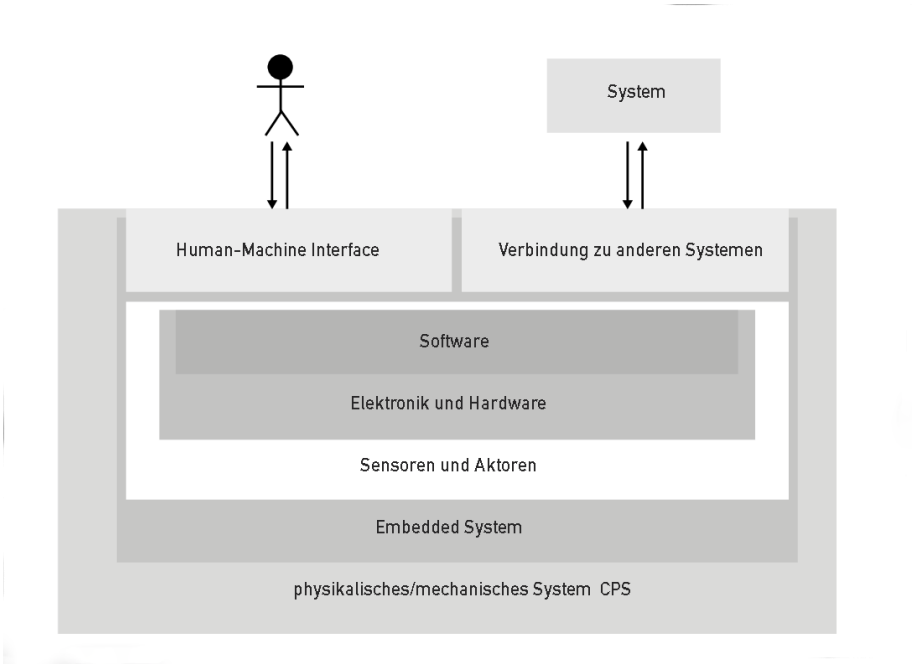


Abbildung 2.6: Interaktion zwischen Mensch und Maschine im CPS [20]

und mit Menschen über Schnittstellen interagieren. [21]
Die Datenverarbeitung entwickelt sich mit der Weiterentwicklung von CPPS zu einer zentralen Herausforderung. Einerseits ist der Wegfall möglichst aller manuell erfasster Daten (beispielsweise durch die Einführung von Radio Frequency Identification (RFID)-Geräten zur Produktverfolgung) für die CPPS Funktionalität förderlich, anderseits resultieren verwendete Sensoren in einem Anstieg heterogener Datenmengen. Ein gut funktionierendes CPPS muss daher in der Lage sein sowohl der hohen Geschwindigkeit der Datengenerierung folgen zu können als auch flexibel genug, um auf Anpassungen des Produktionsprozess reagieren zu können. [22]

2.5 Digital Twin

Durch die ständig zunehmende Erzeugung und Nutzung von Daten nimmt deren Relevanz zu, wodurch auch die Begriffe Digital Twin (DT) und Digital Shadow (DS) häufig diskutiert werden. In der wissenschaftlichen Literaturstudie [23] wurden Veröffentlichungen zum Thema DT analysiert, um aus der Vielzahl von möglichen Definitionen eine möglichst allgemein einsetzbare zu entwickeln. Demnach besteht ein DT aus adaptiven Modellen, welche das Verhalten von physischen Systemen in einem virtuellen System nachbilden, um Echtzeitdaten im Lebenszyklus zu aktualisieren. Der DT reproduziert das physikalische System um Ausfälle vorherzusagen oder Optimierungen zu ermöglichen. Typisch ist das Zusammenspiel von Simulationsmodellen und Daten, welche meist von Sensoren physisch existierender Systeme erfasst werden. Die Menge an Daten wird nach

dem Erfassen gespeichert, archiviert, aufbereitet und als digitaler Schatten bezeichnet. Neben einem detaillierten digitalen Modell (Simulationsmodelle, Wechselbeziehungen) ist jener Schatten also Teil des DT. [23, 24]
Ziel sollen sogenannte Digital Twin-based Cyber Physical Production Systems (DT-CPPS) sein. Im Unterschied zu klassischen, kosten- und zeitintensiven Simulationsprozessen um möglichst auf die Wirklichkeit übertragbare Daten zu erhalten, werden neben Engineering Daten auch Messdaten und Informationen aus Softwaresystemen verwendet. Diese Modelle werden lebenszyklusübergreifend erweitert und adaptiert. Die einzelnen Modelle sind vielfältig einsetzbar und kombinierbar. In einem DT-CPPS wird der Produktionsprozess durch den Einsatz von smarten Objekten (Teile, Maschinen, Werkzeuge etc.) mit Sensoren und Embedded Systems optimiert. Durch intelligente Kommunikation innerhalb des Prozesses können Entscheidungen schnell und autonom getroffen werden, auf Veränderungen kann flexibel reagiert werden und sämtliche Fertigungsparameter sind transparent zu überwachen. [24, 25]
Eine Erweiterung des DT wird als Next Generation Digital Twin (nexDT) beschrieben. Damit soll die Leistung und Effizienz verschiedener Systeme in allen Phasen des Produktlebenszyklus verbessert werden. Beispielsweise sollen Simulationsmodelle speziell für gewisse Umgebungsbedingungen einen Gültigkeitsbereich besitzen und entsprechend der Realbedingungen individuell eingesetzt werden. Die virtuelle Repräsentation soll mit den Echtzeitdaten direkt verknüpft und synchronisiert werden. Außerdem ist eine Integration des nexDT in das Produkt selbst denkbar, damit Kunden auch auf die gesamte virtuelle Historie zugreifen können. Für die Entwicklung zukünftiger Produktgenerationen sollen Informationen aus der Einsatzphase des Produktes in die Produktion oder das Produktdesign einfließen. [26]

2.6 Human-Machine Interaction

Mit der steigenden Technologisierung in vielen Industriebereichen nehmen auch die Anforderungen zur Interaktion zu. Unabhängig von den verschiedenen Möglichkeiten des Austauschs zwischen Mensch und Maschine gilt der allgemeine Grundsatz diese Schnittstellen so einfach und flexibel wie möglich zu gestalten. Üblicherweise wird die Interaktion als Human-Machine Interaction (HMI) bezeichnet. HMI wird, wie in Kapitel 2.3 beschrieben, als Bestandteil von SCADA-Systemen der zweiten Ebene der klassischen Automatisierungspyramide zugeordnet, findet aber auch in anderen Ebenen der Automation Verwendung. [27]
Die angesprochenen Schnittstellen werden als User Interface (UI) bezeichnet und dienen zur Kontrolle des In- bzw. Outputs. Zu einem UI zählen die Software sowie die zum Betrieb nötigen Hardwarekomponenten. Die Umsetzung des UI beeinflusst maßgeblich den notwendigen Aufwand für einen Benutzer Betriebsdaten einzugeben und bereitgestellte

Rückmeldungen des Systems zu interpretieren. In Abbildung 2.7 ist eine beispielhafte Interaktion dargestellt. Der User bekommt Rückmeldung, optisch oder akustisch, und kann seine weiteren Eingaben danach entsprechend adaptieren. [27]

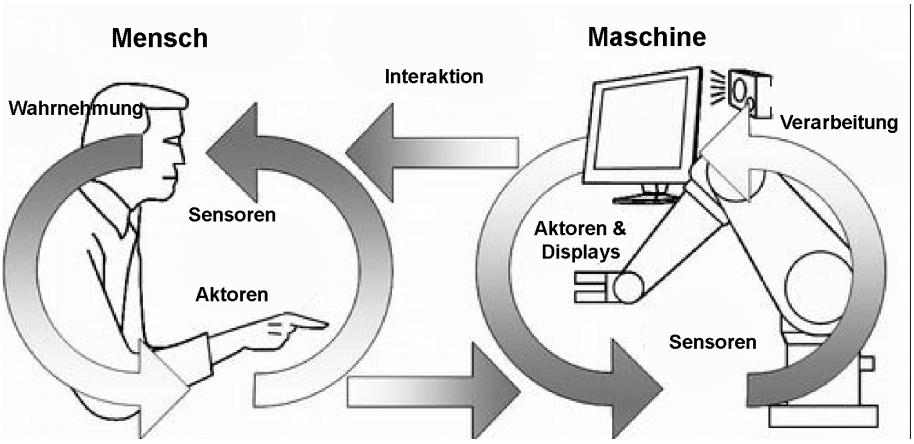


Abbildung 2.7: Beispiele für eine Human-Machine Interaction [27]

Über die letzten Jahre haben sich eine Vielzahl unterschiedlicher UIs etabliert. In nachfolgender Übersicht werden die wichtigsten Varianten aufgezählt. Da sich auch im deutschen Sprachraum durchwegs die englischen Bezeichnungen etabliert haben, wird auf eine Übersetzung bewusst verzichtet.

- Text basierende Interfaces:
  - Command Line Interface (CLI)
  - Batch Interface
  - Non-Command User Interface
  - Natural Language Interface (NLI)
- Graphische Interfaces:
  - Graphical User Interface (GUI)
  - Attentive User Interface (AUI)
  - Touch User Interface (TUI)
  - Menu driven Interface

- Aufkommende, neue Interfaces:
  - Eye Movement based Interface
  - Brain-Computer Interface
  - Motion Tracking Interface
  - Gesture User Interface
  - Voice User Interface [27]

Text basierende Interfaces, wie zum Beispiel die Steuerung über eine Kommandozeile, sind nach wie vor sehr verbreitet. Nutzer der Distribution Linux geben so den Großteil des Inputs ein. Am weitesten verbreitet sind Versionen von GUIs. Dabei wird auf einem Bildschirm Information dargestellt. Mittels Computermaus, Tastatur oder in modernen Systemen per Touchscreen kann der User Eingaben tätigen und Informationen erhalten. In aufwendigeren Applikationen kann die Befehlseingabe und Navigation auch über ein Menü verwirklicht werden. In den letzten Jahren wurden vermehrt Möglichkeiten entwickelt, abseits der gängigen Methoden zu interagieren. Dabei können Computer beispielsweise die Augen- oder Körperbewegungen des Users zur Steuerung verwenden. Vor allem durch Smartphones und smarte Alltagsgeräte ist auch die Verarbeitung von akustischem Input bekannt geworden. Allgemein etablieren sich vermehrt smarte Interaktionsmöglichkeiten, welche möglichst intuitiv bedienbar sind. Auch die Visualisierung des Outputs entfernt sich in Zeiten von Virtual Reality (VR) schrittweise vom klassischen Bildschirm weg. [27]
Ein weiteres Beispiel für Innovation in diesem Bereich sind Brain-Computer Interfaces (BCIs). Diese basieren auf der direkten, in Echtzeit gemessenen Gehirnaktivität des Users. Durch die notwendige Kombination verschiedener wissenschaftlicher Bereiche ist die Entwicklung herausfordernd und umfangreich. Die Anforderungen an BCI-Applikation bezüglich präziser, zeitnaher Signalerfassung und -verarbeitung erfordern hohe Systemleistungen. Der potentielle Nutzen rechtfertigt allerdings den Forschungsaufwand. Vor allem Menschen mit körperlicher Beeinträchtigung könnten damit Computer bedienen, sowie Roboterarme, Prothesen und Rollstühle steuern. [28]

2.7 Low-Cost Digitalisierung

Während große Betriebe seit einigen Jahren die Digitalisierung ihrer Produktionsstandorte forcieren, herrscht in KMUs Uneinigkeit über das Thema Industrie 4.0. Klar scheint zu sein, dass die vollständige Vernetzung nach großindustriellem Vorbild für KMUs ohne entsprechende finanzielle und personelle Möglichkeiten, nur schwierig bewältigbar ist.

Viele Betriebe wollen sich allerdings bewusst früh genug mit dem Thema auseinandersetzten, um den Absprung nicht zu verpassen. Um ein überstürztes Handeln zu vermeiden, sollte das Ziel sein, durch gute Planung sinnvolle Schritte zu setzen, um mit vertretbarem Ressourceneinsatz bestmöglich von der Modernisierung zu profitieren.[29]
Um das Konzept der Low-Cost Automatisierung zu verstehen, muss zuerst die Bedeutung für die Produktion aufgezeigt werden. Wenn Maschinen und Systeme nicht nur kommunizieren, sondern intelligent miteinander vernetzt sind, können in Echtzeit Informationen mit Produkten ausgetauscht werden. Dadurch kann in weiterer Folge eine Kostenreduzierung bei gleichzeitig gesteigerter Produktivität erreicht werden. Mit dem Einsatz derartiger Technologien können Produkte, welche den Qualitätsrichtlinien nicht entsprechen, leichter erkannt und entsprechend nachbearbeitet oder aussortiert werden. Durch die flexible Einbindung von neuen oder Kleinserien-Produkten in laufende Produktionsprozesse können Kundenaufträge genauer terminiert werden. [29]
Hitoshi Takeda beschäftigt sich seit einigen Jahren mit Ansätzen, um möglichst effektiv die Themen Digitalisierung und KMU mit moderatem Kostenaufwand zu vereinen. [30, 31]
Die Grundidee nennt er Low-Cost Intelligent Automation (LCIA). Nach Takeda sollen kleine Betriebe selbst firmeneigenes Know-how aufbauen. Mit dem bewussten Verzicht externe Lösungen teuer anzuschaffen, ergibt sich neben den ökonomischen Vorteilen, eine stetig steigende Kompetenz, vor allem in Bezug auf eigene Produktionsanlagen. Durch die Optimierung bekannter Tätigkeiten auf bekannten Arbeitsplätzen sollen einfache, intelligente Lösungen schnell und ohne großen Aufwand adaptierbar sein. [29]
Ein Ansatz der dabei verfolgt werden kann, ist die Nutzung von Open-Technologies. Bislang werden beispielsweise Steuerungs- und Regelsysteme von externen Betrieben implementiert. Dabei werden Software und Hardware von gewerblichen Herstellern teuer zugekauft. Durch die Etablierung von offenen Datenübertragungsstandards mit entsprechend verwendbaren Übertragungsprotokollen und Schnittstellen, entstehen neue Implemtierungsmöglichkeiten. Open-Hardware Plattformen, wie der Arduino oder Raspberry Pi (siehe Kapitel 6) die ursprünglich für den Privatanwender entwickelt wurden, passen sich an die industriellen Bedürfnisse immer mehr an. Vor allem durch die einfache Integration von verschiedensten Komponenten und die verhältnismäßig geringen Kosten sind sie bereits in vielen Anwendungen echte Alternativen, beispielsweise statt der Verwendung von SPS. In Verbindung mit Open Source Lösungen zur Entwicklung maßgeschneiderter HMIs, statt der typischerweise mitgelieferten, unflexiblen Visualisierungssoftware von Komplettlösungen, können plattformübergreifende Schnittstellen mit zur Verfügung gestellten Modulen, als Low-Cost Systeme realisiert werden. [32]
In Abbildung 2.8 werden einige im Kontext der Low-Cost Digitalisierung sinnvolle, ressourcenschonende Maßnahmen mit Ansätzen, welche diese Kriterien nicht gänzlich erfüllen, verglichen.

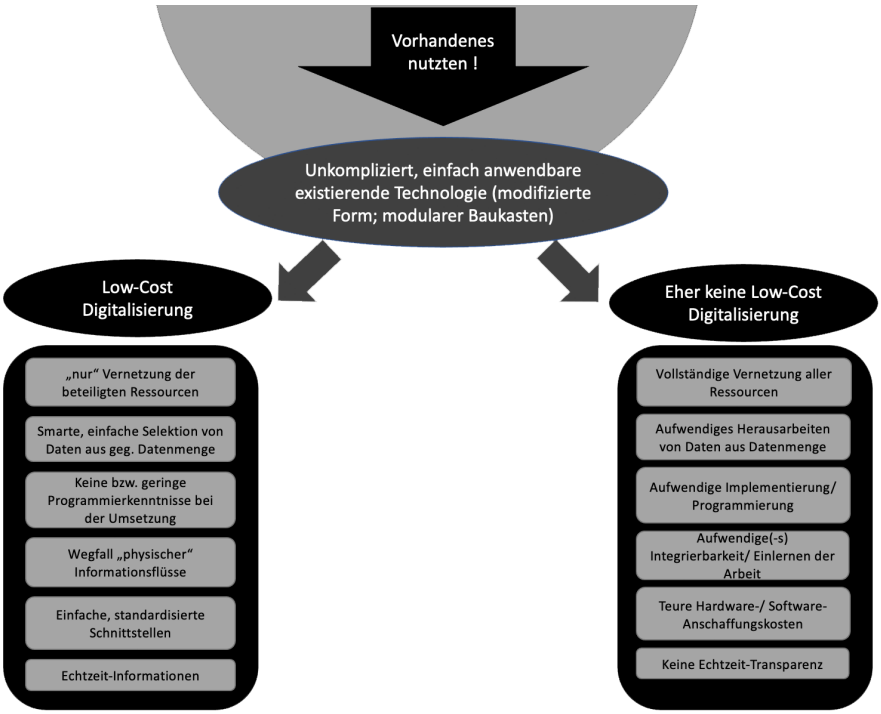


Abbildung 2.8: Low-Cost Digitalisierungsmaßnahmen [29]

2.8 Daten in der Produktion

In den letzten Kapiteln wurde wiederholt auf die Abhängigkeit der Digitalisierungskonzepte von qualitativ hochwertigen Daten eingegangen. Untrennbar ist die vollständige Vernetzung mit der Notwendigkeit große Mengen an Daten zu erfassen und zu verarbeiten verbunden. Diese enormen Datenmengen werden im Kontext der Industrie 4.0 als Big Data bezeichnet. Big Data-Datensätze charakterisiert neben der Datenmenge auch das unterschiedliche Format der generierten Daten und die Geschwindigkeit der Datenerfassung. Für eine möglichst effiziente Datenverarbeitung sollen für erfasste Daten folgende Punkte berücksichtigt werden: [33]

- Generierte Daten sollen einen, auf den Gesamtprozess bezogenen, Nutzen haben.
- Generierte Daten müssen konsistent und vertrauenswürdig sein. Der unerlaubte Zugriff oder eine Modifikation von Prozessdaten ist zu vermeiden.
- Echtzeit Daten mit zeitlich beschränkter Gültigkeit müssen laufend durch aktuelle Daten ersetzt werden.
- Die Richtigkeit der Daten muss durch Validierung der Sensorik stets gegeben sein.
- Generierte Daten sind stets fehlerbehaftet. Die Vollständigkeit, Konsistenz und Qualität der Daten sollte immer kritisch betrachtet werden. [33]

Nicht vermeidbare Messungenauigkeiten, zum Beispiel durch die Linearität von Sensoren, führen oft zu einer unerwünschten Streuung in den generierten Messdaten. Dadurch

steigt mit der Datenmenge auch die Relevanz der Gebiete Datenaufbereitung und Filterung. In der Produktion leistet die Datenanalanalytik einen entscheidenden Beitrag für die erfolgreiche Umsetzung von CPPSs. [22, 33]

2.9 Filter

Jede Messung unterliegt grundsätzlich einem gewissen Messfehler. Die Linearitätsabweichung, auch Systemungenauigkeit genannt, bezeichnet die Differenz zwischen dem tatsächlichen und einem idealen Wert. Diese Linearität führt zu einem Streuband, in dem die Sensordaten verteilt liegen. [34]
Um Rauschen und nicht relevante Anteile von den relevanten Daten beziehungsweise Prozessgrößen trennen zu können (wie in Kapitel 2.8 beschrieben) bedarf es geeigneter Filter. Der für den Anwendungsfall geeignetste Filter hängt stark von den vorliegenden Daten ab, wodurch dessen Auswahl einiges an Erfahrung bedarf. In diesem Kapitel werden in der Arbeit verwendete Filter grundlegend beschrieben.

2.9.1 Gauß-Filter

Der Gauß-Filter ist Teil der sogenannten Tiefpassfilter. Das bedeutet er lässt niedrige (tiefe) Frequenzen passieren und filtert die hochfrequenten Signalteile heraus. Die für jeden Signalfilter charakteristische Filterfunktion beschreibt im Fall des Gauß-Filters eine Gauß'sche Glockenkurve, wodurch er auch seinen Namen bekommt. Mathematisch wird diese Normalverteilung wie folgt beschrieben: [35]

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\mu)^2}{2\sigma^2}) \tag{2.13}$$

wobei μ, und σ die Parameter Mittelwert bzw. Standardabweichung der Normalverteilung darstellen. Bei der Verwendung eines Gauß-Filters wird genau diese Funktion in jeden Punkt der Messwerte gelegt. Danach werden benachbarte Werte gemäß ihres Abstands gewichtet. Der aktuelle Punkt, für den ein gefilterter Wert gesucht wird, liegt genau im Scheitelpunkt der Kurve und hat demnach die höchste Gewichtung. Damit wird klar, dass der Einfluss weiter entfernterer Punkte immer stärker abnimmt und wesentlich von der Wahl des Parameters σ abhängt. Je geringer die Standardabweichung, desto schmäler wird die Kurve (siehe Abbildung 2.9), was in einer geringeren Gewichtung der weiter entfernten Punkte resultiert. Außerdem gilt zu beachten, dass die Randwerte nicht sinnvoll gefiltert werden können, bedingt durch die Tatsache, dass die Glockenkurve (abhängig von gewähltem σ) keine zuordenbaren Werte mehr zur Verfügung hat. [35, 36]

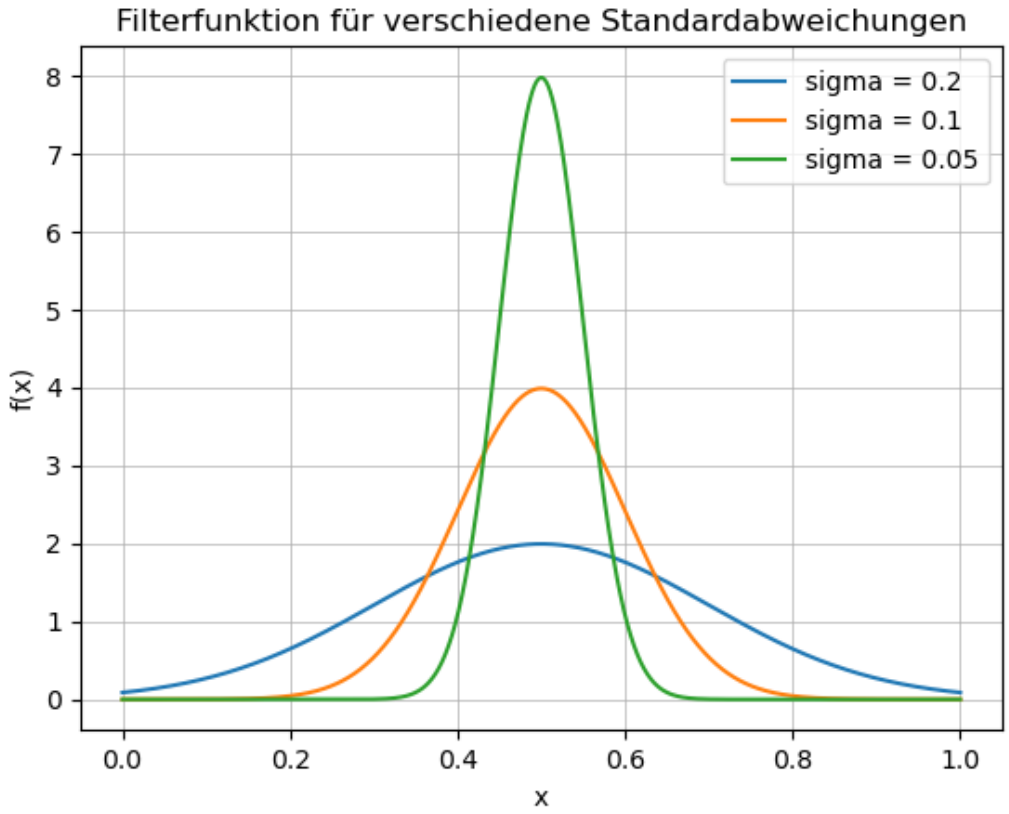


Abbildung 2.9: Filterfunktion des Gauß-Filters für verschiedene Standardabweichungen

2.9.2 Savitzky-Golay-Filter

Der Savitzky-Golay-Filter ist ein von Savitzky und Golay in [37] beschriebener Polynom Filter. Die Funktion beruht auf der Möglichkeit einen Polynom-Fit, ähnlich einem gleitenden Mittelwert, zu betrachten. Vereinfacht beschrieben wird mittels Least-Squares-Abschätzung eine lokale Polynomapproximation gewünschter Ordnung durchgeführt. Aus einer Reihe von Datenpunkten $y(x)$ wird eine Gruppe von $2M+1$ Punkten betrachtet, wobei M das halbe Betrachtungsfenster darstellt. Damit wird ein Polynom N-ter Ordnung [38]

$$p(x) = \sum_{k=0}^{N} a_k x^k \tag{2.14}$$

ermittelt, welches die mittlere quadratische Abweichung

$$\varepsilon = \sum_{x=-M}^{M} (p(x) - y(x))^2 \tag{2.15}$$

minimiert. Die Ermittlung geeigneter Koeffizienten wird in [38] beschrieben. Die verwendete Bereichsgröße kann je nach Anwendung gewählt werden, jedoch ist für die Standardanwendung immer eine ungerade Anzahl an Punkten erforderlich, um einen Schätzwert für den aktuell betrachteten Punkt zu errechnen. Dieses Verfahren wird für jeden Wert

wiederholt, wodurch das Signal stückweise angepasst wird. Konsequenterweise bedeutet das, wenn ein Polynom nullter Ordnung gefittet werden soll, ergibt sich lokal jeweils eine Lineare, was dem Verfahren des gleitenden Mittwelwerts entspricht. Durch die Anpassung der Anzahl gerechneter Punkte und der Ordnung des Polynoms können vor allem Signale mit sehr steilen Flanken vergleichsweise gut abgebildet werden. [38, 39]

Kapitel 3

Programmaufbau

In diesem Kapitel soll ein Überblick über das entwickelte Programm im Gesamten vermittelt werden. Dabei wird vor allem die grundlegende Funktionalität der Applikation erläutert. Ebenso wird die getroffene Gliederung und die daraus resultierende Programmstruktur beschrieben. Um ein Verständnis für die spezifischen Anforderungen der Entwicklung hinsichtlich der Aufgabenstellung Walzwerk aufzubauen, wird einführend die Ist-Situation vor Beginn aufgezeigt.

3.1 Ist-Situation Walzwerk

Am Lehrstuhl für Umformtechnik an der Montanuniversität Leoben wurde im Zuge einer vorangegangen Masterarbeit ein Duowalzwerk, ein Walzwerk mit zwei gleichlaufenden Arbeitswalzen, digitalisiert. Die Zustellung des Walzspalts erfolgt händisch über eine Anstellspindel, welche mittels Handrad bedient wird (siehe Abbildung 3.1).

Kenngröße	Wert
Maximale Walzkraft	300 kN
Maximales Walzmoment	2500 Nm
Walzendurchmesser	Ø 203 mm
Ballenlänge Walzen	220 mm

Tabelle 3.1: Kenngrößen Walzwerk

Die implementierte Sensorik ist in Abbildung 3.2 dargestellt. Die beiden DMS-Kraftmessdosen (KMD) erfassen die auftretenden Walzkräfte, der Linear Variable Differential Transformer (LVDT) den tatsächlichen Walzspalt. Ein Winkelsensor am Handrad misst die jeweilige Position der Zahnräder, welche für die Zustellung des Walzspalts verantwortlich sind. Da diese über die Zustellspindel mit dem Walzspalt korreliert, ist die Messung redundant, weshalb diese Werte hauptsächlich für Kontroll- und Abgleichszwecke dokumentiert werden.

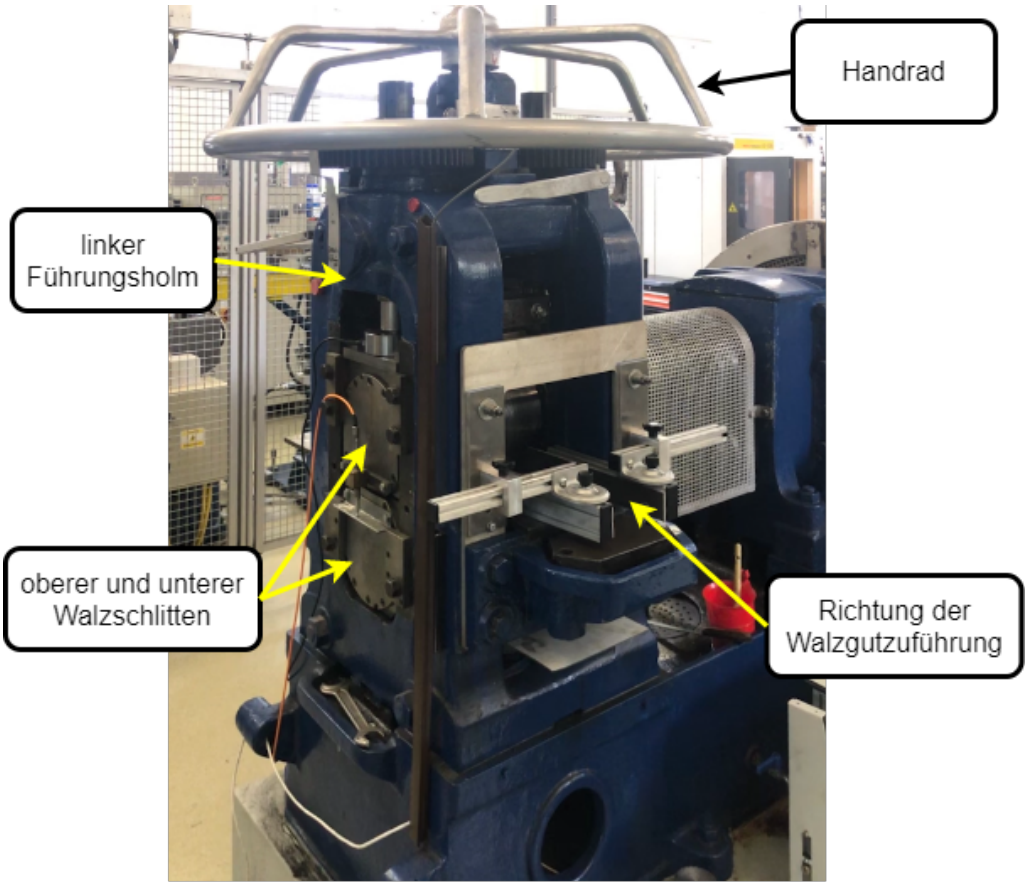


Abbildung 3.1: Walzwerk am Lehrstuhl für Umformtechnik

3.2 Funktionalität

Zur Erweiterung dieses digitalisierten Walzwerks soll eine zusätzliche HMI-Schnittstelle entwickelt und implementiert werden. Die grundsätzliche Funktion des Programms soll zwei Bereiche abdecken.

1. Datenaufbereitung:
   Die während Versuchen am Walzwerk generierten Sensorik-Daten (siehe Kapitel 4.1) sind automatisiert zu importieren und aufzubereiten. Für die weitere Verarbeitung ist es notwendig gefilterte, qualitativ hochwertige Daten zur Verfügung zu stellen.

2. Einbindung GUI:
   Über ein GUI kann ein User einzelne Messungen auswählen. Die Daten werden dann aufbereitet und visualisiert. Für den Bediener relevante Fertigungskenngrößen werden automatisch berechnet und wiedergegeben. Durch diese analytischen Möglichkeiten unmittelbar nach jedem Versuchsdurchlauf, können Maschineneinstellungen flexibel und schnell adaptiert werden, was die Fertigungsqualität erhöht.

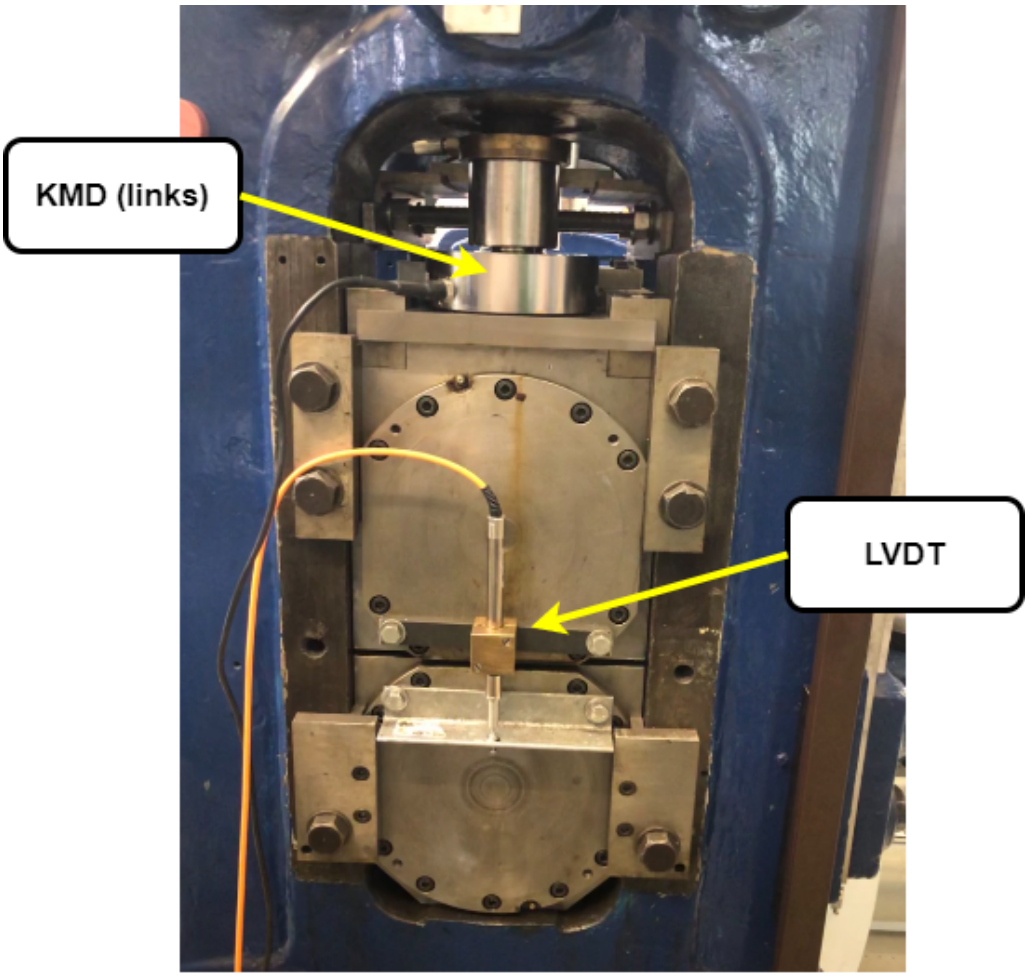


Abbildung 3.2: Implementierte Sensorik am Walzwerk des Lehrstuhls für Umformtechnik

Durch einen möglichst modularen Aufbau sollen einzelne Programmteile einfach adaptierbar sein, um diese auch für andere Anlagen einfach anpassen zu können. Beispielsweise ist der Datenimport aus CSV-Dateien nur in geringem Maß von der Struktur abhängig. Auch bei der Gestaltung von GUIs können, bei entsprechend frühzeitiger Berücksichtigung viele „Bausteine“ mit leichten Anpassungen in neuen Aufgabenstellungen verwendet werden. Der Arbeitsaufwand zur Modernisierung ähnlicher Maschinen am Lehrstuhl kann dadurch erheblich reduziert werden.

3.3 Aufbau

Zur Umsetzung wurde ein Programm in der Programmiersprache Python entwickelt. In den letzten Jahrzehnten haben sich eine Vielzahl an Programmiersprachen etabliert. Nachfolgend wird die Entscheidung mit Python zu entwickeln begründet und die Vorteile der Sprache kurz vorgestellt. Danach wird der Aufbau beziehungsweise die Programmstruktur grundlegend erläutert, bevor die einzelnen Module und Bausteine näher behandelt werden.

3.3.1 Warum Python?

Python ist eine objektorientierte, höhere (auch high-level) Programmiersprache. Das bedeutet vereinfacht gesagt, geschriebener Code wird nicht unmittelbar von Mikroprozessoren verstanden, sondern durch einen Interpreter in Maschinencode übersetzt. Typischerweise ist Python-Code einfach lesbar, was vor allem für Einsteiger als angenehm empfunden wird. Der Python-Interpreter kann auf allen gängigen Betriebssystemen (Windows, Linux und OS X) verwendet werden. Die Tatsache, das Python Open Source und dadurch von jedem frei und uneingeschränkt zu benutzen ist, macht die Verwendung einerseits im universitären Umfeld, andererseits für KMUs sehr interessant. Für wissenschaftliche Nutzung besonders hilfreich ist die umfangreiche Bibliothek. Durch simple Importbefehle können dadurch vielzählige Module die Lösung komplexer Aufgabenstellungen vereinfachen. Ebenso können durch entsprechende Module standardisierte Schnittstellen verwendet werden um externe Geräte einzubinden oder Daten zu übertragen. [40]
Zur Beurteilung der Popularität von Programmiersprachen kann der sogenannte TIOBE Index (siehe Abbildung 3.3) verwendet werden. Dieser Index wird einmal monatlich aktualisiert und reiht sämtliche Programmiersprachen nach der weltweiten Anzahl an ausgebildeten Anwendern, abgehaltenen Kursen und Suchanfragen auf großen Suchmaschinen. Mit Mai 2021 ist Python bereits auf Rang zwei gelistet (Abbildung 3.4). Die in den letzten Jahren klar steigende Beliebtheit ist auf die zunehmende Bedeutung von Data Science Anwendungen und Machine Learning zurückzuführen.

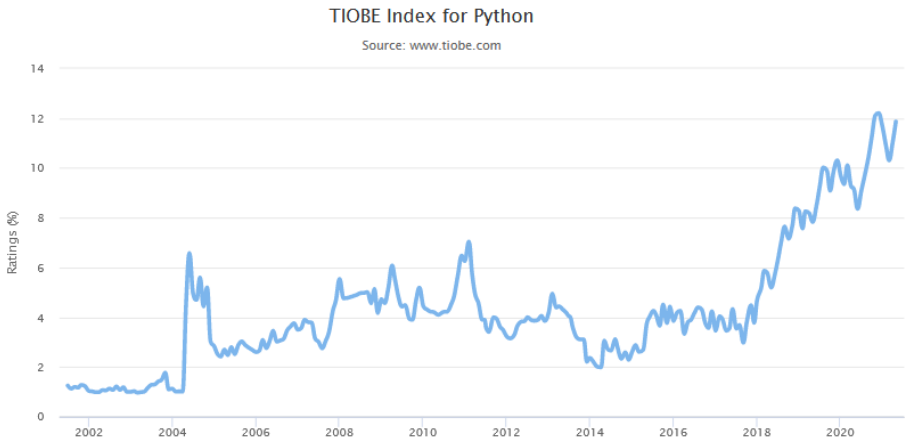


Abbildung 3.3: Entwicklung TIOBE-Index für Python (Mai 2021) [41]

May 2021	May 2020	Change	Programming Language	Ratings	Change
1	1		C	13.38%	-3.68%
2	3	^	Python	11.87%	+2.75%
3	2	v	Java	11.74%	-4.54%
4	4		C++	7.81%	+1.69%
5	5		C#	4.41%	+0.12%
6	6		Visual Basic	4.02%	-0.16%
7	7		JavaScript	2.45%	-0.23%
8	14	^^	Assembly language	2.43%	+1.31%
9	8	v	PHP	1.86%	-0.63%
10	9	v	SQL	1.71%	-0.38%

Abbildung 3.4: TIOBE-Index der „Top 10 Programmiersprachen“ (Mai 2021) [41]

3.3.2 Code-Struktur

In diesem Kapitel wird die Funktionsweise des entwickelten Programms, mittels eines Ablaufdiagramms (Abbildung 3.5) dargestellt. Darin sind alle zur Programmfunktion beitragenden Teile miteinander vernetzt, wodurch die Zusammenhänge und Interaktionen verständlich werden. Zentraler Teil, wie in Abbildung 3.5 ersichtlich, ist das GUI, die Schnittstelle zwischen Mensch und Maschine. Die ausgeführte Datei, laut Konvention meist als main-Datei bezeichnet, startet die Applikation. Als erster Schritt wird das Interface aufgebaut (siehe dazu Kapitel 5). Der weitere Ablauf ist in erster Linie von den Eingaben des Users abhängig. In der main-Datei sind die grafischen Bestandteile des GUI, mit den gewünschten Aktionen verknüpft. Nachdem der Benutzer durch seine Auswahl einen zu betrachtenden Versuch festgelegt hat, werden die Daten vom Server importiert, aufbereitet und visualisiert.

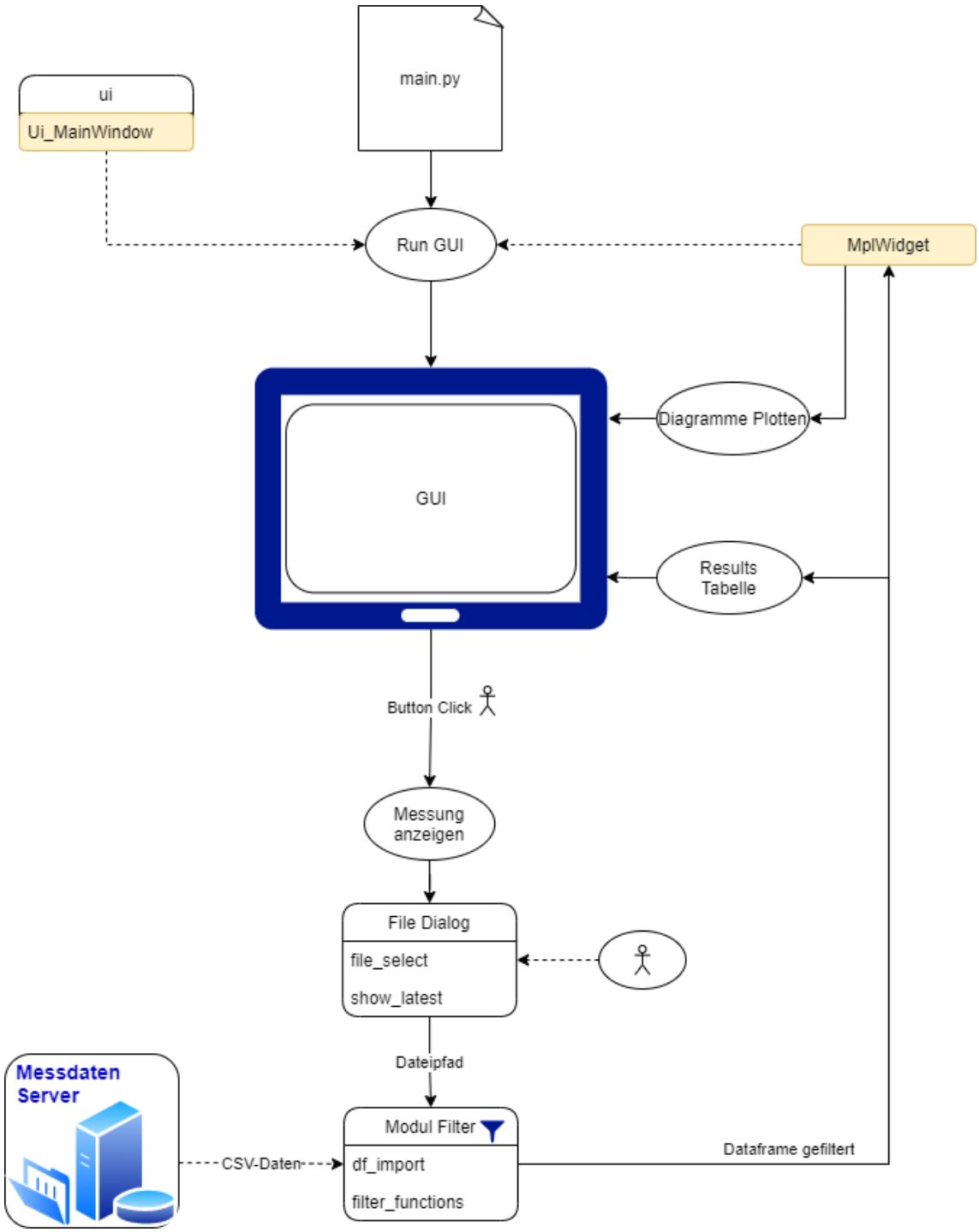


Abbildung 3.5: Diagramm der Programmfunktionalität

Kapitel 4

Datenaufbereitung

In diesem Kapitel wird die Datenaufbereitung und -filterung behandelt. Besonders die Vorteile, welche sich durch die Benutzung geeigneter Datenanalysemodule in Hinblick auf die weitere Verwendung der Daten ergeben, werden aufgezeigt.

4.1 Rohdaten

Die in Kapitel 3.1 beschriebene Sensorik erfasst mit definierter Messfrequenz Messdaten. Die maximale realisierbare Messfrequenz beträgt 1 kHz. Aufgrund der teilweise kurzen Dauer diverser Walzversuche sind generierte Datensätze grundsätzlich mit möglichst hoher Frequenz am aussagekräftigsten. Im Verlauf der Datenanalyse ergab sich allerdings eine technisch maximal realisierbare Messfrequenz. (Kapitel 4.4)
Diese Rohdaten werden für jeweiligen Walzversuch in Comma-Separated Values (CSV)-Dateien geschrieben. Jeder Versuch wird automatisch mit einem generierten, eindeutigen Namen am Server gespeichert. In Abbildung 4.1 sind einige Zeilen eines typischen Datensatzes dargestellt. Der Dateiname links oben besteht immer aus Datum (YYYY-MM-DD) und Versuch mit laufender Versuchsnummer. In der ersten Zeile stehen Bezeichnungen für die jeweiligen Spalten, um die Daten leichter zuordnen zu können.

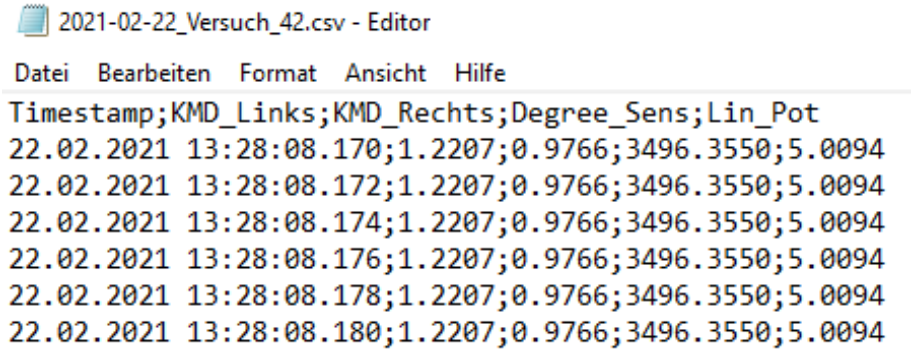


Abbildung 4.1: Ausschnitt einer beispielhaften CSV-Datei mit Messdaten

4.2 Datenimport

Für die gesamte Bearbeitung der Daten wurden verschiedene Python-Programme entwickelt. Im ersten Schritt sollen jene, in CSV-Dateien geschriebene Rohdaten importiert und in einer möglichst übersichtlichen, leicht zugreifbaren Form abgelegt werden.

4.2.1 CSV-Datenimport

Das Einlesen der Daten wurde mithilfe des Python-Moduls Pandas realisiert. Dieses Modul wurde speziell für Datenanalyseanwendungen entwickelt und erleichtert so den Umgang mit großen Datenmengen. Der große Vorteil, welcher sich mit der Nutzung ergibt, sind die verwendbaren Daten-Strukturen „Series“ und „Dataframe“. Ersteres ist eine eindimensionale Anordnung (Array) von Daten. Diese Daten besitzen einen Index entsprechend ihrer Reihenposition und ein Label, auch Achsenüberschrift.
Durch das Aneinanderreihen mehrerer Serien erhält man ein zweidimensionales Array, das Dataframe. Zur Vorstellung kann es mit einer üblichen Tabelle verglichen werden. Darin können verschiedenste Datentypen gespeichert werden (Objekte, Int64, Float64, Bool, Datetime64...).

Im Regelfall besitzt jede CSV-Datei eine Überschriftenzeile (siehe Abbildung 4.1), diese können dann im Dataframe automatisch auch als Label benutzt werden. Der Importbefehl (Listing 4.1) liest die entsprechende CSV-Datei ein und baut daraus ein Dataframe auf.

Listing 4.1: Pandas CSV-Datenimport

```
import pandas as pd

df = pd.read_csv(filepath, delimiter=';', error_bad_lines= False,
    parse_dates = ['Timestamp'], date_parser = lambda x: pd.
    to_datetime(x, errors = 'coerce'))
```

df - DataFrame

Index	Timestamp	KMD_Links	KMD_Rechts	Degree_Sens	Lin_Pot
0	2021-02-22 19:28:48.073000	1.1719	0.8301	4210.2	1.7741
1	2021-02-22 19:28:48.079000	1.1719	0.8301	4210.2	1.7741
2	2021-02-22 19:28:48.083000	1.1719	0.8301	4210.2	1.7741
3	2021-02-22 19:28:48.100000	1.2695	0.8301	4207.48	1.7741

Abbildung 4.2: Erstelltes Dataframe nach dem CSV-Import mittels Pandas

Die Bedeutung der angeführten Input-Argumente für den Datenimport werden in Kapitel 4.2.2 näher erläutert. In Abbildung 4.2 sind beispielhaft einige Zeilen eines erstellten Dataframe abgebildet. Die Spaltenbezeichnungen werden aus der ersten Zeile der CSV-Datei automatisch übernommen. Jede Zeile der CSV-Datei, also jeder Messzeitpunkt der Sensoren ist durch den sogenannten „Timestamp“ eindeutig definiert und zuordenbar. Um diese Informationen entsprechend nutzen und für die Aufbereitung verwenden zu können, wird die Spalte ['Timestamp'] als Spalte mit Zeitdaten gesondert ausgewiesen.

4.2.2 Exception-Handling

Um den Datenimport stabil gestalten zu können, ist es wichtig im Vorfeld möglichst viele Szenarien, welche zu Fehlermeldungen führen können, zu identifizieren. Vor allem sogenannte „Bad Lines“ in den Rohdaten dürfen für das Einlesen kein Abbruchkriterium darstellen. Bei dem Import vieler hunderter Datensätze zeigten sich vor allem drei typische Fälle:

- Elementanzahl der Reihe $\neq$ Spaltenanzahl Dataframe (=Bad Line)
- Timestamp kann nicht richtig übersetzt werden
- CSV-Datei besitzt keine Überschriftenzeile

Die ersten beiden Fälle können beide direkt, durch Verwenden entsprechender Input-Argumente, abgefangen werden. Durch Setzen des Parameters `error_bad_lines` auf True, werden solche Zeilen übersprungen, eine kurze Infomeldung wird ausgegeben und die Datei kann ohne Abbruch weiter eingelesen werden. Ähnlich funktioniert auch der `date_parser`. Sollte ein Timestamp von der Parse-Funktion (`pandas.to_datetime()`) nicht übersetzt werden können, wird diese Zeile für den Import ignoriert ohne einen Programmstop zu verursachen. Durch schlichtes Ignorieren fehlerhafter Zeilen gehen zwar Daten verloren, allerdings stehen diese quantitativ bei 500 bis 1000 Messungen pro Sekunde in keinem Verhältnis zum Vorteil eines stabilen Import-Prozesses.
Ist keine Überschriftenzeile in der CSV-Datei enthalten, werden die Spalten standardmäßig aufsteigend nummeriert. Allerdings wird eine Spalte `['Timestamp']` gesucht um den Zeitstempel entsprechend zu übersetzen. Existiert keine solche Spalte, bricht das Programm mit einem sogenannten Value Error ab. Um in solchen Fällen, vor allem bei bekannten möglichen Fehlermeldungen, ein weiteres Ausführen des Programms zu ermöglichen, kann in Python eine sogenannte Ausnahmebehandlung definiert werden. Dabei wird im try-Block jener Code ausgeführt, welche ein Ausnahmerisiko darstellt. Sollte eine Ausnahme (ein Fehler), wie beispielsweise das Fehlen der Spaltenlabels auftreten, kann in einem except-block ein Ausnahmeverhalten für eben diesen Fehler definiert werden.

Listing 4.2: try-except Block Datenimport

```
try:
  df = pd.read_csv(file, delimiter=';', error_bad_lines=False,
      parse_dates=['Timestamp'], date_parser=lambda x: pd.
      to_datetime(x, errors="coerce"))

except ValueError:
  colnames = ['Timestamp', 'KMD_Links', 'KMD_Rechts', 'Degree_Sens
      ', 'Lin_Pot']
  print('No column names defined in file.\n Changed to: ',
      colnames)

  df = pd.read_csv(file, delimiter=';', header=0, names=colnames,
      error_bad_lines=False, parse_dates=['Timestamp'],
      date_parser=lambda x: pd.to_datetime(x, errors="coerce"))
```

Wenn also ein Value Error auftritt, bricht das Programm nicht ab, sondern springt in die definierte Ausnahmebehandlung (Listing 4.2). Dort wird die CSV-Datei ebenfalls eingelesen, allerdings werden die Spalten gemäß einer Liste mit den gewünschten Bezeichnungen benannt, wodurch der Konflikt vermieden wird. Durch beschriebene Maßnahmen kann ein automatisierter Import sichergestellt werden, was anhand von 2600 getesteten Datensätzen belegt wurde.

4.3 Datenfilterung

Nach dem vorhergegangenen Kapitel 4.2 liegen die aufzubereitenden Daten in einem Pandas Dataframe. Allerdings ist ein erfolgreicher Datenimport noch lange keine Garantie für qualitativ hochwertige Datensätze. Ziel der weiteren Aufbereitung muss es sein, durch eine Reihe an Datenfilterungsschritten einerseits sämtliche Redundanzen und Daten ohne echten Informationsgehalt auszusortieren, andererseits in Hinblick auf die weitere Verwendung derart vorzubereiten, dass benötigte Information leicht und komfortabel zugänglich ist.

Allgemeine Datenaufbereitungsschritte:

1. Daten nach Timestamp sortieren
2. Duplikate entfernen
3. Not a Number(NaN)-Werte entfernen
4. Spalten mit ausschließlich Nullwerten löschen
5. Indizes neu vergeben

Spezielle Datenaufbereitungsschritte Daten Walzwerk:

6. Spalte `['elapsed time']` einfügen
7. Spalte `[F_ges']` einfügen

Die beiden eingefügten zusätzlichen Spalten machen das Arbeiten mit dem Dataframe in weiterer Folge komfortabler. Da beide Kraftmessdosen jeweils nur einen Teil der Gesamtkraft abbilden, wird die Summe dieser Messwerte als F_ges gesondert abgelegt. Der bereits beschriebene Zeitstempel, also der individuelle Zeitpunkt jedes Messpunktes, gibt den absoluten Zeitpunkt wieder. Für die weitere Bearbeitung, vor allem für spätere Visualisierungen, ist eine neu eingeführte Zeitachse, welche mit dem Messbeginn bei Null startet und den relativen Abstand zum Startpunkt jedes Datensatzes in Sekunden wiedergibt, wesentlich informativer.
Gezeigte Schritte zur Datenaufbereitung, im wesentlichen das automatisierte Aussortieren unvollständiger oder fehlerhafter Datensätze, ist grundlegender Bestandteil von Data Science-Aufgabenstellungen. Da dafür, bis auf die individuell erstellten Spalten, die technische Bedeutung dieser Daten irrelevant ist, können Teile eines durchdachten, modular aufgebauten Programms mit wenigen Änderungen für ähnliche Konzepte herangezogen werden.

4.4 Optimale Messfrequenz

In Kapitel 4.3 wurde die neu eingeführte, bei Null startende Zeitachse beschreiben. Mit diesen Zeitdaten wurden im Zuge des Datenstudiums erste Diagramme wie etwa Walzkraft über Zeit oder Walzspalt über Zeit erstellt. Dabei fiel auf, dass im Bereich des eigentlichen Walzvorgangs (maximale Veränderung der Werte innerhalb kurzer Zeit) untypisch lineares Verhalten sichtbar wurde. Daraufhin wurden die Messdaten, beziehungsweise deren exakten Zeitdaten genauer untersucht.
Die ersten Vorversuche wurden mit maximaler Frequenz, von 1 kHz gemessen. Gerade für Versuche im universitären Umfeld wo, im Gegensatz zu industriellen Walzprozessen, sehr viele verschiedene Versuchsreihen mit repräsentativen, eher klein dimensioniertem Walzgut durchgeführt werden, ist das Bestreben dabei möglichst viele Datenpunkte zu generieren logisch. Bei idealen Bedingungen sollte demnach die zeitliche Differenz zwischen einzelnen Messungen gleichbleibend 1 ms betragen. Allerdings konnten in den Datensätze Abstände von bis zu 0.6 Sekunden festgestellt werden. Um diese Unregelmäßigkeiten zu quantifizieren wurde jeweils die Zeitdifferenz zur vorangegangenen Messung über die Gesamtzeit aufgetragen (siehe Abbildung 4.3). Bei 7000 importierten Messpunkten kann hier 12 mal ein unverhältnismäßig langer Zeitunterschied gezeigt werden. Durch diese Darstellung ist die Periodizität der signifikant größeren Abstände gut erkennbar.

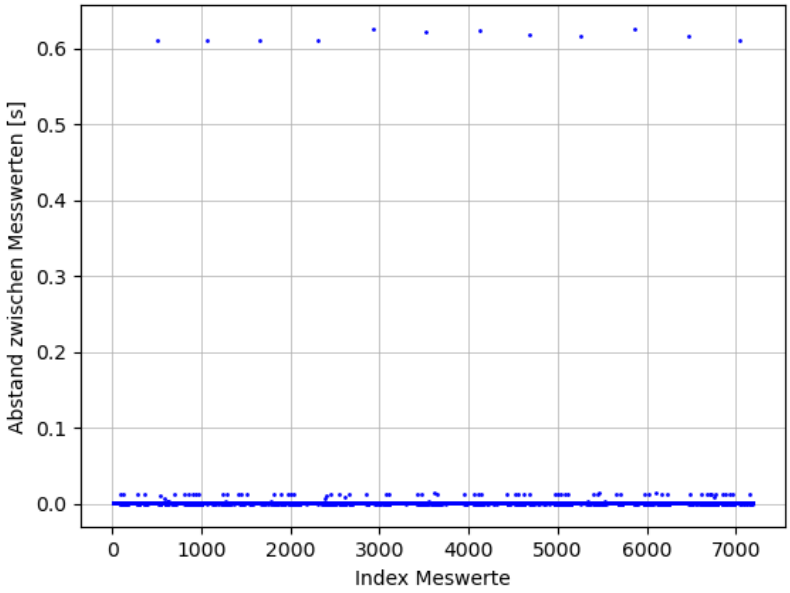


Abbildung 4.3: Zeitdifferenzen zwischen Messwerten, Messfrequenz 1 kHz

Aufgrund dessen kann eine eventuell zu hoch gewählte Messfrequenz vermutet werden. Pufferspeicher dienen im Allgemeinen dazu Daten zwischenzuspeichern. Sie kompensieren Schwankungen in der Signalübertragung und -umwandlung. Wenn durch eine zu hoch gewählte Frequenz diese Pufferspeicher voll sind, gehen nachfolgend Daten verloren. Genau solange, bis wieder ausreichend Pufferkapazitäten vorhanden sind. Dadurch ergeben sich periodisch lange Verzögerungen. Als naheliegende Abhilfemaßnahme wurde die Messfrequenz reduziert. Dadurch geht zwar, begründet durch die niedrigere Abtastrate, Informationsdichte verloren, allerdings wird der gesamte Prozess ohne unverhältnismäßig lange Messverzögerungen abgebildet. In der Praxis bewährte sich eine Frequenz von 500 Hz, mit welcher sämtliche Messverzögerungen vermieden werden können.

4.5 Glättung Walzspaltdaten

In Kapitel 2.9 wurden bereits grundlegend eingesetzte Filter beschrieben. Die Messdaten des LVDT, welche die aktuelle Walzspalthöhe abbilden, benötigen zusätzlich die Anwendung eines Filters, um später übersichtliche, einfach abzulesende Verläufe zu erstellen. Durch die Messungenauigkeit des Sensors, werden auch im Standbetrieb des Walzwerks, ohne aktuellen Walzprozess, leicht schwankende Werte aufgenommen. Durch den insgesamt kleineren Messbereich, verglichen mit der Kraftaufnahme, werden diese Schwankungen allerdings stärker abgebildet.
Nachdem jeder Versuch mit unterschiedlichen Maschinenparametern und Walzgut durchgeführt wurde, unterscheiden sich auch die Eingangsdaten für die Filterung entsprechend.

Ziel ist es also nicht Filterparameter für einen beispielhaften Versuch bestmöglich zu optimieren, sondern ein gutes Ergebnis beim erstmaligen Filtern zuvor nicht studierter Datensätze zu erreichen. Das Ergebnis einer Signalfilterung bei der Verwendung eines Gauß-Filters ist von der Wahl einer geeigneten Standardabweichung der Filterfunktion abhängig. In Abbildung 4.4 wurden verschiedene σ-Werte verwendet und gegenübergestellt.

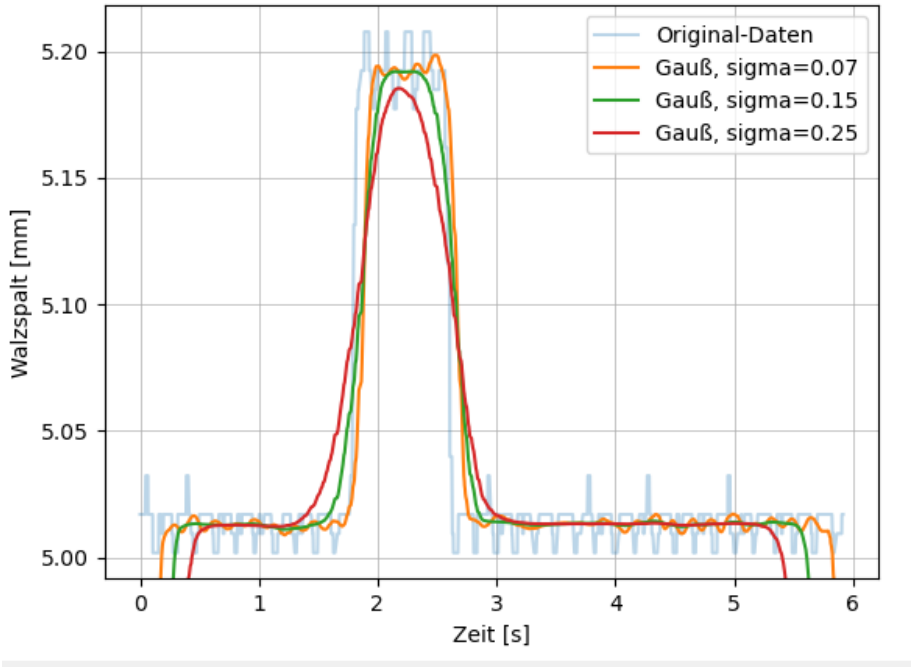


Abbildung 4.4: Vergleich Gauß-Filter für verschiedene σ-Werte

Erwartungsgemäß kann die steile Flanke umso besser abgebildet werden, je kleiner die Standardabweichung der Gauß-Filterfunktion gewählt wird. Allerdings wird im Bereich vor dem eigentlichen Walzvorgang das Filtersignal noch von dem Rauschen beeinflusst, was vermieden werden soll. Für eine automatisierte Auswertung des vom Maschinenbediener eingestellten Walzspalts ist eine möglichst lineare Abbildung der Walzspalthöhe vor Beginn des eigentlichen Walzvorgangs anzustreben. Bei einer zu breit gewählten Gauß-Verteilung (große σ-Werte) wird dieses Kriterium zwar erfüllt, doch schnellen Signaländerungen kann nur langsam gefolgt werden. Dadurch sind sowohl die Flanken als auch der Bereich des Umformvorgangs nicht ausreichend gut abgebildet. Der für das Beispiel Walzwerk gewählte Wert muss also eine Kompromisslösung zwischen guter Flankenabbildung und zufriedenstellender Filterung der Rauschanteile in konstanten Bereichen darstellen.

In Abbildung 4.5 sind verschiedene Filtereinstellungen für den Polynomfilter Savitzky-Golay verglichen. Durch die Vielzahl an Datenpunkten ist durch eine Erhöhung der Polynomordnung kein merkbarer Unterschied feststellbar, weshalb alle abgebildeten Varian-

ten mit Polynomen zweiter Ordnung gefittet wurden. Der Parameter n gibt die Anzahl der berücksichtigten Messpunkte für die lokale Approximation an (Fensterlänge). Eine weitere Verbesserung wurde durch die Verwendung einer zweiten Filterstufe erreicht. Dabei wurden die Schätzwerte aus erstem Filterschritt mit gleichen Parametern ein weiteres Mal approximiert. Der Vorteil dieses Filters ist die sehr gute Abbildung von steilen Flanken. In Abbildung 4.6 werden die beiden Filtertypen verglichen. Die Filterfunktion des Gauß-

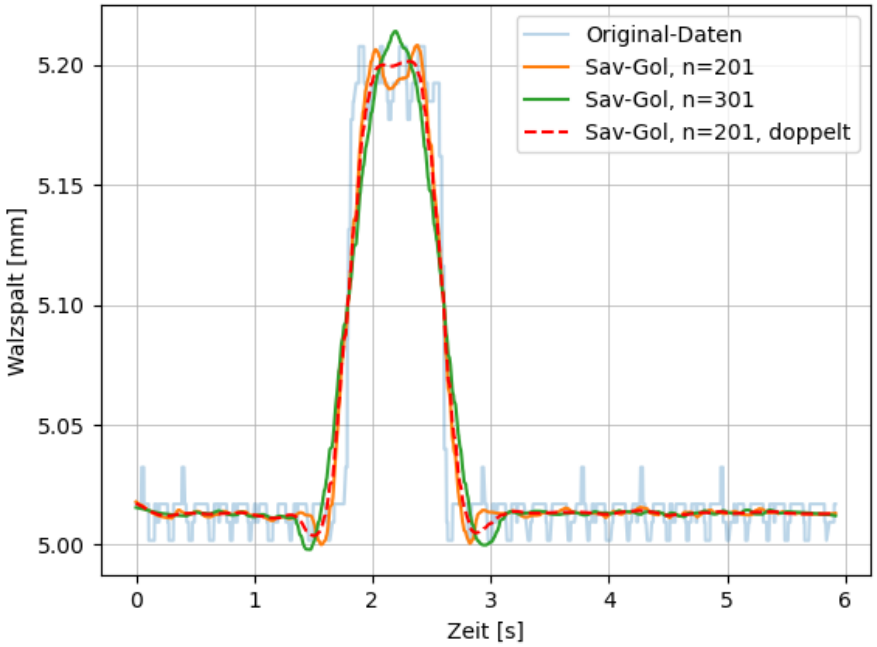


Abbildung 4.5: Vergleich Savitzky-Golay-Filter, mit variierender Fernsterlänge

Filters wurde mit einem $\sigma = 0.15$ abgebildet, der Savitzky-Golay-Filter in der doppelt gefilterten Variante mit einer Fensterbreite von 201 Messpunkten und einem Polynom-Fit zweiter Ordnung. Der große Nachteil der Gauß-Filterung ist der Informationsverlust an den Rändern, welcher durch die Art der Filterung verursacht wird (siehe Kapitel 2.9.1). Der Savitzky-Golay-Filter liefert auch für die Randbereiche brauchbare, gefilterte Werte. Außerdem bildet er die Flanken besser ab, als der Gauß-Filter. Entscheidend für die Verwendung eines Gauß-Filters war schlussendlich die gleichbleibend gute Abbildung konstanter Bereiche, was für die Auswertung hohe Relevanz besitzt und eine realistische Abbildung der Auffederung (keine Überschätzung des maximal gemessenen Walzspalts) ermöglicht.

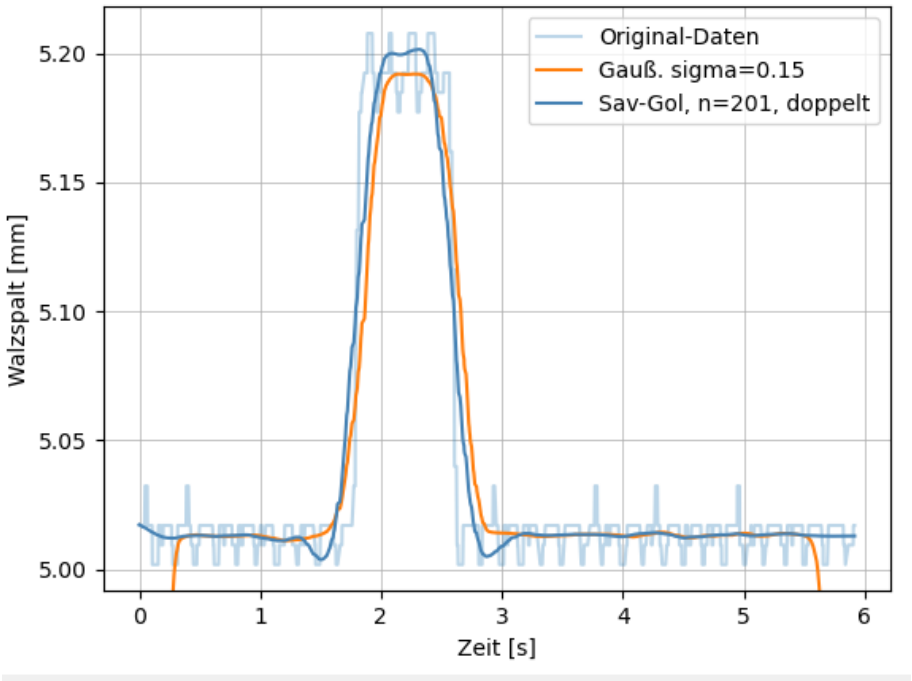


Abbildung 4.6: Vergleich Gauß-Filter zu Savitzky-Golay-Filter, optimierte Parameter

Kapitel 5

Graphical User Interface

Im Kapitel 2.6 wurde die Notwendigkeit für die Schaffung von Schnittstellen zwischen Mensch und Maschine bereits aufgezeigt. Allgemein ist die Möglichkeit der Human-Machine Interaction (HMI) grundlegender Bestandsteil jedes CPS. In folgenden Kapiteln werden, neben einer grundlegenden Charakterisierung von GUIs, die Anforderungen bezüglich der Entwicklung eines Interfaces für das beispielhafte Walzwerk am Lehrstuhl für Umformtechnik diskutiert.

5.1 Anforderungen Interface

Als Startpunkt für jegliche Überlegungen muss der zukünftige Einsatzzweck und die zu erfüllende Aufgabe abgegrenzt werden. In Kapitel 2.6 wurden bereits verschiedene Möglichkeiten der Interaktion aufgezeigt. Da durch die tägliche Verwendung von Computern, Laptops und Smartphones jeder an GUIs gewöhnt ist und zur Darstellung von Diagrammen, basierend auf den Messdaten, eine grafische Schnittstelle bestens geeignet ist, fiel die Entscheidung ein GUI zu entwickeln. Da der Trend ganz klar weg von Maus und Tastatur hin zu modernen Eingabemethoden wie Touchscreen geht, wird das UI als Touch User Interface (TUI) ausgelegt. Die Vorteile sind einfache, hilfsmittelfreie Bedienbarkeit. Im Designprozess muss darauf durchaus Rücksicht genommen werden. Beispielsweise haben Touchsysteme so gut wie nie eine Tastatur zur Dateneingabe. Müssen oftmals Daten mittels Tastatur eingegeben werden, kann das mit Kompromisslösungen wie einer Bildschirmtastatur langfristig unkomfortabel sein. Außerdem ist zu beachten, dass im Gegensatz zur Computermaus mittels Touchscreen nur Links-Klick Eingaben möglich sind. Da das TUI, abgesehen von der Eingabemöglichkeit, sich nicht wesentlich vom GUI unterscheidet, wird auch das Touch Interface oftmals als GUI bezeichnet.
Auch die zu erwartenden Anwenderbedürfnisse müssen beim Design berücksichtigt werden. Angefangen von der Sprache und Beschriftung einzelner GUI-Elemente, bis hin zum Einsatzort. Ein Interface, welches in der Werkstätte bei einem Fertigungsprozess zur An-

wendung kommt, unterliegt anderen Erwartungen hinsichtlich intuitiver Bedienbarkeit als vergleichsweise eine Benutzeroberfläche für eine Präzessionsmessvorrichtung. Daraus ergeben sich im wesentlichen folgende Anforderungspunkte:

- Design möglichst einfach und übersichtlich
- Intuitive Benutzung ohne spezielles Training
- Bedienbar ohne Tastatur
- Bedienbar per Touchscreen ohne Eingabetool (Computermaus, Stift)

5.2 GUI Design mit Qt

Für das Erstellen von GUIs gibt es viele Tools, die alle individuelle Vor- und Nachteile besitzen. Auch in Verbindung mit Python gibt es einige Open Source-Lösungen. Durch die vielfältigen Aufgabenstellungen im GUI-Design hat sich bislang auch keine Umgebung als Standard etabliert. Die drei am häufigsten verwendeten Frameworks zur GUI-Entwicklung in Python sind Tkinter, wxPython und Qt. [42]
Die Entscheidung mit Qt zu entwickeln stützt sich auf die hohe Programmierflexibilität. Die Verbindung zwischen GUI-Elementen (Kapitel 5.2.1) basiert auf dem Signale-Slots-Konzept, wodurch ein variabler Umgang mit auftretenden Ereignissen möglich ist. Außerdem werden zwei unterschiedliche Skripten verwendet. Eines, welches für den Aufbau des GUIs zuständig ist und ein zweites, mit zugehörigem Python-Code.
Qt selbst, ist ein Open Source GUI-Toolkit und Entwickler-Framework. Damit können plattformübergreifende GUI-Applikationen entwickelt werden. Die Qt Bibliothek benutzt allerdings die Programmiersprache C++ , weshalb die Python-Anbindung PyQt verwendet wird. Auch wenn dadurch in Python entwickelt werden kann, wird bei der Verwendung von Qt-Objekten und Methoden die grundlegende Kenntnis des C++ Syntax vorausgesetzt. [43]

5.2.1 GUI-Elemente

In diesem Kapitel wird der Aufbau und die Funktionsweise von GUIs beschrieben, wodurch auch die Unterschiede zu Text basierenden Programmen verdeutlicht werden. Das Hauptelement jeder Anwendung ist die sogenannte QApplication-Klasse. Jede Anwendung benötigt, um zu funktionieren, genau ein Objekt dieser Klasse. Dieses Objekt besitzt den Event Loop (Abbildung 5.1). Solange die Applikation ausgeführt wird, werden alle Events in dieser Schleife abgearbeitet. Jede Interaktion (Mausbewegung, Mausklick, Buttonklick) verursacht ein Event und wartet in der Event Warteschlange auf die Bearbeitung. Damit der User interagieren kann, werden verschiedene Widgets verwendet. Einige

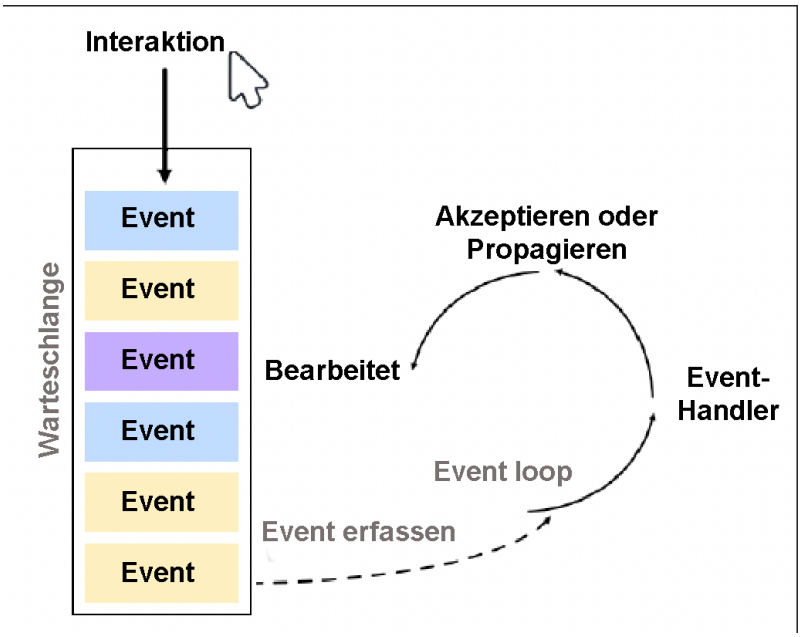


Abbildung 5.1: Schematische Darstellung eines Qt Event Loops [43]

Beispiele solcher Widgets sind Labels zur Textausgabe, Inputfelder für Benutzereingaben, Pushbuttons oder Schieberegler. Diese Widgets senden, je nach Funktion, bestimmte Signale, welche im Programm entsprechend für gewünschte Aktionen verwendet werden können. Für grafische Darstellungen gibt es außerdem sogenannte Canvas-Widgets, welche unter anderem für Diagrammdarstellungen benutzt werden können. [43]

5.2.2 Qt Creator

In Kapitel 5.2 wurde bereits kurz erwähnt, dass durch die Verwendung von PyQt der grafische Aufbau des User Interfaces in einem eigenen Skript beschrieben werden kann. Ein großer Vorteil, da dieses strukturell und inhaltlich stets gleich aufgebaut wird und automatisiert erstellt werden kann. Qt Creator ist eine Entwicklungsumgebung, in der die grafische Oberfläche des GUI, ähnlich wie in reinen Design-Programmen erstellt werden kann. Widgets können beliebig mittels Drag and Drop in Layouts platziert werden. Außerdem sind Eigenschaften wie Größe, Name, Label und Verhalten direkt definierbar. Qt Creator generiert daraus den Code, welcher diese Oberfläche mit gesetzten Eigenschaften beschreibt, allerdings ebenfalls in der Programmiersprache C++. Um dieses Tool auch in einem Python-Projekt nutzen zu können, wird ein Modul namens QtPy benötigt, welches den Ordnerinhalt (also die automatisch generierten C++ Dateien) in Python-Code übersetzt. Eine manuell auszuführende build-Datei generiert ein Python-Skript mit gleichwertigem Ergebnis. Diese Datei kann nun vom ausführenden Python-Skript zur Erstellung des GUI verwendet werden. Wichtig ist es diese automatisch generierte Datei nicht zu editieren, da sämtliche Änderungen beim erneuten Kompilieren verloren gehen.

5.3 GUI Walzwerk

In diesem Kapitel wird die grafische Benutzeroberfläche, welche speziell als Interface für die Aufgabenstellung Walzwerk entwickelt wurde, vorgestellt. Besonderen Wert wurde auch darauf gelegt, viele Programmteile so modular wie möglich zu gestalten. Dies ermöglicht eine einfache Adaption für geplante zukünftige Digitalisierungsvorhaben, ohne ein komplett neues GUI designen zu müssen.

5.3.1 Funktionsumfang

In Abbildung 5.2 ist das entwickelte GUI nach Start der Applikation dargestellt. Um Walzversuche hinsichtlich ihrer Qualität zu bewerten, ist die Darstellung drei verschiedener Diagramme notwendig. Der zeitliche Verlauf von Gesamtwalzkraft, deren Aufteilung auf die linke und rechte Kraftmessdose, sowie der zeitliche Verlauf des Walzspalts, welcher Aufschluss über die maximale Auffederung während des Walzprozesses gibt.
Aufgrund dessen, dass der Großteil aller Python-Entwickler Diagramme mithilfe des Pakets Matplotlib erstellt, wurde dieses in die PyQt-Umgebung eingebunden. Das Verwenden und Steuern mehrerer unabhängiger Plots in einem gemeinsamen Fenster kann mithilfe der zur Verfügung gestellten Modul-Bibliothek nicht hinreichend realisiert werden, weshalb dazu eine benutzerdefinierte Widget-Klasse entwickelt wurde (siehe Kapitel 5.3.2). Zur Bedienung stehen dem User vier Buttons zur Verfügung. Um einen Walzver-

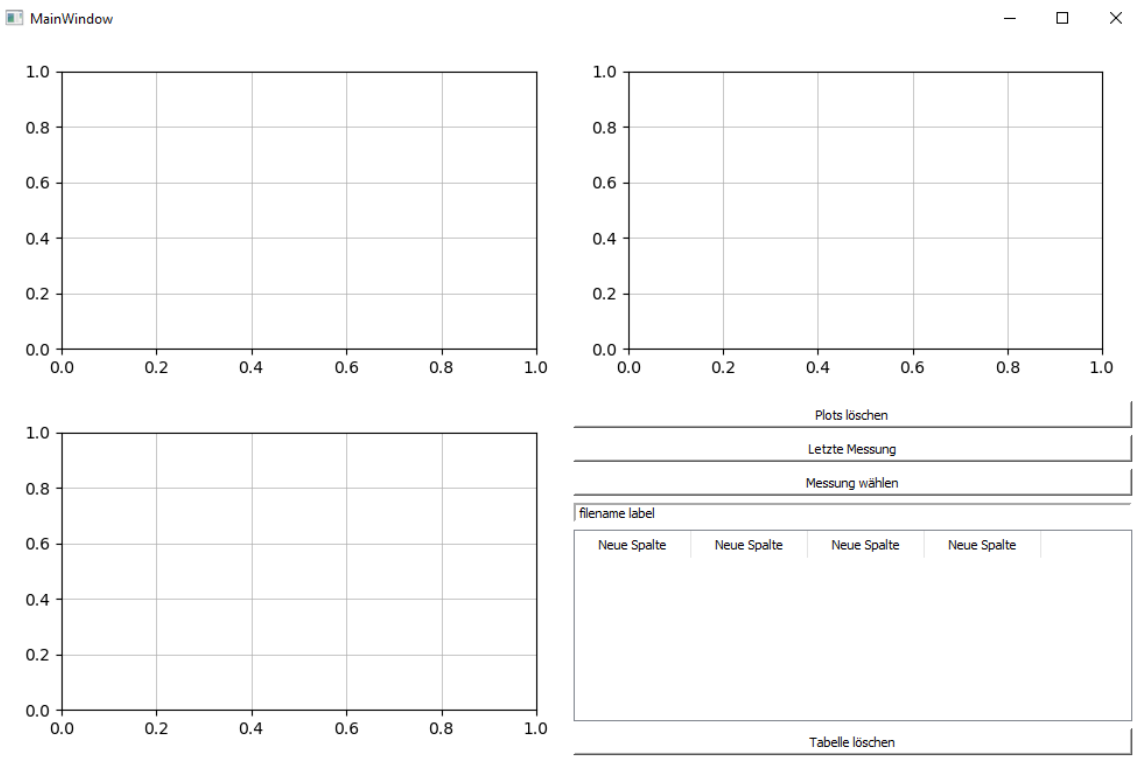


Abbildung 5.2: GUI zu Applikationsbeginn

such zur Auswertung auszuwählen, können zwei Wege verwendet werden. Der Button „Letzte Messung“ findet am Server die jüngste Messdatendatei und verwendet diesen Datensatz zur Aufbereitung. Möchte der User eine vergangene Messung betrachten, öffnet der Button „Messung wählen“ einen File-Dialog, verwendet wird der Default File Dialog des jeweiligen Betriebssystems, in dem eine beliebige CSV-Datei selektiert werden kann. Mittels „Plots löschen“, werden die Diagramme der aktuell dargestellten Messung in den abgebildeten Ausgangszustand zurückversetzt.
Im Bereich rechts unten befindet sich eine Ergebnistabelle. Darin werden für den Maschinenbediener relevante Kenngrößen, zum Beispiel die maximale Walzkraft oder der eingestellte Walzspalt zu Beginn der Messung ausgegeben. In Abbildung 5.3 ist diese Tabelle in befülltem Zustand zu sehen. Die Werte zum aktuell dargestellten Versuch werden immer in der ersten Zeile ausgegeben und farblich hinterlegt. Wird ein neuer Versuch geladen, werden die vorherigen Daten in die jeweils nächste Zeile überschrieben. Dadurch können beispielsweise die Werte von Versuchen ähnlicher Voraussetzungen miteinander verglichen werden. Der darunterliegende Button „Tabelle löschen“ entfernt den gesamten Inhalt der Ergebnistabelle. Darüber befindet sich zusätzlich ein Label. In diesem wird der Dateipfad der aktuell dargestellten Messung angezeigt.

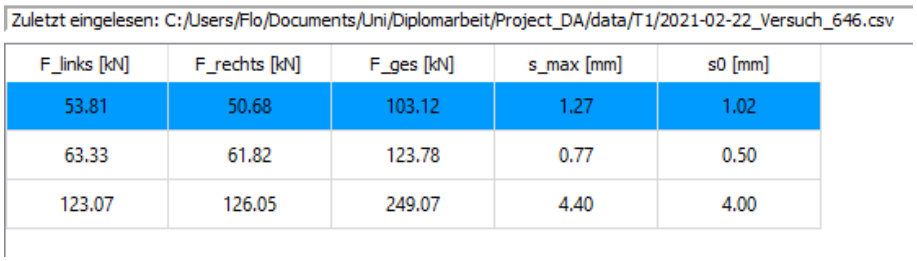


Abbildung 5.3: Befüllte Ergebnistabelle mit Versuchskennwerten

5.3.2 Custom Widgets

Zur gleichzeitigen Darstellung drei verschiedener Plots, welche separat steuerbar sein sollen, war die Konstruktion einer benutzerdefinierten Widgetklasse notwendig. Um während des GUI-Designprozesses Widgets zu verwenden, welche nicht in der Standardbibliothek enthalten sind, muss vorerst ein vorhandenes Widget eingesetzt werden. Besonders gut eignet sich dazu das Object QWidget, welches ohne primäre Funktionalität in gewünschter Größe eingebaut wird und erstmal als Platzhalter fungiert. Danach kann in den Eigenschaften dieses Platzhalteobjekts ein benutzerdefiniertes Widget verknüpft werden. In Abbildung 5.4 ist der zugehörige Dialog abgebildet. Darin ist der Klassenname des Custom Widgets und der Dateiname, in welcher die entsprechende Klasse definiert ist, anzugeben. Danach ändert sich der Objekttyp und das Widget mit dem gewünschten Namen kann verwendet werden. In der angesprochenen Klassendefinition ist eine init-Methode

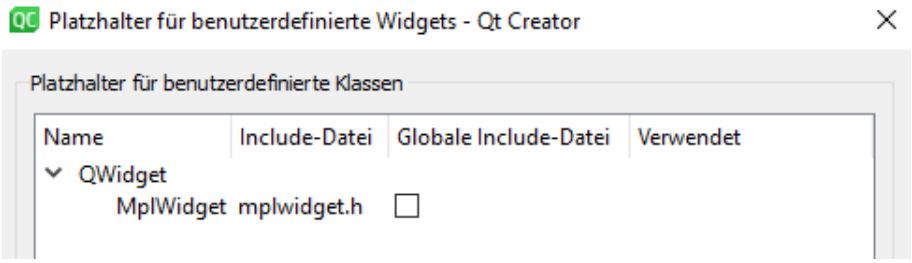


Abbildung 5.4: Definition eines Platzhalters für benutzerdefinierte Widgets, Qt Creator

notwendig (siehe Listing 5.1). Im Zuge des Setup-Prozesses für das GUI, gemäß des von PyQt generierten Codes, müssen auch die benutzerdefinierten Widgets erstellt werden. Die init-Methode wird mit jedem erstellten Objekt ausgeführt und bestimmt damit die Anfangseigenschaften. Die Klasse erweitert die Elternklasse FigureCanvasQTAgg, welche im wesentlichen die Einbindung des Moduls Matplotlib in grafische Benutzeroberflächen ermöglicht. Die init-Methode der Oberklasse erstellt ein Canvas Widget zur Darstellung von Matplotlib-Abbildungen.

Listing 5.1: Initialisierungs-Methode des benutzerdefinierten MplWidgets

```
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg

class MplWidget(FigureCanvasQTAgg):

  def __init__(self, parent):
    self.fig = plt.figure(tight_layout=True)
    super().__init__(self.fig)
    self.setParent(parent)
    self.ax = plt.axes()
    self.ax.grid(b=True, linewidth=.5)
```

Jedes MplWidget-Objekt verfügt also nach dem Erstellen über eine Matplotlib-Figure, mit gleichwertiger Funktionalität und Handhabung wie Abbildungen in gebräuchlicher Skriptumgebung.

Kapitel 6

Implementierung

Nach der programmiertechnischen Umsetzung, wurde das entwickelte Interface in das CPPS Walzwerk integriert. In diesem Kapitel werden zuerst Anforderungen und mögliche Umsetzungskonzepte diskutiert. Abschließend wird die realisierte Implementierung näher erläutert und die Inbetriebnahme reflektiert.

6.1 Anforderung

Um das entwickelte GUI bestmöglich und sinnvoll im universitären Betrieb einzubinden, gilt es verschiedene Implementierungsmöglichkeiten in Betracht zu ziehen. Da die Werkstätte, bezüglich Größe und Maschinenausrüstung mit kleineren Betrieben vergleichbar ist, soll das Konzept beispielhaft für die Chancen der Digitalisierung auch im kleinen Maßstab sein. Daraus ergeben sich zwei grundlegende Interessen. Einerseits ist der Kostenrahmen für notwendige Anschaffungen so zu wählen, dass die Motivation vorhandene Maschinen zu modernisieren nicht durch eingeschränkte finanzielle Möglichkeiten vermindert wird. Durch die Gegenüberstellung von fertigungstechnischem Nutzen zu vertretbarem Ressourceneinsatz kann ein Machbarkeitsnachweis für Low-Cost-Lösungen erbracht werden.
Andererseits soll die Implementierung für entsprechend qualifiziertes Personal betriebsintern durchführbar sein. Damit wird der notwendige Mitarbeitereinsatz dahingehend aufgewertet, dass aufgebautes Know-how sowohl für den laufenden Betrieb, als auch für die Umsetzung zukünftiger Projekte zweckhaft ist. Mit Schwerpunkt auf diese beiden Punkte sind die in Kapitel 2.7 festgelegten Kriterien für Low-Cost Automation in KMUs hinreichend erfüllt.

6.2 Konzeptübersicht

Zuvor genannte Anforderungen (Kapitel 6.1) erfüllen im wesentlichen verschiedene Lösungen. In den nächsten Kapiteln werden mögliche Varianten vorgestellt und bewertet. Dabei ist besonders der Implementierungsaufwand, der Kostenaspekt für Anschaffungen sowie die Flexibilität und technische Eignung zu bewerten.

6.2.1 Konzept Mikrocontroller

Die wohl kostengünstigste Variante ist die Verwendung eines Mikrocontrollers, wie dem Arduino-Board Uno. Diese sind bereits im unteren zweistelligen Euro Bereich erhältlich und enthalten typischerweise ein Eingabe/Ausgabe - Board, einen Mikrocontroller und eine integrierte plattformunabhängige Entwicklungsumgebung. Diese basiert auf dem Java Programmierstandard und enthält Bibliotheken, wodurch auch in C oder C++ programmiert werden kann. Allerdings ist ein Arduino nur in der Lage bereits kompilierten Code auszuführen. Er kann also nicht völlig alleine betrieben werden und ist standardmäßig nicht netzwerkfähig. Für die Aufgabenstellung Walzwerk ist die Einbindung ins Lehrstuhlnetzwerk vorgesehen, wodurch einige zusätzliche Module notwendig wären.
Vor allem aufgrund entscheidender Nachteile, wie der aufwendigen Nachrüstung, der nicht standardmäßigen Netzwerkfähigkeit und eingeschränkter leistungstechnischer Möglichkeiten (unter anderem keinen integrierten Arbeitsspeicher) entspricht die Verwendung eines Mikrocontrollers, trotz gegebenen Preisvorteils nicht den Anforderungen. [44]

6.2.2 Konzept Einplatinencomputer

Einplatinencomputer (auch single board computer SBC) bekommen ihren Namen durch die Unterbringung aller relevanter Elektronikkomponenten auf einer Leiterplatte. Der bekannteste Vertreter, Raspberry Pi (Abbildung 6.1), ist ein völlig funktionsfähiger Minicomputer, der mit dem eigenen Betriebssystem Raspian betrieben wird. Neue Modelle verfügen standardmäßig über einen HDMI-Ausgang, mehrere USB-Anschlüsse und volle Netzwerkfähigkeit (Ethernet, WLAN und Bluetooth). Die Kosten sind mit bis zu maximal 40 Euro ebenfalls sehr niedrig. Der entscheidende Vorteil gegenüber dem Arduino ist neben dem eingebauten Arbeitsspeicher und wesentlich höherer Prozessorleistung, dass Programme auch direkt entwickelt, geändert und kompiliert werden können. Außerdem ist die notwendige Anbindung an einen Bildschirm über die HDMI-Schnittstelle wesentlich einfacher. Das Entwickeln mit Python funktioniert durch den vorinstallierten Python-Compiler und eine eigene Raspian Python IDE problemlos. Einzig das Betriebssystem ist für Linux-Einsteiger anfangs herausfordernd. [44]

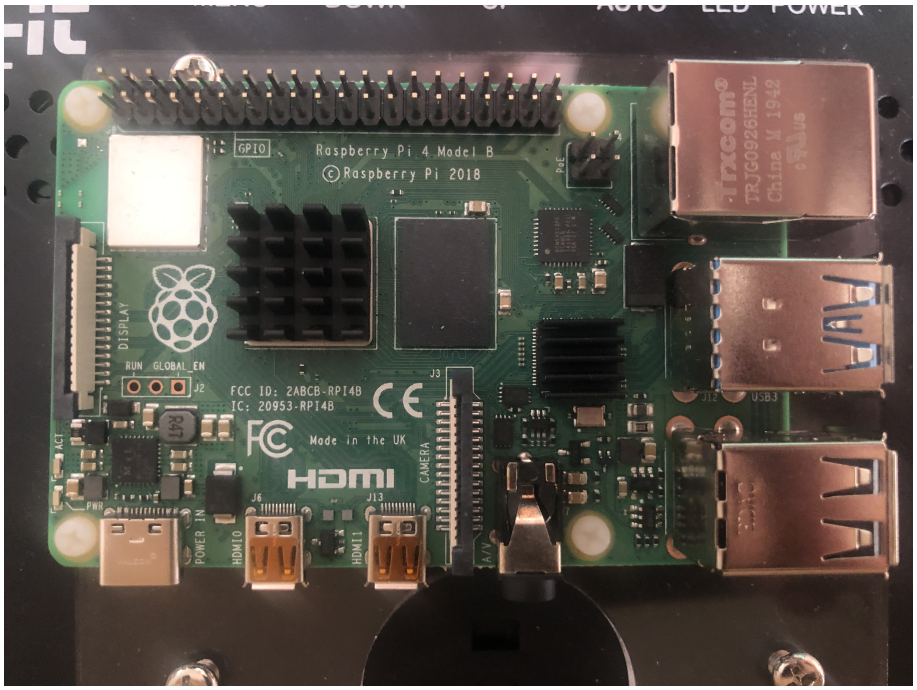


Abbildung 6.1: Einplatinencomputer Raspberry Pi 4

6.2.3 Bewertung

Nachdem das Konzept Mikrocontroller den Anforderungen nicht genügte, wird es in die Entscheidungsfindung nicht weiter eingebunden. In Kapitel 5.1 wurde bereits die gewünschte Bedienbarkeit per Touchscreen erwähnt. Die etablierte alternative zur Variante Einplatinencomputer stellt ein üblicher Tabletcomputer dar, weshalb diese beiden Lösungen in Tabelle 6.1 verglichen werden.

	Raspberry + Touchscreen	Tablet
VT	+ Komponenten kostengünstig + Einfache Netzwerkintegration + Programmierfreiheit (Linux) + Vielfältige Anschlüsse + Lerneffekt (Engineering Education)	+ Mobil, dadurch flexibler + „Vertrautes“ Betriebssystem + Sofort betriebsbereit
NT	- Zubehör nicht inklusive - Kabelgebunden - Inbetriebnahme aufwendiger	- Wesentlich teurer - Eingeschränkte Programmierfreiheit - Meist keine Standardanschlüsse

Tabelle 6.1: Vergleich Konzept Einplatinencomputer mit Tabletcomputer

Da für das Walzwerk eine stationäre Lösung (fixe Wandmontage) realisierbar ist und das in Python entwickelte GUI problemlos und ohne notwendige Adaptionen verwendet werden kann, wurde die Entscheidung für die Anschaffung eines Raspberry Pi Systems mit Touschscreen-Bildschirm getroffen. Durch den Kostenvorteil gegenüber einem leistungsstarken Tabletcomputer entspricht die gewählte Variante ebenso dem Anspruch einer LCIA-Implementierung.

6.3 Inbetriebnahme

Für die Realisierung der gewählten Variante wurden neben dem Einplatinencomputer auch ein den Robustheitsanforderungen einer Werkstätte entsprechender Touchscreen-Bildschirm mit 10 Zoll Bildschirmdiagonale ausgewählt. In Tabelle 6.2 sind sämtliche benötigten Komponenten in einer detaillierten Auflistung dargestellt. Nachdem das gesamte Programm vollständig in der Open Source-Sprache Python geschrieben wurde, sind die Gesamtkosten sehr moderat und in erster Linie von den internen Mitarbeiterstunden für Entwicklung und Implementierung abhängig.

Komponente	Kosten
Raspberry Pi Model 4	41 €
10,1” Touchscreen Display für SBC (Fa. Joy-IT)	156 €
Datenkabel USB-A auf USB-C	11€
MircoSD Karte 16 GB	6 €
Gesamtpreis	214 €

Tabelle 6.2: Überblick Anschaffungskosten

Bevor mit dem HMI gearbeitet werden kann, muss der Raspberry aufgesetzt werden. Dafür muss die MicroSD Karte in FAT32 formatiert und das Raspberry Image installiert werden. Nach dem Anschluss aller notwendigen Kabel an den Touchscreen, wurde das entwickelte Pythonprojekt überspielt. Da Python-Code plattformübergreifend verwendet werden kann, ist keine weitere Adaption notwendig. Über die Linux-Kommandozeile können notwendige Python-Module, welche in der Standardbibliothek nicht vorhanden sind, installiert werden. Nach der Integration in das lehrstuhlinterne Netzwerk ist seitens des Programms der Zugriff auf Server-Messdaten möglich. In Abbildung 6.2 sieht man das entwickelte Interface, wie es am Touchscreen für den Bediener angezeigt wird. Die Bedienung erfolgt im Regelfall ausschließlich per Touchscreen, weshalb alle GUI-Elemente entsprechend groß designt wurden. Abbildung 7.2 zeigt einen User bei der Touch-Bedienung der laufenden Applikation.

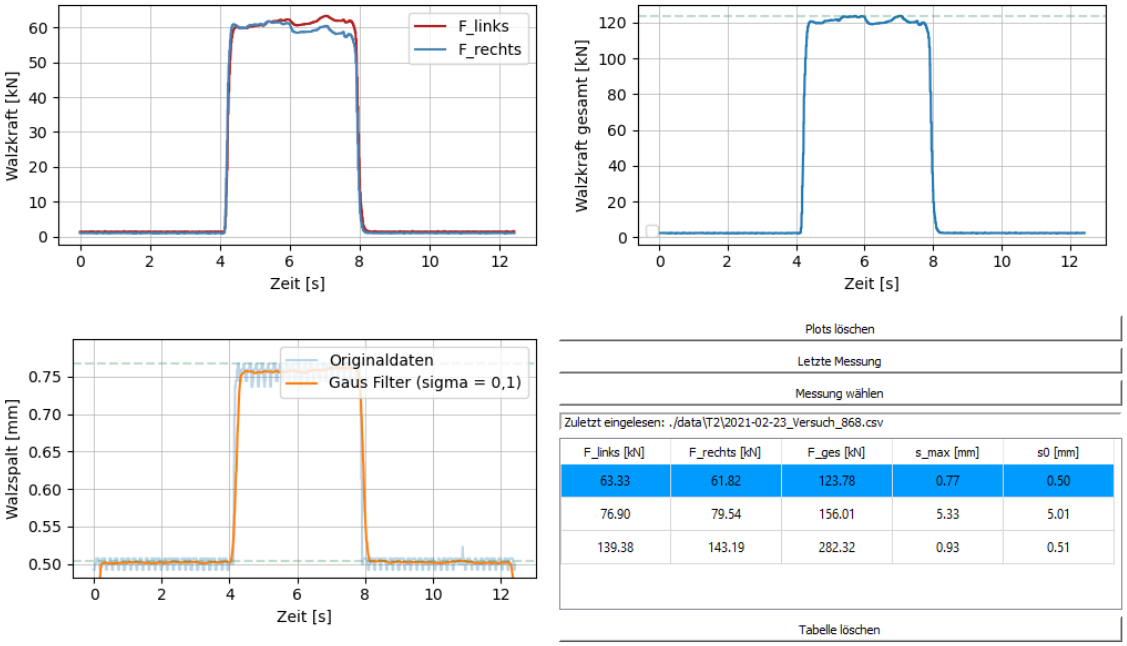


Abbildung 6.2: Entwickeltes GUI im Betrieb

Kapitel 7

Ergebnisse

Die im Zuge dieser Arbeit entwickelte HMI-Schnittstelle ermöglicht es dem Maschinenbediener zeitnah Walzversuche zu beurteilen. Dadurch konnte die Qualität und Effizienz von im universitären Umfeld durchgeführten Versuchsreihen gesteigert werden. Durch die automatisierte Datenaufbereitung und die dadurch geschaffenen qualitativ hochwertigen Daten ist die weitere Analyse, auch außerhalb der Verwendung für das HMI möglich. Das GUI ermöglicht die Auswahl, Visualisierung und Bewertung von Versuchen, was dem Maschinenbediener die Möglichkeit gibt, auf systematische Fehler, mögliche Abweichungen und veränderte Maschinenparameter zu reagieren.
Durch die Verwendung der Open Source-Programmiersprache Python und die Wahl möglichst kostengünstiger, aber geeigneter Hardwarekomponenten konnten die Gesamtumsetzungskosten für die Implementierung gering gehalten werden. Dadurch ist die HMI-Schnittstelle eine beispielhafte Umsetzung einer LCIA-Maßnahme. Durch die Implementierung mittels eines Touchscreen-Bildschirms kann auf weitere Eingabehilfen verzichtet werden, wodurch eine einfach bedienbare, robuste und kompakte Lösung erzielt werden konnte.
Abbildung 7.1 zeigt die Rückseite des Bildschirms, mit dem betriebsbereiten und verkabelten Rapsberry Pi 4, vor der Montage. In Abbildung 7.2 ist die ins lehrstuhlinterne Netzwerk eingebunde Schnittstelle im laufenden Betrieb abgebildet.
Der modulare Programmaufbau ermöglicht die Verwendung allgemeiner, nicht speziell auf die Aufgabenstellung Walzwerk abgestimmter Programmteile in angedachten zukünftigen Modernisierungen. Durch entsprechende Adaptionen und die bei der Implementierung gesammelten Erfahrungen kann von einer wesentlich kürzen Entwicklungs- und Implementierungszeit bei ähnlichen folgenden Aufgabenstellungen ausgegangen werden, wodurch die aufzuwendenden Gesamtinvestitionskosten zusätzlich gesenkt werden können.

Abbildung 7.1: Betriebsbereiter Bildschirm mit montiertem, verkabeltem Raspberry Pi

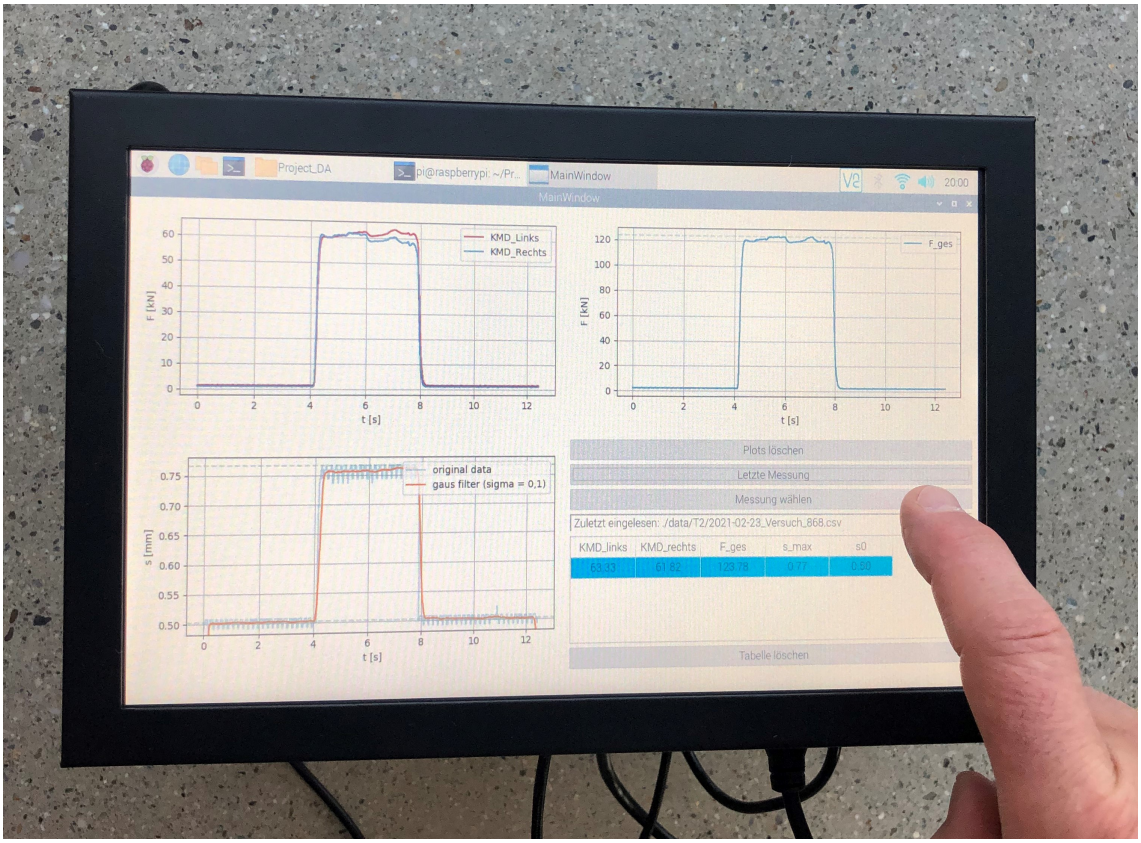


Abbildung 7.2: Touch-Bedienung des GUI

Kapitel 8

Zusammenfassung und Ausblick

Durch vorangegangene Digitalisierung in Verbindung mit dem implementierten HMI entspricht das Walzwerk am Lehrstuhl für Umformtechnik den Standards eines CPPS im Zeitalter Industrie 4.0. Diese Arbeit soll durch die geringen notwendigen Investitionskosten aufzeigen, dass auch KMUs sinnvolle Low-Cost-Maßnahmen mit moderatem Ressourcenaufwand bewerkstelligen können. Die geschaffenen Möglichkeiten für den Maschinenbediener, durch aufbereitete und visualisierte Daten direkte Rückmeldung aus durchgeführten Versuchen zu erhalten und gegebenenfalls darauf zu reagieren, sind in Anbetracht der dafür notwendigen Investitionen immens.
Im Bereich der Interaktion zwischen Mensch und Maschine werden ständig neue Technologien entwickelt. In Zukunft wird es möglicherweise nicht mehr notwendig sein über ein physisches Interface zu kommunizieren. Die hauptsächlich aus der Unterhaltungsindustrie bekannten Virtuell Reality(VR)-Brillen sind durchaus für den Einsatz in der Produktion denkbar. Beispielsweise könnten Prozessparameter oder Diagramme direkt im Sichtfeld des Maschinenbedieners eingeblendet werden, ähnlich dem Head-up-Displays in Autos. Mittels VR könnten auch für den Produktionsprozess kritische Szenarien simuliert werden, um den Maschinenbediener im Trainingsumfeld auf unerwartetes Maschinenverhalten vorzubereiten und angebrachte Reaktionen zu erlernen. [45]
Auch konventionelle Eingabemethoden könnten durch Stimmerkennung, Augenbewegungen oder Gestik ersetzt werden. Noch weiter in die Zukunft gedacht sind BCIs Gegenstand aktueller Forschungen. Damit könnte, durch Messung und Auswertung der Gehirnaktivität die Notwendigkeit einer aktiven Eingabe komplett entfallen. [28]
Auch ist mit einer vermehrten Einbindung von Robotern in der Produktion auszugehen. Damit verbunden müssen künftig Möglichkeiten geschaffen werden, die es dem Menschen ermöglichen mit den zunehmend leistungfähigeren Robotern zu interagieren. Demnach bedarf es Schnittstellen zwischen Menschen, selbstständig arbeitenden, mobilen Robotern und Computern bzw. Maschinen. Auch für die Weiterentwicklung bestehender Human-Robot Collaboration (HRC) Konzepte ist die Integration von VR-Systemen geplant. [1, 46]

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literaturverzeichnis

[1] Alfons Botthof and Ernst Hartmann, editors. *Zukunft der Arbeit in Industrie 4.0*. Springer Vieweg, Berlin, 2015.

[2] Ulrich Sendler. *Industrie 4.0*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[3] Jiafu Wan, Hu Cai, and Keliang Zhou. Industrie 4.0: Enabling technologies. In *Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things*, pages 135–140. IEEE, 2015.

[4] Manfred Broy, editor. *Cyber-Physical Systems: Innovation durch softwareintensive eingebettete Systeme ; [acatech Symposium*. acatech DISKUTIERT. Springer, Berlin, 2010.

[5] Engineering National Academies of Sciences and Medicine. *A 21st century cyber-physical systems education*. The National Academies Press, Washington, DC, 2016.

[6] Saeid Nahavandi. Industry 5.0—a human-centric solution. *Sustainability*, 11(16), 2019.

[7] Kadir Alpaslan Demir, Gözde Döven, and Bülent Sezen. Industry 5.0 and human-robot co-working. *Procedia Computer Science*, 158:688–695, 2019.

[8] Harald Kugler. *Umformtechnik: Umformen metallischer Konstruktionswerkstoffe ; mit 20 Tabellen, 273 Fragen*. Fachbuchverl. Leipzig im Carl Hanser Verl., München, 2009.

[9] Eckart Doege and Bernd-Arno Behrens. *Handbuch Umformtechnik: Grundlagen, Technologien, Maschinen ; mit 55 Tabellen*. VDI. Springer, Berlin, 2007.

[10] Karl-Heinrich Grote, Beate Bender, and Dietmar Göhlich. *Dubbel*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018.

[11] Hartmut Hoffmann, Günter Spur, and Reimund Neugebauer. *Handbuch Umformen*. Edition Handbuch der Fertigungstechnik. Carl Hanser Fachbuchverlag, s.l., [2., vollst. neu bearb. aufl.] edition, 2012.

[12] Bruno Buchmayr. *Werkstoff- und Produktionstechnik mit Mathcad: Modellierung und Simulation in Anwendungsbeispielen ; [mit 160 Beispielen*. Engineering online library. Springer, Berlin, 2002.

[13] A. Herbert Fritz and Günter Schulze, editors. *Fertigungstechnik*. Springer-Lehrbuch. Springer Vieweg, Berlin, 10., neu bearb. aufl. edition, 2012.

[14] Marc-Fabian Körner, Dennis Bauer, Robert Keller, Martin Rösch, Andreas Schlereth, Peter Simon, Thomas Bauernhansl, Gilbert Fridgen, and Gunther Reinhart. Extending the automation pyramid for industrial demand response. *Procedia CIRP*, 81:998–1003, 2019.

[15] Tobias Meudt, Malte Pohl, and Joachim Metternich. Die automatisierungspyramide - ein literaturüberblick. 2017.

[16] David Siepmann. Industrie 4.0 - fünf zentrale paradigmen. In *Einführung und Umsetzung von Industrie 4.0 : Grundlagen, Vorgehensmodell und Use Cases aus der Praxis*, pages 35–46. Springer Gabler, Berlin, 2016.

[17] F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl. Industry 4.0 – an introduction in the phenomenon. *IFAC-PapersOnLine*, 49(25):8–12, 2016.

[18] National Science Foundation - NSF. Cyber-physical systems (cps) (nsf21551). 2016.

[19] Radhakinsan Baheti and Helen Gill. Cyber-physical systems. *The impact of control technology*, 12(1):161–166, 2011.

[20] Malte Brettel, Niklas Friederichsen, Michael Keller, and Marius Rosenberg. How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective. *World Academy of Science, Engineering and Technology, International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, 8:37–44, 2014.

[21] László Monostori. Cyber-physical production systems: Roots, expectations and r&d challenges. *Procedia CIRP*, 17:9–13, 2014.

[22] Pengcheng Fang, Jianjun Yang, Lianyu Zheng, Ray Y. Zhong, and Yuchen Jiang. Data analytics-enable production visibility for cyber-physical production systems. *Journal of Manufacturing Systems*, 57:242–253, 2020.

[23] Concetta Semeraro, Mario Lezoche, Hervé Panetto, and Michele Dassisti. Digital twin paradigm: A systematic literature review. *Computers in Industry*, 130:103469, 2021.

[24] VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik. Simulation und digitaler zwilling im anlagenlebenszyklus. *VDI-Statusreport*, 2020(02), 2020.

[25] Kai Ding, Felix T.S. Chan, Xudong Zhang, Guanghui Zhou, and Fuqiang Zhang. Defining a digital twin-based cyber-physical production system for autonomous manufacturing in smart shop floors. *International Journal of Production Research*, 57(20):6315–6334, 2019.

[26] Stefan Boschert, Christoph Heinrich, and Roland Rosen. Next generation digital twin. In *Proc. tmce*, pages 209–218, 2018.

[27] Mirza Abdur Razzaq, Muhammad Ali, Kashif Hussain, and Saleem Ullah. A survey on user interfaces for interaction with human and machines. *International Journal of Advanced Computer Science and Applications*, 8, 2017.

[28] Memmott Tab, Koçanaoğulları Aziz, Lawhead Matthew, Klee Daniel, Dudy Shiran, Fried-Oken Melanie, and Oken Barry. Bcipy: brain–computer interface software in python. *Brain-Computer Interfaces*, 0(0):1–18, 2021.

[29] Hannes Hönig and Björn Lorenz. Low-cost-digitalisierung in der produktion. *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 112(12):895–898, 2017.

[30] Hitoshi Takeda. *LCIA-Low Cost Intelligent Automation: Produktivitätsvorteile durch Einfachautomatisierung*. MI Wirtschaftsbuch, 2004.

[31] Hitoshi Takeda. *Das synchrone Produktionssystem: Just-in-time für das ganze Unternehmen*. Vahlen, 2014.

[32] R. Kazala and P. Straczynski. The most important open technologies for design of cost efficient automation systems. *IFAC-PapersOnLine*, 52(25):391–396, 2019.

[33] Radu F. Babiceanu and Remzi Seker. Big data and virtualization for manufacturing cyber-physical systems: A survey of the current status and future outlook. *Computers in Industry*, 81:128–137, 2016.

[34] Ekbert Hering and Gert Schönfelder, editors. *Sensoren in Wissenschaft und Technik: Funktionsweise und Einsatzgebiete*. Praxis. Vieweg + Teubner, Wiesbaden, 1. aufl. edition, 2012.

[35] Klaus Stein. Filterung: Gauß-filter. = https://stein-sw.de/filterung-gauss-filter/. [Online, zuletzt abgerufen am 01.06.2021].

[36] Hans-Joachim Mittag. *Statistik: Eine interaktive Einführung*. Springer-Lehrbuch. Springer Berlin, Berlin, 2., wesentl. überarb. aufl. edition, 2012.

[37] Abraham. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[38] Ronald W. Schafer. What is a savitzky-golay filter? [lecture notes]. *IEEE Signal Processing Magazine*, 28(4):111–117, 2011.

[39] William H. Press and Saul A. Teukolsky. Savitzky-golay smoothing filters. *Computers in Physics*, 4(6):669, 1990.

[40] Vidya M Ayer, Sheila Miguez, and Brian H Toby. Why scientists should learn to program in python. *Powder Diffraction*, 29(S2):S48–D64, 2014.

[41] index | tiobe - the software quality company. `https://www.tiobe.com/tiobe-index/`. [Online; zuletzt abgerufen am 13.05.2021].

[42] Allen B. Downey. *Think Python*. O'Reilly, Sebastopol CA, 1. ed. edition, 2012.

[43] Martin Fitzpatrick. Create gui applications with python & qt5: The hands-on guide to making apps with python. 2020.

[44] Arduino vs. raspberry pi. `https://www.ionos.at/digitalguide/server/knowhow/arduino-vs-raspberry-pi/`. [Online; zuletzt abgerufen am 24.05.2021].

[45] Athirah Syamimi, Yiwei Gong, and Ryan Liew. Vr industrial applications—a singapore perspective. *Virtual Reality & Intelligent Hardware*, 2(5):409–420, 2020.

[46] Andrea de Giorgio, Mario Romero, Mauro Onori, and Lihui Wang. Human-machine collaboration in virtual reality for adaptive production engineering. *Procedia Manufacturing*, 11:1279–1287, 2017.

[47] Raspberry pi 4 model b specifications – raspberry pi. `https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-product-brief.pdf`. [Online, zuletzt abgerufen am 29.05.2021].

[48] reichelt elektronik GmbH &amp and Team, Co. KG Internet. Debo lcd 10 hdmi - entwicklerboards - display lcd-touch, 10", hdmi. `https://www.reichelt.at/index.html?ACTION=7&LA=3&OPEN=0&INDEX=0&FILENAME=C300%2FRB-LCD10-2-DATENBLATT.pdf`, note =.

Anhang A

Python-Code

Listing A.1: main.py

```
import filter
import sys
from qtpy import QtWidgets,QtGui, QtCore
from ui.mainwindow import Ui_MainWindow
from file_dialog import file_select, latest_file

#-------------------------------------------------------
# (c)2021, Florian Messner
# Lehrstuhl für Umformtechnik, MU Leoben
# Erstellt am: 11.03.2021
#-------------------------------------------------------


class GUI(QtWidgets.QMainWindow):
    def __init__(self,parent=None):
        # Initialisierung, Setup des UI

        super().__init__(parent)

        self.setWindowTitle('Filter program')
        self.setMinimumSize(700,500)

        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        self.connect_signals()
        self.name_buttons()


    def connect_signals(self):
        # Klick Events der Buttons mit den entsprechenden
```

```
        # Funktionen verbinden

        self.ui.button_latest.clicked.connect(self.
            click_button_latest)
        self.ui.button_clear.clicked.connect(self.click_button_clear
            )
        self.ui.button_select.clicked.connect(self.
            click_button_select)
        self.ui.button_cleartxt.clicked.connect(self.
            click_button_cleartxt)

    def name_buttons(self):
        #Möglichkeit der manuellen Anpassung der Button Labels

        self.ui.button_latest.setText('Letzte Messung')
        self.ui.button_clear.setText('Plots löschen')
        self.ui.button_select.setText('Messung wählen')
        self.ui.button_cleartxt.setText('Tabelle löschen')

    def click_button_latest(self):
        #automatische Auswahl der letzten Messung
        filepath = latest_file()

        # Einlesen und Aufbereiten der Versuchsdaten
        df = filter.filt_df(filepath)
        y_gaus = filter.Gauss_filter(df['elapsed time'], df['Lin_Pot
            '], 1, 0.1)

        # Plotten
        self.click_button_clear()
        self.ui.Mpl_Ft.plot_Ft(df)
        self.ui.Mpl_st.plot_st(df, y_gaus)
        self.ui.Mpl_Fges.plot_Fges(df)

        # Befüllen Ergebistabelle
        self.results_table(df, filepath)

    def click_button_clear(self):
        # Löschen der dargestellten Plots

        self.ui.Mpl_Ft.clear_window()
        self.ui.Mpl_st.clear_window()
        self.ui.Mpl_Fges.clear_window()

    def click_button_select(self):
        # manuelle Auswahl eines Versuchs per File Dialog
        filepath = file_select()
```

```
        if not filepath:
            return
        # Einlesen und Aufbereiten der Versuchsdaten
        df = filter.filt_df(filepath)
        y_gaus = filter.Gauss_filter(df['elapsed time'], df['Lin_Pot
            '], 1, 0.1)

        # Plotten
        self.click_button_clear()
        self.ui.Mpl_Ft.plot_Ft(df)
        self.ui.Mpl_st.plot_st(df, y_gaus)
        self.ui.Mpl_Fges.plot_Fges(df)

        # Befüllen Ergebnisstabelle
        self.results_table(df, filepath)

    def click_button_cleartxt(self):
        # Reseten des Inhalts Ergebnisstabelle
        self.ui.res_table.clearContents()
        self.ui.res_table.setRowCount(0)


    def results_table(self,df, filepath):
        # Darstellung der berechneten Daten

        # Ausgabe des Dateipfad von dargestelltem Versuch
        self.ui.label.setText('Zuletzt eingelesen: '+filepath)

        res = filter.get_table_values(df)
        # notwendige größe der Tabelle bestimmen
        if self.ui.res_table.rowCount() == 0:
            self.ui.res_table.clear()
            self.ui.res_table.setColumnCount(len(res.keys()))
            self.ui.res_table.setHorizontalHeaderLabels(res.keys())
        self.ui.res_table.insertRow(0)

        # Befüllen der Tabelle
        for i in range(self.ui.res_table.columnCount()):
            for key, value in res.items():
                if key == self.ui.res_table.horizontalHeaderItem(i).
                    text():
                    item = QtWidgets.QTableWidgetItem(format(value,
                        '.2f'))
                    self.ui.res_table.setItem(0, i, item)
        # Formatierung
        for j in range(self.ui.res_table.columnCount()):
            self.ui.res_table.item(0, j).setBackground(QtGui.QColor
```

```
                (0, 155, 255))
            for k in range(1, self.ui.res_table.rowCount()):
                self.ui.res_table.item(k, j).setBackground(QtGui.
                    QColor(255, 255, 255))

        delegate = AlignDelegate(self.ui.res_table)
        self.ui.res_table.setItemDelegate(delegate)


class AlignDelegate(QtWidgets.QStyledItemDelegate):
    # Klasse zur zentrierten Ausrichtung der Tabellenwerte

    def initStyleOption(self, option, index):
        super(AlignDelegate, self).initStyleOption(option, index)
        option.displayAlignment = QtCore.Qt.AlignCenter


def run_app():
    # Starten der Applikation, Aufbau des GUI, Anzeigen des GUI
    app = QtWidgets.QApplication(sys.argv)
    window = GUI()
    window.showMaximized()
    sys.exit(app.exec_())


run_app()
```

Listing A.2: file_dialog.py - Versuchsauswahl

```
from PyQt5.QtWidgets import QFileDialog
import os
from glob import glob
#-----------------------------------------------------------
# (c)2021, Florian Messner
# Lehrstuhl für Umformtechnik, MU Leoben
# Erstellt am: 15.03.2021
#-----------------------------------------------------------

def file_select():

    # path ist der angezeigte Startpfad im File Dialog
    path = './data'
    filedialog = QFileDialog()
    filter = 'csv(*.csv)'
    file,_ = QFileDialog.getOpenFileName(filedialog, 'CSV-Datei ausw
        ählen!', path, filter)
    return file
```

```
def latest_file():
    # Output ist der Pfad des letzten, durchgeführten Versuches

    csvlst = []
    # Ordner in dem gesucht wird
    dir_name = "./data"
    for (root, dirs, files) in os.walk(dir_name, topdown=True):
        for file in files:
            if file.endswith('.csv'):
                csvlst.append(os.path.join(root, file))
    latest = max(csvlst,key = os.path.getmtime)
    return latest
```

Listing A.3: df_import.py - Datenimport, Datenaufbereitung

```
import pandas as pd

#-----------------------------------------------------------
# (c)2021, Florian Messner
# Lehrstuhl für Umformtechnik, MU Leoben
# Erstellt am: 11.03.2021
#-----------------------------------------------------------

def filt_df(file):
    # Datenaufbereitung
    # input: file: Dateipfad der CSV Datei
    # output: gefiltertes Dataframe
    try:
        df = pd.read_csv(file, delimiter=';', error_bad_lines=False,
             parse_dates=['Timestamp'],
                          date_parser=lambda x: pd.to_datetime(x,
                              errors="coerce"))
    except ValueError:
        # Spaltenbeschriftung wenn CSV-Datei keinen Labelheader
            besitzt
        colnames = ['Timestamp', 'KMD_Links', 'KMD_Rechts', '
            Degree_Sens', 'Lin_Pot']
        print('No column names defined in file.\n Changed to: ',
            colnames)

        df = pd.read_csv(file, delimiter=';', header=0, names=
            colnames,
                          error_bad_lines=False, parse_dates=['
                              Timestamp'],
```

```
                            date_parser=lambda x: pd.to_datetime(x,
                                errors="coerce"))

    # Datenfilterung
    df.sort_values(['Timestamp'], inplace=True)
    df_dup = df[df.duplicated(['Timestamp']) == False]
    df_nan = df_dup.dropna(axis=0, how='any', inplace=False)
    df_nan2 = df_nan.replace(0, float('NaN'), inplace=False)
    df_null = df_nan2.dropna(axis=1, inplace=False, how='all')
    df_fil = df_null.reset_index(inplace=False, drop=False)

    # Einfügen neuer Spalten
    df_fil.insert(0, 'elapsed time', (df_fil['Timestamp']-df_fil['
        Timestamp'].iloc[0]).dt.total_seconds())
    df_fil.insert(1, 'F_ges', df_fil['KMD_Links']+df_fil['KMD_Rechts
        '])
    return df_fil
```

Listing A.4: filter_functions.py - Funktionen zur Datenverarbeitung

```
import numpy as np
from scipy import signal, special

#-------------------------------------------------------
# (c)2021, Florian Messner
# Lehrstuhl für Umformtechnik, MU Leoben
# Erstellt am: 06.04.2021
#-------------------------------------------------------

def Gauss_filter(x, y_in, n, sigma):
    # Gauß-Filter
    # nötiger Input: x...arraylike, 1D, x-Werte
    # y_in: dazugehörige y-Eingangswerte, gleiche Länge (für
        convolution)
    # n..default 1, für mehrfache Filterung
    # sigma... charakteristischer wert für die Breite der Gauß-Kurve
    #
    # output: gefilterten y-Werte

    g = 0
    for i in range(n):
        k = i+1
        g += special.binom(n, k) * (-1)**(k+1)*1/(sigma*np.sqrt(2*np
            .pi*k))*np.exp(-(x-np.
             mean(x))**2/(2*sigma**2*k**2))
    y = signal.convolve(y_in, g, mode='same')/np.sum(g)
    return y
```

```
def get_table_values(df):
    # Aufbau eines Results Dictionary
    # Befüllung mit den Werten für die Ergebnisstabelle

    res = {}
    res['F_links [kN]'] = max(df['KMD_Links'])
    res['F_rechts [kN]'] = max(df['KMD_Rechts'])
    res['F_ges [kN]'] = max(df['F_ges'])
    res['s_max [mm]'] = max(df['Lin_Pot'])
    s0_ = df[df['Lin_Pot'] < 0.95 * max(df['Lin_Pot'])]
    res['s0 [mm]'] = np.mean(s0_['Lin_Pot'])
    return res
```

Listing A.5: mplwidget.py - Custom Klasse MplWidget

```
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
import numpy as np

#-------------------------------------------------------
# (c)2021, Florian Messner
# Lehrstuhl für Umformtechnik, MU Leoben
# Erstellt am: 31.03.2021
#-------------------------------------------------------

class MplWidget(FigureCanvasQTAgg):
    # Klasse für benutzerdefiniertes Widget MplWidget

    def __init__(self, parent):
        # Initialisierung
        # Erstellen Widget, Figure, Axis Objekt
        self.fig = plt.figure(tight_layout=True)
        super().__init__(self.fig)
        self.setParent(parent)
        self.ax = plt.axes()
        self.ax.grid(b=True, linewidth=.5)


    def clear_window(self):
        # Reset Figure, Plot löschen
        self.fig.clear()
        self.ax = self.fig.add_subplot(111)
        self.ax.grid(b=True, linewidth=.5)
        plt.tight_layout()
        self.draw()
```

```
    def plot_st(self, df, s_gaus):
        # Plotten des Diagramms Walzspalt über Zeit

        self.ax.plot(df['elapsed time'], df['Lin_Pot'], alpha=.3)
        self.ax.plot(df['elapsed time'], s_gaus, scaley=False)
        self.ax.legend(['Originaldaten', 'Gaus Filter (sigma = 0,1)'
            ], loc='upper right')

        s0_ = df[df['Lin_Pot'] < 0.90 * max(df['Lin_Pot'])]
        s0 = np.mean(s0_['Lin_Pot'])
        self.ax.axhline(max(df['Lin_Pot']), color ='seagreen',
            linestyle ='--', alpha=0.3)
        self.ax.axhline(s0, color ='seagreen', linestyle ='--',
            alpha=0.3)

        self.ax.set(xlabel='Zeit [s]', ylabel='Walzspalt [mm]')
        self.ax.set_ylim([min(df['Lin_Pot']) - 0.01, None])
        self.ax.grid(b=True, linewidth=.5)
        self.draw()


    def plot_Ft(self, df):
        # Plotten des Diagramms Walzkraft über Zeit

        color = ['firebrick', 'steelblue']
        df.plot(x='elapsed time', y=['KMD_Links', 'KMD_Rechts'],
                label=['F_links', 'F_rechts'], legend=True, color=
                    color, ax=self.ax)

        self.ax.set(xlabel='Zeit [s]', ylabel='Walzkraft [kN]')
        plt.legend(loc='upper right')
        self.ax.grid(b=True, linewidth=.5)
        self.draw()


    def plot_Fges(self, df):
        # Plotten des Diagramms gesmat Walzkraft über Zeit

        df.plot(x='elapsed time', y='F_ges', legend=False, ax=self.
            ax)
        self.ax.axhline(max(df['F_ges']), color ='seagreen',
                        linestyle ='--', alpha=0.3)
        self.ax.set(xlabel='Zeit [s]', ylabel='Walzkraft gesamt [kN]
            ')
        self.ax.grid(b=True, linewidth=.5)
        self.draw()
```

Listing A.6: generierter Quellcode des Mainwindow vom GUI

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'ui\mainwindow.
    ui'
#
# Created by: PyQt5 UI code generator 5.15.4
#
# WARNING: Any manual changes made to this file will be lost when
    pyuic5 is
# run again.  Do not edit this file unless you know what you are
    doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1126, 840)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
            Maximum, QtWidgets.QSizePolicy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(MainWindow.sizePolicy().
            hasHeightForWidth())
        MainWindow.setSizePolicy(sizePolicy)
        MainWindow.setMinimumSize(QtCore.QSize(700, 500))
        MainWindow.setLayoutDirection(QtCore.Qt.RightToLeft)
        MainWindow.setStyleSheet("background-color: rgb(255, 255,
            255);\n"
"")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
            Maximum, QtWidgets.QSizePolicy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.centralwidget.sizePolicy()
            .hasHeightForWidth())
        self.centralwidget.setSizePolicy(sizePolicy)
        self.centralwidget.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.centralwidget.setAutoFillBackground(False)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setContentsMargins(9, 9, 9, 9)
```

```
self.gridLayout.setSpacing(6)
self.gridLayout.setObjectName("gridLayout")
self.Mpl_Ft = MplWidget(self.centralwidget)
self.Mpl_Ft.setEnabled(True)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    MinimumExpanding, QtWidgets.QSizePolicy.MinimumExpanding
    )
sizePolicy.setHorizontalStretch(1)
sizePolicy.setVerticalStretch(1)
sizePolicy.setHeightForWidth(self.Mpl_Ft.sizePolicy().
    hasHeightForWidth())
self.Mpl_Ft.setSizePolicy(sizePolicy)
self.Mpl_Ft.setMinimumSize(QtCore.QSize(0, 0))
self.Mpl_Ft.setAutoFillBackground(False)
self.Mpl_Ft.setObjectName("Mpl_Ft")
self.gridLayout.addWidget(self.Mpl_Ft, 0, 0, 1, 1)
self.Mpl_st = MplWidget(self.centralwidget)
self.Mpl_st.setEnabled(True)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    MinimumExpanding, QtWidgets.QSizePolicy.MinimumExpanding
    )
sizePolicy.setHorizontalStretch(1)
sizePolicy.setVerticalStretch(1)
sizePolicy.setHeightForWidth(self.Mpl_st.sizePolicy().
    hasHeightForWidth())
self.Mpl_st.setSizePolicy(sizePolicy)
self.Mpl_st.setMinimumSize(QtCore.QSize(0, 0))
self.Mpl_st.setAutoFillBackground(False)
self.Mpl_st.setObjectName("Mpl_st")
self.gridLayout.addWidget(self.Mpl_st, 3, 0, 1, 1)
self.Mpl_Fges = MplWidget(self.centralwidget)
self.Mpl_Fges.setEnabled(True)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    MinimumExpanding, QtWidgets.QSizePolicy.MinimumExpanding
    )
sizePolicy.setHorizontalStretch(1)
sizePolicy.setVerticalStretch(1)
sizePolicy.setHeightForWidth(self.Mpl_Fges.sizePolicy().
    hasHeightForWidth())
self.Mpl_Fges.setSizePolicy(sizePolicy)
self.Mpl_Fges.setMinimumSize(QtCore.QSize(0, 0))
self.Mpl_Fges.setLayoutDirection(QtCore.Qt.LeftToRight)
self.Mpl_Fges.setAutoFillBackground(False)
self.Mpl_Fges.setObjectName("Mpl_Fges")
self.gridLayout.addWidget(self.Mpl_Fges, 0, 1, 1, 1)
self.verticalLayout_4 = QtWidgets.QVBoxLayout()
self.verticalLayout_4.setObjectName("verticalLayout_4")
```

```
self.button_clear = QtWidgets.QPushButton(self.centralwidget
    )
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.Fixed)
sizePolicy.setHorizontalStretch(1)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.button_clear.sizePolicy().
    hasHeightForWidth())
self.button_clear.setSizePolicy(sizePolicy)
self.button_clear.setStyleSheet("")
self.button_clear.setObjectName("button_clear")
self.verticalLayout_4.addWidget(self.button_clear)
self.button_latest = QtWidgets.QPushButton(self.
    centralwidget)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.Fixed)
sizePolicy.setHorizontalStretch(1)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.button_latest.sizePolicy()
    .hasHeightForWidth())
self.button_latest.setSizePolicy(sizePolicy)
self.button_latest.setObjectName("button_latest")
self.verticalLayout_4.addWidget(self.button_latest)
self.button_select = QtWidgets.QPushButton(self.
    centralwidget)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.Fixed)
sizePolicy.setHorizontalStretch(1)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.button_select.sizePolicy()
    .hasHeightForWidth())
self.button_select.setSizePolicy(sizePolicy)
self.button_select.setObjectName("button_select")
self.verticalLayout_4.addWidget(self.button_select)
self.label = QtWidgets.QLabel(self.centralwidget)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.Minimum)
sizePolicy.setHorizontalStretch(1)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.label.sizePolicy().
    hasHeightForWidth())
self.label.setSizePolicy(sizePolicy)
self.label.setFrameShape(QtWidgets.QFrame.Panel)
self.label.setFrameShadow(QtWidgets.QFrame.Sunken)
self.label.setLineWidth(2)
self.label.setObjectName("label")
self.verticalLayout_4.addWidget(self.label)
```

```
self.res_table = QtWidgets.QTableWidget(self.centralwidget)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.MinimumExpanding)
sizePolicy.setHorizontalStretch(1)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.res_table.sizePolicy().
    hasHeightForWidth())
self.res_table.setSizePolicy(sizePolicy)
self.res_table.setMinimumSize(QtCore.QSize(0, 50))
self.res_table.setAutoScroll(False)
self.res_table.setShowGrid(True)
self.res_table.setObjectName("res_table")
self.res_table.setColumnCount(4)
self.res_table.setRowCount(0)
item = QtWidgets.QTableWidgetItem()
self.res_table.setHorizontalHeaderItem(0, item)
item = QtWidgets.QTableWidgetItem()
self.res_table.setHorizontalHeaderItem(1, item)
item = QtWidgets.QTableWidgetItem()
self.res_table.setHorizontalHeaderItem(2, item)
item = QtWidgets.QTableWidgetItem()
self.res_table.setHorizontalHeaderItem(3, item)
self.res_table.horizontalHeader().setVisible(True)
self.res_table.verticalHeader().setVisible(False)
self.verticalLayout_4.addWidget(self.res_table)
self.button_cleartxt = QtWidgets.QPushButton(self.
    centralwidget)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.
    Minimum, QtWidgets.QSizePolicy.Fixed)
sizePolicy.setHorizontalStretch(1)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.button_cleartxt.sizePolicy
    ().hasHeightForWidth())
self.button_cleartxt.setSizePolicy(sizePolicy)
self.button_cleartxt.setObjectName("button_cleartxt")
self.verticalLayout_4.addWidget(self.button_cleartxt)
self.gridLayout.addLayout(self.verticalLayout_4, 2, 1, 2, 1)
spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.
    QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)
self.gridLayout.addItem(spacerItem, 4, 0, 1, 1)
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 1126, 21))
self.menubar.setObjectName("menubar")
MainWindow.setMenuBar(self.menubar)
self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")
```

```
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "
            MainWindow"))
        self.button_clear.setText(_translate("MainWindow", "clear
            plots"))
        self.button_latest.setText(_translate("MainWindow", "show
            latest"))
        self.button_select.setText(_translate("MainWindow", "select
            measurement"))
        self.label.setText(_translate("MainWindow", "filename label"
            ))
        item = self.res_table.horizontalHeaderItem(0)
        item.setText(_translate("MainWindow", "Neue Spalte"))
        item = self.res_table.horizontalHeaderItem(1)
        item.setText(_translate("MainWindow", "Neue Spalte"))
        item = self.res_table.horizontalHeaderItem(2)
        item.setText(_translate("MainWindow", "Neue Spalte"))
        item = self.res_table.horizontalHeaderItem(3)
        item.setText(_translate("MainWindow", "Neue Spalte"))
        self.button_cleartxt.setText(_translate("MainWindow", "clear
             textbox"))
from mplwidget import MplWidget
```

Anhang B

Datenblätter

Auf den nächsten Seiten sind die Datenblätter für verbaute Hardwarekomponenten angeführt.

- Raspberry Pi Model 4 [47]
- Joy-IT Touchscreen [48]

RB-LCD10-2

10.1“ IPS Touchscreen Display mit Metallgehäuse

Das 10“ Touch Display ist ein robustes Multitalent. Es zeichnet sich besonders durch seine robuste Verarbeitung aus. Das mattschwarze Metallgehäuse ergänzt sich hierbei perfekt mit der Hartbeschichtung des Displays (H3).

Durch die große Anzahl an Anschlüssen und den eingebauten Lautsprechern eignet es sich unter anderem als Zweitmonitor für PC und Notebook, Werkstattdisplay, Überwachungsmonitor für Kamerasysteme oder mobilen Einsatz im Caravan.

Es ist zudem durch die vorhandenen Bohrungen auf der Rückseite ohne Weiteres zur Wandmontage geeignet.

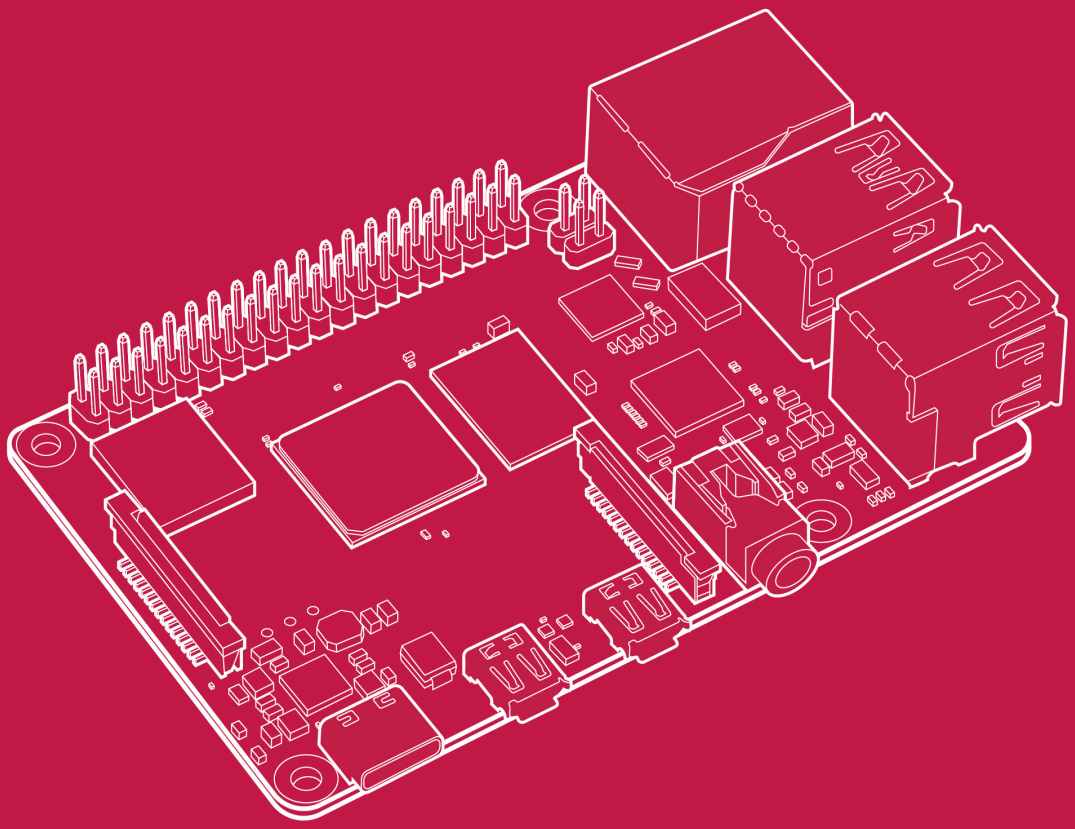
HAUPTMERKMALE	
Display	10,1“ IPS (25,65cm)
Auflösung	1280 x 800 Pixel
Helligkeit	350 cd/m²
Kontrast	800:1
Blickwinkel	160° - 170°
Farben	16,7M
Betriebstemperatur	0 - 50°C
Energieversorgung	12V / 2A

BESONDERHEITEN	
Features	Integrierte Lautsprecher Blendschutz Hartbeschichtung (3H)
Anschlüsse	HDMI, VGA, AV, BNC, 2x USB
Wandhalterung	VESA kompatibel (75x75)

WEITERE DETAILS	
Abmessungen (ohne Fuß)	244 x 163 x 32mm
Gewicht	1,2 kg
Artikelnummer	RB-LCD10-2
EAN	4250236815633
Zolltarifnummer	8473302000
Lieferumfang	Display, Adapterplatine, HDMI-Kabel, USB-Kabel Typ A, USB-Kabel Typ A-MicroUSB, Netzteil, Ersatzschrauben, Montagematerial, Fernbedienung

Veröffentlicht: 08.05.2020

Raspberry Pi 4 Computer
Model B

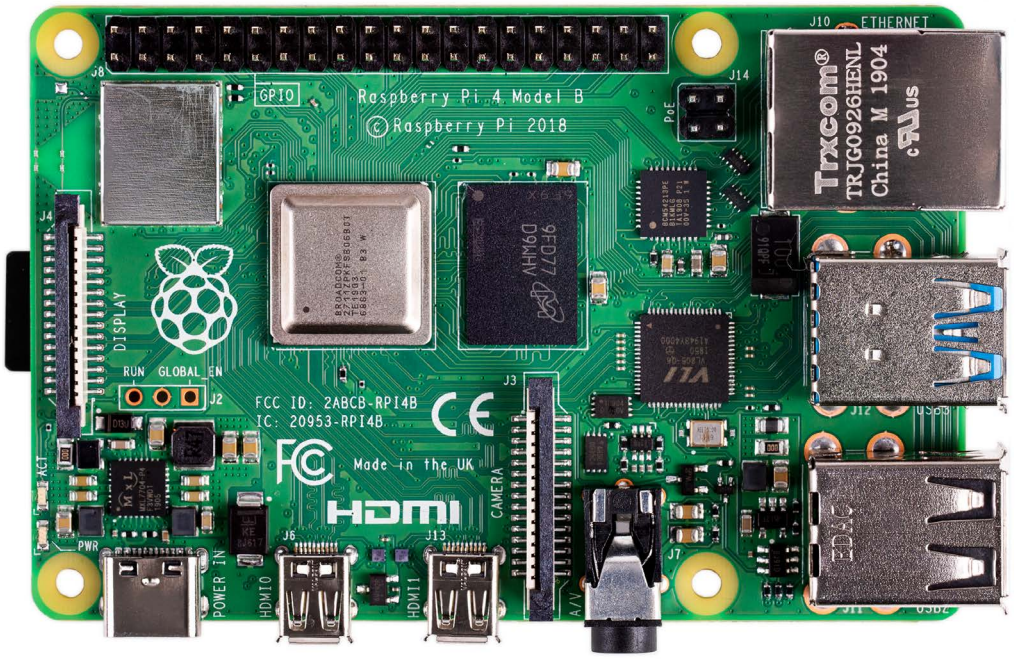


Published in June 2019
by Raspberry Pi Trading Ltd.

www.raspberrypi.org

Raspberry Pi

Overview



Raspberry Pi 4 Model B is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption. For the end user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems.

This product's key features include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on).

The dual-band wireless LAN and Bluetooth have modular compliance certification, allowing the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market.

Specification

Processor:	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory:	1GB, 2GB or 4GB LPDDR4 (depending on model)
Connectivity:	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports.
GPIO:	Standard 40-pin GPIO header (fully backwards-compatible with previous boards)
Video & sound:	2 × micro HDMI ports (up to 4Kp60 supported) 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port
Multimedia:	H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics
SD card support:	Micro SD card slot for loading operating system and data storage
Input power:	5V DC via USB-C connector (minimum 3A[1]) 5V DC via GPIO header (minimum 3A[1]) Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature 0–50°C
Compliance:	For a full list of local and regional product approvals, please visit https://www.raspberrypi.org/documentation/hardware/raspberrypi/conformity.md
Production lifetime:	The Raspberry Pi 4 Model B will remain in production until at least January 2026.