Diplomarbeit

Realisierung einer mobilen ortsaufgelösten Druckmessvorrichtung zur Analyse von Einflussfaktoren auf die Fußdruckverteilung im kfz-Kontext

eingereicht an der

Fakultät Kraftfahrzeugtechnik der

Westsächsischen Hochschule Zwickau

zur Erlangung des akademischen Grades eines

Diplomingenieurs (FH)

vorgelegt von: Zepeng Yan Geb. am:24.07.1998

Studiengang： Kraftfahrzeugtechnik

Stuienschwerpunk: Kraftfahrzeuge und Mechatronik (KFM)

Ausgegeben von:	**Prof. Martin Dannemann**
Erstbetreuer:	**Prof. Martin Dannemann**
Zweiterpüfer:	**Prof. Marco Beier**

Autorenreferat

Titel: Realisierung einer mobilen ortsaufgelösten Druckmessvorrichtung zur Analyse von Einflussfaktoren auf die Fußdruckverteilung im kfz-Kontext

Zusammenfassung:

Diese Arbeit stellt die Entwicklung eines Druckvisualisierungssystems vor, das auf Arduino-Mikrocontroller-Technologie und SVG-Grafiken basiert. Ziel ist die Erfassung und Visualisierung von Druckverteilungen, beispielsweise auf einem Fahrpedal oder einer Fußmatte in Fahrzeugen. Das System verwendet acht druckempfindliche Widerstände (FSR), die in einer speziell entwickelten Anordnung platziert sind, um eine präzise Erfassung der Druckverteilung zu ermöglichen. Die vom Arduino erfassten Daten werden mittels einer synchronen Echtzeitübertragung an ein Visualisierungsmodul übermittelt. Das Projekt umfasst die Entwicklung der Sensorhardware, die Programmierung des Arduino für die Datenerfassung und Übertragung sowie die Implementierung einer Visualisierungsschnittstelle in Visual Studio. Besondere Aufmerksamkeit wurde der Interaktivität des Systems gewidmet, sodass Änderungen des Drucks unmittelbar auf dem Bildschirm reflektiert werden. Die Visualisierung nutzt farbliche Intensitäten, um unterschiedliche Druckstärken darzustellen: Je größer der Druck, desto intensiver die Farbe.

Durch eine Kombination aus kostengünstiger Hardware und Open-Source-Software bietet das System eine innovative und benutzerfreundliche Lösung für Anwendungsfälle wie die Überwachung von Belastungsverteilungen, ergonomische Analysen und Fahrzeugdesign. Die Flexibilität des Systems ermöglicht eine einfache Anpassung an verschiedene Einsatzgebiete, was es für die Forschung und Entwicklung in der Automobilindustrie besonders attraktiv macht.

Schlüsselwörter: Diese Arbeit befasst sich mit der Integration von Sensoren (Sensorintegration) und der Entwicklung einer synchrone Echtzeitübertragung zur Druckvisualisierung mithilfe eines Arduino-Mikrocontrollers und SVG-Grafiken, insbesondere im Hinblick auf ergonomische Analysen in Fahrzeuganwendungen

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit in allen Teilen selbstständig angefertigt und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt habe, und dass die Arbeit in gleicher oder ähnlicher Form in noch keiner anderen Prüfung vorgelegen hat. Mir ist bewusst, dass ich Autor/in der vorliegenden Arbeit bin und volle Verantwortung für den Text trage.

Ich erkläre, dass ich wörtlich oder sinngemäß aus anderen Werken – dazu gehören auch Internetquellen – übernommene Inhalte als solche kenntlich gemacht und die entsprechenden Quellen angegeben habe.

Mir ist bewusst, dass meine Arbeit auf Plagiate überprüft werden kann. Mir ist bekannt, dass es sich bei der Abgabe eines Plagiats um ein schweres akademisches Fehlverhalten handelt und dass Täuschungen nach der für mich gültigen Prüfungsordnung geahndet werden.

Zusätzlich versichere ich, dass ich auf künstlicher Intelligenz (KI) basierende Werkzeuge nur in Absprache mit den Prüfern verwendet habe. Dabei stand meine eigene geistige Leistung im Vordergrund, und ich habe jederzeit den Prozess steuernd bearbeitet.

Diese Werkzeuge habe ich im Quellenverzeichnis in der Rubrik „Übersicht verwendeter Hilfsmittel“ mit ihrem Produktnamen und einer Übersicht des im Rahmen dieser Prüfungs-/Studienarbeit genutzten Funktionsumfangs unter Angabe der Textstelle in der Arbeit vollständig aufgeführt.

Ich versichere, dass ich keine KI-basierten Tools verwendet habe, deren Nutzung die Prüfer explizit schriftlich ausgeschlossen haben. Ich bin mir bewusst, dass die Verwendung von Texten oder anderen Inhalten und Produkten, die durch KI-basierte Tools generiert wurden, keine Garantie für deren Qualität darstellt.

Ich verantworte die Übernahme jeglicher von mir verwendeter maschinell generierter Passagen vollumfänglich selbst und trage die Verantwortung für eventuell durch die KI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.

Ort, Datum: **Zwickau , 2025/4/19** Unterschrift: **Yan,zepeng**

Inhalt

Vorwort

Ich möchte mich an dieser Stelle herzlich bei allen Personen bedanken, die mich während der Anfertigung dieser Diplomarbeit unterstützt haben. Mein besonderer Dank gilt Herrn Prof. Martin Dannemann für die engagierte Betreuung, die fachliche Begleitung sowie die wertvollen Hinweise zur technischen Umsetzung des Projekts.

Ebenso danke ich den Mitarbeiterinnen und Mitarbeitern der Fakultät Kraftfahrzeugtechnik für die Bereitstellung der technischen Infrastruktur und des notwendigen Equipments, welches die Durchführung der praktischen Versuche maßgeblich ermöglicht hat.

Darüber hinaus möchte ich meinen Kommilitoninnen und Kommilitonen danken, die mir mit konstruktivem Feedback bei der Validierung der entwickelten Visualisierungslösung geholfen haben.

Nicht zuletzt gilt mein Dank meiner Familie für ihre Geduld und Unterstützung während der gesamten Studienzeit

1. Einleitung

In modernen Fahrzeugen gewinnt die präzise Erfassung und Analyse von Kraftverteilungen zunehmend an Bedeutung – sei es zur Optimierung der Ergonomie, zur Verbesserung der Fahrzeugsicherheit oder zur Unterstützung innovativer Fahrerassistenzsysteme[1]. .Besonders im Bereich der Pedalbedienung spielt die ortsaufgelöste Kraftmessung eine zentrale Rolle, da sie Aufschluss über das Benutzerverhalten und die Wirksamkeit von Bedienelementen geben kann.

Derzeit verfügbare Systeme zur Kraftmessung und -visualisierung sind jedoch häufig mit mehreren Nachteilen verbunden. Dazu zählen unter anderem die hohen Kosten für professionelle Messtechnik, der fehlende Fokus auf Echtzeitfähigkeit, die geringe Anpassungsfähigkeit an spezifische Anwendungsszenarien sowie eine unzureichend intuitive Datenvisualisierung. Gerade im Bereich kostensensitiver Entwicklungsprojekte und Prototypen fehlt es an flexiblen, leicht integrierbaren Lösungen.[2].

Ziel dieser Arbeit ist die Entwicklung eines kostengünstigen, modular aufgebauten Systems zur ortsaufgelösten Kraftmessung mit einem Mehrzonen-Force-Sensing-Resistor (FSR), basierend auf einem Einplatinenrechner (Arduino). [3]. Das System soll in der Lage sein, verteilte Kräfte in Echtzeit zu erfassen, zu analysieren und benutzerfreundlich darzustellen. Ein besonderer Fokus liegt auf der Visualisierung mittels skalierbarer Vektorgrafiken (SVG), welche die Darstellung unterschiedlicher Kraftintensitäten ermöglichen.

Zur Umsetzung dieses Ziels wurde zunächst eine umfassende Recherche zu aktuellen Sensoriklösungen und Messverfahren durchgeführt. Anschließend wurde eine Datenerfassungseinheit auf Basis von FSR-Sensoren und Arduino-Hardware konzipiert und aufgebaut. [4]. Die gesammelten Sensordaten werden in Echtzeit über eine serielle Schnittstelle an ein PC-basiertes Visualisierungsmodul übermittelt, das mithilfe einer C#-Anwendung in Visual Studio die Druckverteilung dynamisch und farbkodiert darstellt.

Als praxisnahes Messszenario diente die Untersuchung der Kraftverteilung beim Betätigen eines Bremspedals in unterschiedlichen ergonomischen Sitzpositionen. Die

aufgezeichneten Kraftprofile wurden analysiert und auf ihre Aussagekraft hinsichtlich ergonomischer und funktionaler Kriterien hin ausgewertet.[5].

Diese Arbeit zeigt, wie durch eine Kombination aus kosteneffizienter Hardware, zielgerichteter Softwareentwicklung und ingenieurwissenschaftlicher Methodik ein leistungsfähiges System zur Druckanalyse im Fahrzeugumfeld entstehen kann.

2. Sensoren und Messmethoden zur ortsaufgelösten Kraft-/Druckmessung

2.1 Einleitung zur Sensorwahl

In der Entwicklung mechatronischer Systeme spielt die Auswahl geeigneter Sensoren eine zentrale Rolle, insbesondere bei der Erfassung verteilter Druckkräfte. Für das vorliegende Projekt wurden verschiedene Sensortechnologien evaluiert, um eine Balance zwischen Messgenauigkeit, Kosten, Integrationsfähigkeit und Reaktion auf reale Belastungsszenarien zu gewährleisten. Dabei kristallisierten sich drei Sensortypen als relevant heraus: Dehnungsmessstreifen (DMS), piezoelektrische Sensoren und Force-Sensing-Resistoren (FSR)[6]. [3].

Dehnungsmessstreifen (DMS)

Dehnungsmessstreifen nutzen die Eigenschaft, dass sich der elektrische Widerstand eines Leiters bei mechanischer Dehnung proportional verändert. Diese Technologie liefert sehr präzise, lineare und reproduzierbare Ergebnisse und wird häufig in industriellen oder wissenschaftlichen Anwendungen eingesetzt. Aufgrund des erforderlichen Verstärkers sowie der komplexen Verkabelung und Kalibrierung ist der Einsatz jedoch mit hohem Aufwand verbunden und für kompakte, kostensensitive Systeme nur eingeschränkt geeignet.[7].

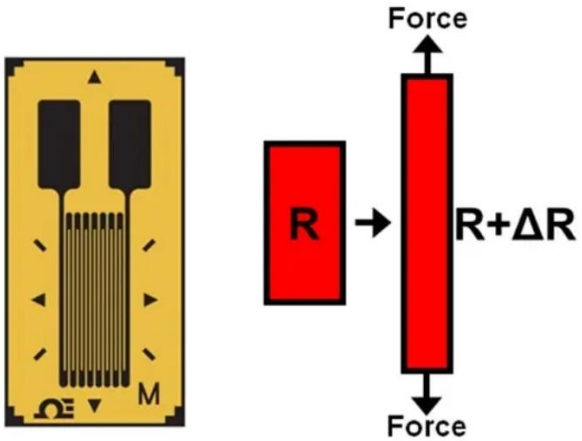


Abbildung 1 DMS Struktur

https://www.kistler.com/DE/de/dehnungsmessstreifen-funktion-anwendung-und-vorteile-/C00000156

Piezoelektrische Sensoren

Piezoelemente erzeugen elektrische Ladung, wenn sie mechanisch verformt werden. Ihre hohe Empfindlichkeit und Reaktionsgeschwindigkeit machen sie ideal für dynamische Anwendungen, etwa in der Schwingungs- und Stoßanalyse. Jedoch liefern sie bei statischer Belastung kein konstantes Signal und erfordern in der Regel teure Auswerteelektronik[15]. . Für den vorliegenden Anwendungsfall – die kontinuierliche Überwachung verteilter Druckverteilungen – sind sie daher weniger geeignet.[8].

Abbildung 2 Struktur der Piezoelektrische Sensoren

https://pruftechnik.com/nl/piezo-sensor-definitie-voordelen-hoe-ze-werken/

Force-Sensing-Resistoren (FSR)

FSR-Sensoren ändern ihren Widerstand abhängig vom anliegenden Druck. Sie zeichnen sich durch eine besonders kompakte Bauform, geringe Kosten, einfache Integration und hohe Robustheit aus. [9] Die Messcharakteristik ist jedoch deutlich nichtlinear und temperaturabhängig. [10]. Durch geeignete Kalibrierverfahren lässt sich diese Schwäche jedoch zumindest teilweise kompensieren. FSRs sind vor allem dort geeignet, wo Mehrzonenerfassung, intuitive Visualisierung und einfache Systemintegration im Vordergrund stehen [9].

DFSR-Drucksensoren (Force-Sensing Resistor: kraftabhängiger Widerstand) sind Widerstände, die bei Krafteinwirkung ihren Widerstand ändern:

Sie bestehen aus drei wesentlichen Komponenten:

1. Trägerfolie: Auf ihrer Innenseite ist eine schwarze sogenannte FSR-Schicht, die aus halbleitendem Material besteht, aufgedruckt.

2. Klebeschicht: Diese doppelseitig klebende Schicht verbindet die Komponenten des Sensors fest miteinander und stellt einen konstanten Abstand zwischen den beiden Trägerfolien her.

3. Trägerfolie für die Elektroden: Auf ihrer Innenseite sind kammartige Elektroden

aufgedruckt, die sich nicht berühren.

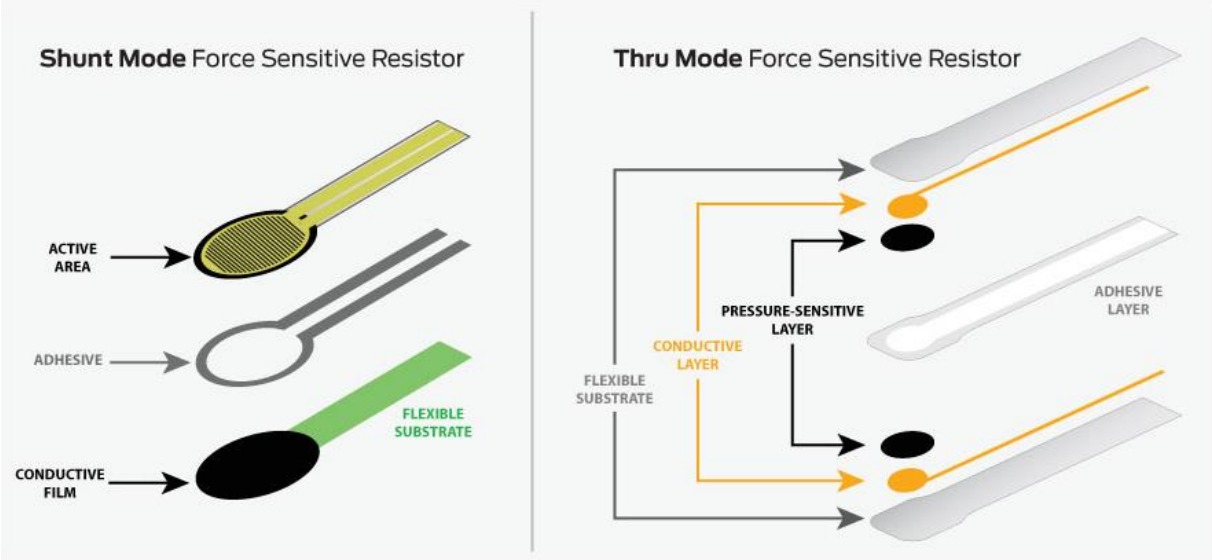


Abbildung 3 Prinzipielle Aufbau FSR-Sensor
https://www.tekscan.com/blog/flexiforce/how-does-force-sensing-resistor-fsr-work

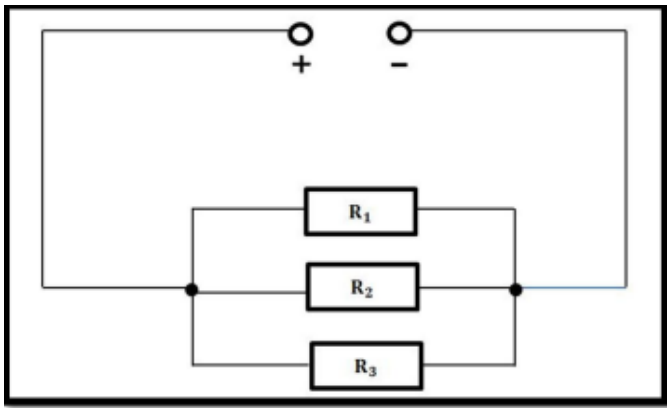
Dadurch werden Widerstandsbrücken aufgebaut, deren Anzahl mit steigender Kraftauswirkung auf die Sensoroberfläche zunimmt und welche alle parallel

zueinander geschaltet sind. Dies könnt ihr euch mit folgendem Schaubild noch einmal klar machen

Parallelschaltung von Widerständen:

$$\frac{1}{R_{Ges}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \cdots + \frac{1}{R_n}$$

Der Gesamtwiderstand R_{ges} verringert sich also mit zunehmender Druckausübung auf das Sensor-Bauelement.



Sensor-Bauelement

Abbildung 4 : http://www.electrade.com/index.php/drucksensoren.html

FSR-Sensoren weisen ein deutlich **nichtlineares Kraft-Widerstands-Verhalten** auf. Insbesondere im unteren und oberen Druckbereich zeigen sich starke Abweichungen von einem linearen Verlauf.

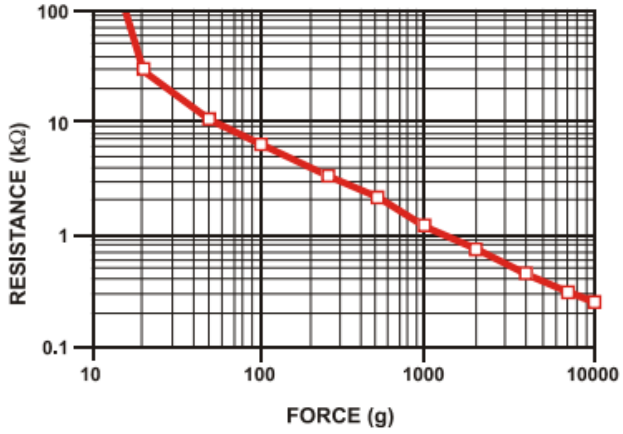


Abbildung 5 nichtlineares Kraft-Widerstands-Verhalten vom FSR-Sensor [25].

FSR-Sensoren zeichnen sich durch moderate Genauigkeit und geringe Kosten aus, was sie ideal für Anwendungen wie Multi-Sensor-Systeme macht.

In diesem Projekt wird die Spannungsteilerkonfiguration verwendet, um den FSR-Widerstand in ein Spannungssignal umzuwandeln. Diese einfache Schaltung ermöglicht eine Optimierung des Messbereichs durch Anpassung des Serienwiderstands.

Erweiterte Vergleichsanalyse

Zur Auswahl geeigneter Sensoren für die ortsaufgelöste Kraftmessung wurden drei etablierte Technologien systematisch verglichen: Dehnungsmessstreifen (DMS), piezoelektrische Sensoren und Force-Sensing-Resistoren (FSR). Die Bewertung erfolgte anhand technischer, funktionaler und wirtschaftlicher Kriterien. Tabelle 1 zeigt eine zusammenfassende Gegenüberstellung.

Tabelle 1 Vergleichsanalyse zur 3 Sensor

Sensortyp	**Funktionsprinzip**	**Vorteile**	**Nachteile**	**Typische Spezifikationen**
Dehnungsmesstreifen	Widerstansänderung bei Dehnung	Sehr präzise, etabliert[12].	Aufwendig, teuer, schlecht skalierbar[12].	Gauge-Faktor: ca. 2.0–2.2 Nichtlinearität: <0,1 % Temp. Bereich: −50 bis +200 °C[20].
Piezoelektrischer Sensor	Spannung bei Verformung	Hochsensibel, schnelle Reaktion[13].	Nur dynamisch nutzbar, teuer, komplexe Elektronik[13].	Sensitivität: 0,1–100 pC/N Frequenz: bis >100 kHz Reaktionszeit: <1 ms[21]. [21].

FSR (Force-Sensing Resistor)	Widerstandsänderung bei Druck	Günstig, flexibel, einfach zu integrieren[14].	Nichtlinear, begrenzte Genauigkeit[14].	Messbereich: 0–20 N Wiederholbarkeit: ±3–10 % Preis: €2–5[23]. [24].

2.2 Herausforderungen bei der Integration mehrerer Sensoren

Beim Einsatz mehrerer FSR-Sensoren in einem Array treten zusätzliche Herausforderungen auf:

- **Sensorinterferenzen**: Mechanische Kopplung oder elektrische Überschneidung benachbarter Sensoren kann zu fehlerhaften Messwerten führen.
- **Einzelkalibrierung**: Jeder Sensor hat eine leicht abweichende Kennlinie, sodass eine individuelle Kalibrierung erforderlich ist.
- **Synchronisation**: Gleichzeitige Messung mehrerer Kanäle verlangt konsistente Sampling-Zeitpunkte.
- **Datenrate**: Mehr Sensoren bedeuten mehr Daten → höhere Anforderungen an Mikrocontrollerleistung und Übertragungsbandbreite.

Diese Herausforderungen wurden im vorliegenden Projekt durch Segmentierung, Kalibrierformeln und serielle Datenübertragung bei konstanter Baudrate adressiert.**Entscheidung für FSR-Sensoren im Projektkontext**

Basierend auf den dargestellten Anforderungen – insbesondere Modularität, Visualisierbarkeit, Mehrzonenfähigkeit und niedrige Systemkosten – fiel die Entscheidung im Projekt bewusst zugunsten von FSR-Sensoren aus. Ihre gute Verfügbarkeit, bestehende Arduino-Bibliotheken sowie vorhandene Maker-Community-Beispiele erleichterten zudem die Integration. Trotz nichtlinearer Kennlinie

konnten durch MATLAB-gestützte Kalibrierung und segmentierte Umrechnung zuverlässige Ergebnisse erzielt werden[15].

2.3 Literaturüberblick: Anwendungen von FSR-Sensorik in Technik und Forschung

Force-Sensing-Resistoren (FSRs) haben sich in den letzten Jahren als vielseitig einsetzbare Sensorikkomponenten etabliert, insbesondere in Bereichen, in denen eine flexible, kostengünstige und flächige Druckerfassung benötigt wird. Während sie in der industriellen Hochpräzisionsmesstechnik weniger verbreitet sind als Dehnungsmessstreifen oder piezoelektrische Sensoren, finden sie im Bereich der Mensch-Maschine-Interaktion, Robotik und Fahrzeuginnenraumanalyse breite Anwendung.

Eine Studie von Kim et al. (2018) beschreibt den erfolgreichen Einsatz von FSR-Matrizen zur Erkennung von Sitzhaltungen in Fahrzeugsitzen.[16]. Dabei konnten durch druckbasierte Clusteranalysen unterschiedliche Sitzpositionen und Haltungsänderungen detektiert werden, was für Komfortsysteme oder Fahrermüdungswarnungen genutzt werden kann. In einem anderen Projekt entwickelte Gupta et al. (2019) ein Fußdruckmesssystem für die Ganganalyse bei Patienten mit neurologischen Erkrankungen[17]. Hier bewährte sich die FSR-Technologie aufgrund ihrer Biegsamkeit und der Möglichkeit, sie in textile Materialien einzubetten.

Auch im automobilen Kontext kommen FSRs zum Einsatz: Laut einer Untersuchung von Li et al. (2021) wurde ein Pedalkraft-Erkennungssystem für Fahrerassistenzfunktionen auf Basis von FSR-Folien realisiert[18]. Die dort eingesetzten Sensoren zeigten eine ausreichende Reproduzierbarkeit sowie ein anpassbares Verstärkungsschema zur Echtzeitdatenerfassung. Diese Forschung unterstreicht die Relevanz von FSRs in dynamischen, sicherheitsrelevanten Anwendungen.

Neben den praktischen Implementierungen existieren auch umfassende Übersichtsarbeiten: Zhang und Lee (2020) analysierten in ihrer Review die physikalischen Prinzipien, Alterungseffekte und Temperaturdriftprobleme von FSRs und gaben Hinweise zur Kalibrierung für Echtzeitanwendungen[19].

Diese Literaturbasis zeigt, dass FSR-Sensorik trotz gewisser Einschränkungen (nichtlineares Verhalten, Hysterese) für viele anwendungsorientierte Forschungsprojekte eine geeignete Grundlage bietet. Besonders die Kombination mit Software-gestützten Auswertungsverfahren und Echtzeitvisualisierung steigert ihren praktischen Nutzwert erheblich.

3.Konzeption und Aufbau einer Datenerfassungseinheit für FSR-Sensor

3.1 Konzept

Das Ziel des entwickelten Systems besteht in der ortsaufgelösten Erfassung und synchronen Darstellung der Druckverteilung auf einem Fußpedal. Dabei sollen die gemessenen Kräfte möglichst intuitiv, visuell verständlich und nahezu in Echtzeit dargestellt werden. Im Vordergrund stehen eine modulare Systemstruktur, kosteneffiziente Komponentenwahl sowie die Möglichkeit zur späteren Erweiterung um weitere Sensoren und Visualisierungsfunktionen.

Zur Umsetzung dieses Ziels wurde ein mehrstufiges Gesamtsystem entworfen, das sowohl Hardware- als auch Softwarekomponenten umfasst. Die Systemarchitektur folgt dabei einem klar strukturierten Datenfluss vom physischen Drucksignal bis hin zur grafischen Visualisierung auf dem Bildschirm. Die folgende Abbildung zeigt das grundlegende Blockdiagramm des entwickelten Systems:

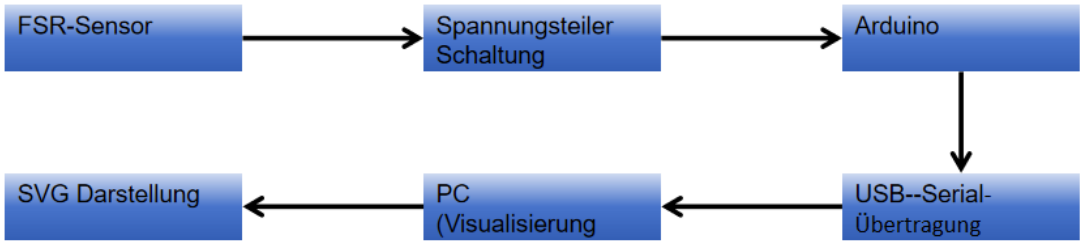


Abbildung 6 grundlegende Blockdiagramm zum Experiment

Das Gesamtsystem gliedert sich in folgende Hauptmodule:

- **FSR-Sensoren**: Erfassen die mechanischen Druckkräfte, wobei jeder Sensor eine definierte Zone der Fußmatte abdeckt.
- **Spannungsteiler-Schaltung**: Wandelt den variablen Widerstand des Sensors in ein analoges Spannungssignal um, das anschließend digitalisiert werden kann.

- **Arduino (Mikrocontroller)**: Liest die analogen Eingangssignale aus, berechnet anhand kalibrierter Kennlinien die Massewerte und sendet diese über eine serielle Schnittstelle weiter.

- **USB-Serial-Verbindung**: Stellt die Kommunikation zwischen Mikrocontroller und PC sicher (z. B. über einen USB-TTL-Adapter mit FTDI-Chip).

- **Visual Studio Anwendung (PC)**: Empfängt die Datenströme, analysiert sie und übergibt sie an das SVG-basierte Visualisierungsmodul.

- **SVG-Grafik (Frontend):** Stellt die Druckverteilung visuell dar, wobei Farbintensität und -verlauf die Belastungsstärke in den jeweiligen Zonen repräsentieren.

Durch diese strukturierte Aufteilung kann jedes Modul unabhängig entwickelt und getestet werden, wodurch eine klare Trennung von Zuständigkeiten und eine einfache Erweiterbarkeit gewährleistet werden.

3.2 Detaillierung

3.2.1 Verwendete Hardwarekomponenten

Für die Realisierung des Messsystems standen folgende Hardwarekomponenten zur Verfügung:

- **Elektronische Fußmatte mit 8 FSR-Sensoren**
  Eine vom Betreuer bereitgestellte Messmatte, in die acht Force-Sensing-Resistoren (FSRs) integriert sind. Jeder Sensor ist einer spezifischen Zone des Fußbereichs zugeordnet und ermöglicht so eine ortsaufgelöste Kraftmessung.

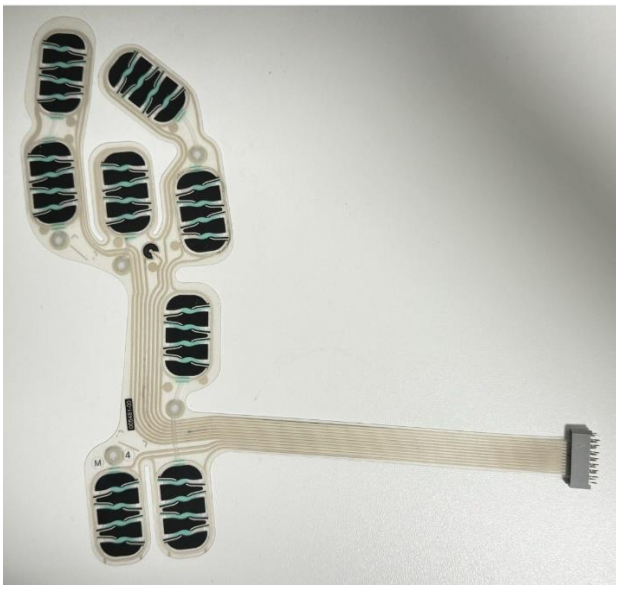


Abbildung 7 Fußmatte

Der reale Sensor ist ein achtzonen-FSR-Drucksensor, der in Form einer Fußmatte gestaltet ist, um sich an die Form der Fußsohle anzupassen. Dieser Sensor weist folgende Merkmale auf:

1. **Acht Drucksensorbereiche**:
    - Die Anordnung entspricht der Form einer Fußsohle und umfasst zwei Bereiche an der Ferse, einen im Mittelfußbereich und fünf im Vorfußbereich.
    - Jeder Bereich sammelt unabhängig Drucksignale und ermöglicht die Erstellung eines detaillierten Druckverteilungsmusters der Fußsohle.
2. **Arbeitsprinzip**:
    - Jeder Druckbereich gibt Signale durch Widerstandsänderungen aus, wobei der Widerstand umgekehrt proportional zum Druck ist. Diese Signale werden von einem Arduino erfasst und verarbeitet und in Spannungsdaten umgewandelt.
3. **Praktische Anwendungsbeispiele**:

- Im Rahmen des Experiments wurde der Sensor erfolgreich in ein simuliertes Fußpedalsystem integriert. Dabei konnte eine präzise und synchron echtzeitfähige Darstellung der Druckverteilung realisiert werden.

4. **Datenvisualisierung**:

    - Durch die Kombination mit SVG-Bildtechnologie werden die Druckgrößen der einzelnen Bereiche der Fußsohle dynamisch durch Farbänderungen dargestellt.

Widerstandssortiment(10 kΩ)

Zur Umsetzung der Spannungsteiler-Schaltung wurde ein Satz von 10-kΩ-Festwiderständen verwendet. Diese ermöglichen eine zuverlässige Umwandlung des druckabhängigen FSR-Widerstands in ein analoges Spannungssignal, das vom Mikrocontroller ausgelesen werden kann.

Arduino Mega 2560

Als zentrale Steuer- und Recheneinheit dient ein Arduino Mega 2560 Board. Es bietet ausreichend analoge Eingänge, um alle acht FSR-Signale simultan zu erfassen, und verfügt über ausreichend Rechenkapazität zur Umsetzung der Kalibrierformeln.

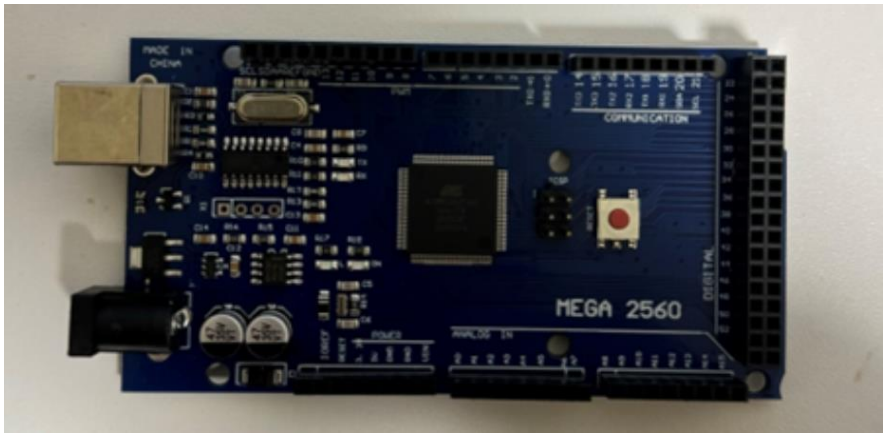


Abbildung 8 Arduino mega2560

- **USB-to-Serial Adapter (FTDI232)**
  Zur Datenübertragung vom Mikrocontroller an den PC wird ein USB-TTL-Konverter eingesetzt. Dieser stellt die serielle Kommunikation (Baudrate: 9600) sicher und versorgt im Versuchsaufbau gleichzeitig den Arduino mit Strom

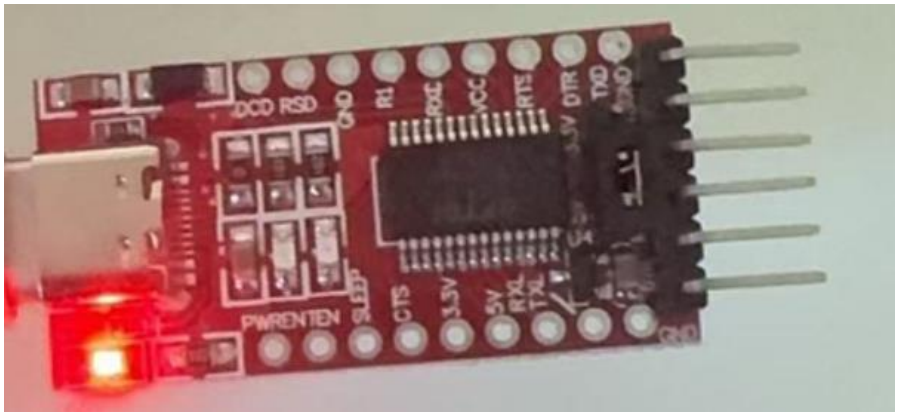


Abbildung 910 USB-to-Serial Adapter (FTDI232)

- **Visual Studio (Windows-PC)**

  Zur grafischen Darstellung der Messergebnisse wurde eine PC-Anwendung in C# mit Visual Studio entwickelt. Diese übernimmt die Datenverarbeitung, das Parsen der seriellen Daten und die visuelle Ausgabe mittels SVG-Grafik.

3.3.2 Systemaufbau und Verschaltungskonzept

Das zentrale Ziel beim Systemaufbau war es, eine zuverlässige und möglichst latenzarme Signalverarbeitungskette zu realisieren – vom Druckimpuls auf dem Sensor bis hin zur farbcodierten Visualisierung auf dem Bildschirm. Abbildung X veranschaulicht den prinzipiellen Aufbau:

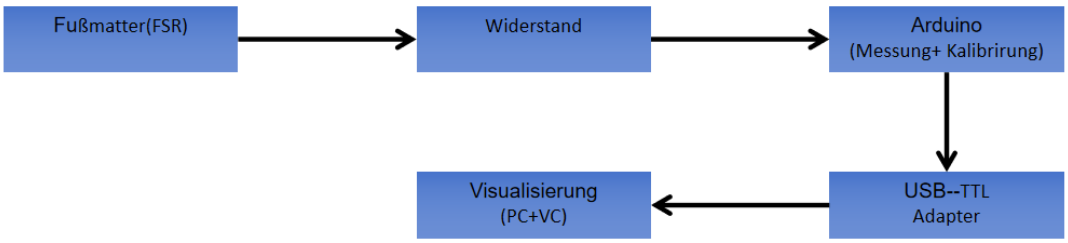


Abbildung 11 Datenfluss und Modulverschaltung (Blockdiagramm)

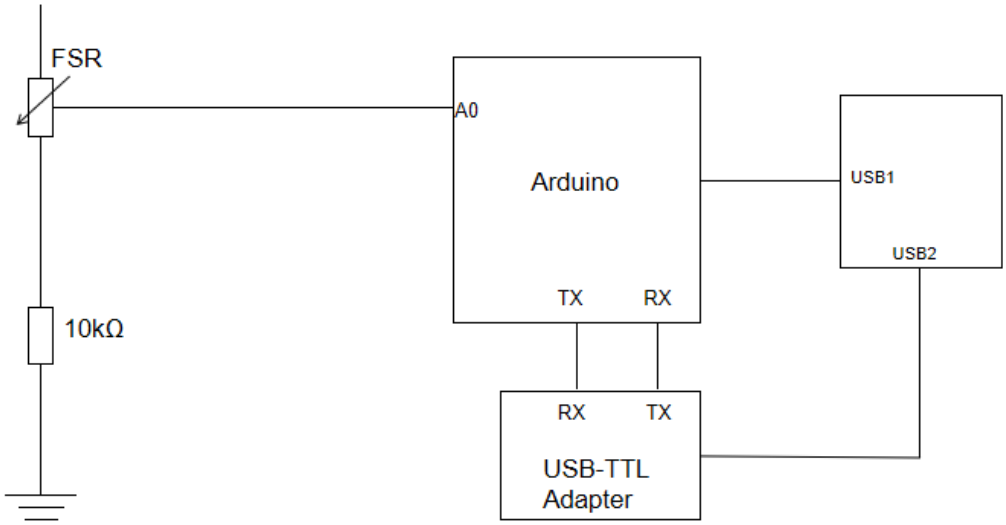


Abbildung 12 Skizze zur Modulverschaltung

1. **Signalaufnahme:**
   Jeder FSR-Sensor reagiert auf Druckeinwirkung durch eine Widerstandsänderung. Diese Änderung wird über eine Spannungsteiler-Schaltung in eine analoge Spannung übersetzt.

2. **Signalverarbeitung:**
   Der Arduino liest diese Spannungssignale an den analogen Eingängen ein, berechnet anhand der vorher kalibrierten Formel die entsprechende Masse und versendet diese als serielle Zeichenkette über USB.

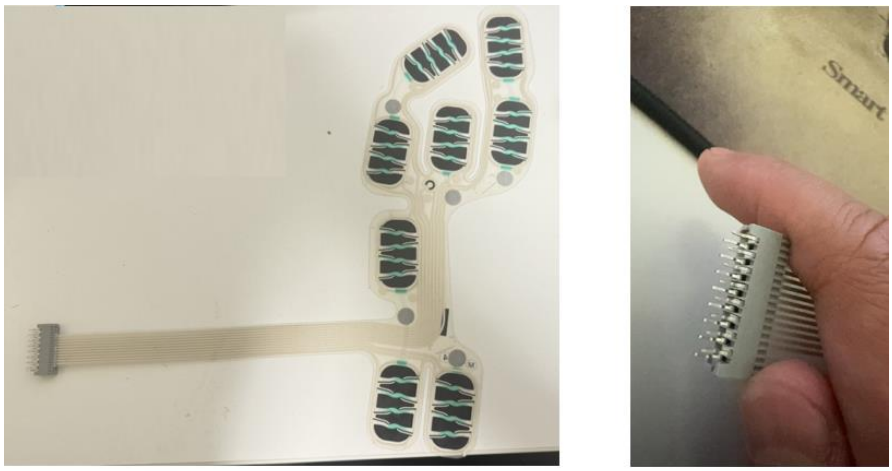
3. **Datenübertragung:**
   Die Daten werden über einen USB-zu-Seriell-Wandler an den PC übertragen. Der Datenstrom enthält identifizierbare Werte für alle acht Sensoren in regelmäßigen Intervallen (~8,6 FPS).

4. **Datenvisualisierung:**
   Eine eigens entwickelte Software in Visual Studio liest die seriellen Daten aus und aktualisiert ein SVG-Bild, wobei die Druckintensität jeder Zone durch die Helligkeit des grünen Farbkanals dargestellt wird.

Durch diesen modularen Aufbau konnten alle Komponenten unabhängig voneinander getestet und bei Bedarf ausgetauscht oder erweitert werden. Gleichzeitig wurde die Möglichkeit geschaffen, das System später um weitere Sensoren oder Visualisierungsfunktionen (z. B. 3D-Druckbild) zu ergänzen.

3.4 Umsetzung und Aufbau

3.4.1 Der interne Schaltungsaufbau des Sensors



(a) Fussmatter (b)Anschlüsse (Foto des Aufbaus)

Abbildung 13 interne Schaltungsaufbau des Sensors

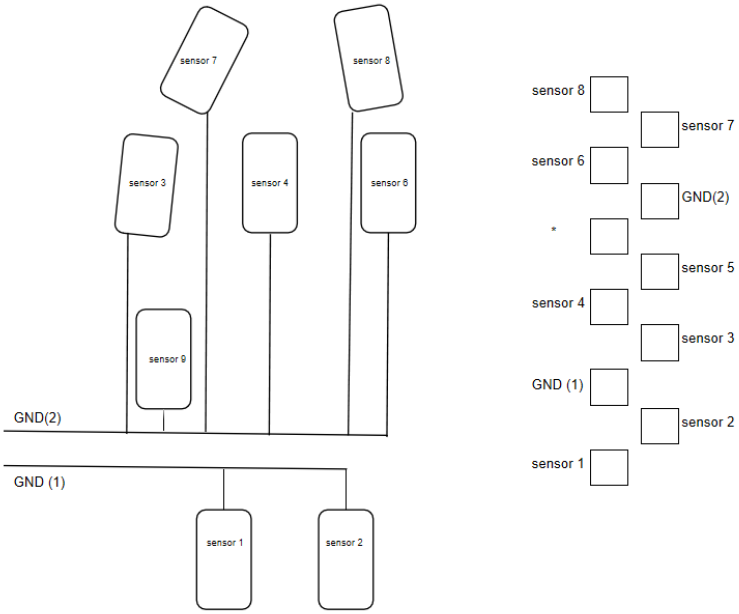


Abbildung 14 Skizze der interne Schaltungsaufbau des Sensors

Die vom Betreuer bereitgestellte elektronische Fußmatte verfügt über insgesamt **zehn Anschlussleitungen**. Davon sind **acht Leitungen** jeweils einem der acht FSR-Sensoren zugeordnet, während die verbleibenden **zwei Leitungen als gemeinsame Masseleitungen (GND)** fungieren.

Zur Optimierung der Verdrahtung wurden die Sensoren in zwei Gruppen unterteilt, die jeweils eine **geteilte Masseverbindung** nutzen:

- Die **obere Zone** der Matte (bestehend aus sechs Sensoren im Vorfuß- und Mittelfußbereich) ist über eine **gemeinsame Masseleitung** verbunden.
- Die **untere Zone** (bestehend aus zwei Sensoren im Fersenbereich) nutzt eine separate **zweite Masseleitung**.

Diese Struktur reduziert den Verkabelungsaufwand, verbessert die Übersichtlichkeit und gewährleistet gleichzeitig eine zuverlässige Referenzmasse für jede Sensorgruppe.

3.4.2 Der elektrische Verschaltungsaufbau des Gesamtsystems

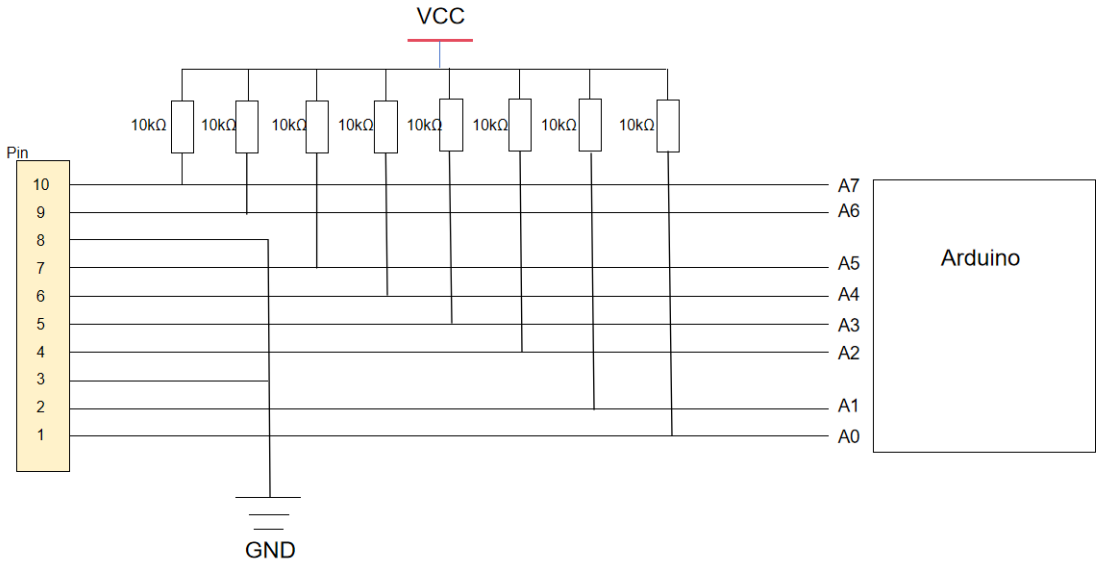


Abbildung 15 elektrische Verschaltungsaufbau des Systems

Die physikalische Verdrahtung des Systems wurde auf Grundlage einer Spannungsteilerschaltung realisiert, wobei jeder FSR-Sensor mit einem Festwiderstand von 10 kΩ verbunden ist. Die gesamte Verschaltung erfolgte auf einem Breadboard und wurde mit einem Arduino Mega 2560 sowie einem PC verbunden. Die Verbindungsmethoden und -strukturen werden im Folgenden detailliert beschrieben.

Verkabelung der Sensorleitungen (A0–A7)

Jede Sensorleitung wurde wie folgt verarbeitet：

1. Zunächst wurden die beiden GND-Leitungen der oberen und unteren Zone auf dem Breadboard auf gemeinsame Masse gelegt（durch Verbindung mit GND des Arduino）.

2. Dann wurde jeder einzelne Sensorausgang mit einem **10 kΩ-Widerstand** in Serie geschaltet，um Spannungsteilerschaltung zu bauen.

3. Die Knotenpunkte zwischen FSR und Widerstand wurden jeweils an einen analogen Eingang (A0 bis A7) des Arduino Mega 2560 angeschlossen：

Tabelle 2 Sensorspostion

Sensor	Arduino-Pin	Position
FSR 1	A0	Ferse links
FSR 2	A1	Ferse rechts
FSR 3	A2	Fußsohle links
FSR 4	A3	Fußsohle mitten
FSR 5	A4	Fußsohle rechts
FSR 6	A5	Zehenspitze links
FSR 7	A6	Zehenspitze rechts
FSR 8	A7	Fußgewölbe

Verbindung zum PC und Spannungsversorgung

Der Arduino Mega wurde über ein **USB-Kabel** mit dem PC verbunden, das gleichzeitig zwei Funktionen erfüllt:

- **Spannungsversorgung** des Arduino-Boards;
- **Serielle Kommunikation** zur Datenübertragung (über COM-Port).

Das System wurde so konzipiert, dass keine externe Stromquelle erforderlich ist – die komplette Stromversorgung erfolgt über den USB-Anschluss.

Um die vom Arduino gemessenen Sensordaten zur weiteren Visualisierung an die PC-Anwendung zu übermitteln, ist die Verbindung zwischen dem Arduino-Board und dem USB-to-Serial-Adapter erforderlich. Dabei wird die serielle Kommunikation wie folgt realisiert:

- Der **TX-Pin** des Arduino wird mit dem **RX-Pin** des Adapters verbunden,
- der **RX-Pin** des Arduino mit dem **TX-Pin** des Adapters,
- sowie **GND (Masse)** auf beiden Seiten miteinander verbunden.

Nur durch diese korrekte Kreuzverschaltung der Datenleitungen ist eine stabile und bidirektionale Datenübertragung zwischen Mikrocontroller und PC gewährleistet.

Foto vom Aufbau

Zur besseren Veranschaulichung des beschriebenen Aufbaus wurde ein Foto der fertigen Testplattform aufgenommen:

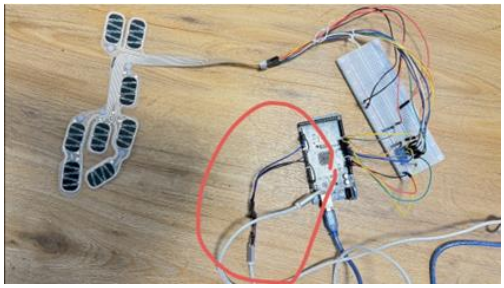


Abbildung 16 endliche Aufbau der alle Elektronische Geräte

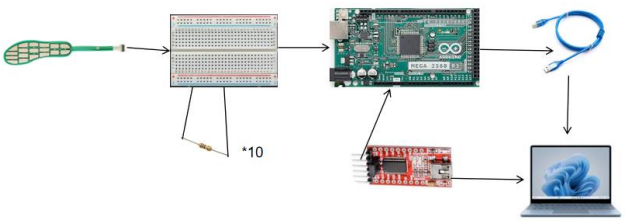


Abbildung 17 Skizze der endliche Aufbau der alle Elektronische Geräte

Hier ist deutlich zu erkennen:

- Die grauen Flachbandleitungen, die von der Fußmatte abgehen;
- Die Aufteilung der Masseleitungen;
- Die übersichtliche Anordnung der 10 kΩ-Widerstände auf dem Breadboard;
- Die Verbindung aller A0–A7 Pins mit den Sensor-Knotenpunkten;
- Die doppelte USB-Verbindung (Strom & Daten) vom Arduino zum PC.

3.4.3 Optimierung von Hardwaresystemen

Obwohl der initiale Aufbau des Systems mit einem Breadboard erfolgreich funktionierte, zeigten sich bei intensiver Nutzung mehrere Einschränkungen, die einen realen Praxiseinsatz – insbesondere im Fahrzeug – deutlich erschwerten. Die provisorische Verkabelung mit Jumper-Wires war mechanisch instabil, anfällig für Wackelkontakte und aufgrund der offenen Struktur nur bedingt transportfähig. Zudem war das System insgesamt sehr sperrig, was das Anlegen der Sensorsohle an den Fuß und das Tragen im Schuh erheblich beeinträchtigte. Die ungeschützte Anordnung der Komponenten auf dem Breadboard führte dazu, dass bei kleinsten Bewegungen oder Vibrationen Verbindungen verloren gingen – ein untragbarer Zustand für dynamische Einsatzszenarien wie eine Fahrzeugintegration.

Um diesen Schwächen zu begegnen und die Alltagstauglichkeit zu verbessern, wurde nach der erfolgreichen Erprobungsphase eine vollständige Systemintegration auf einer Lötplatine vorgenommen. Auf dieser wurden der Arduino-Mikrocontroller, sämtliche Vorwiderstände sowie das USB-zu-Seriell-Modul (FT232) fest verlötet und platzsparend angeordnet. Durch die feste Verdrahtung konnten potenzielle Kontaktprobleme eliminiert werden. Die resultierende Trägerplatine war nicht nur deutlich robuster und kompakter, sondern ließ sich auch problemlos in einem Fahrzeug oder in tragbaren Anwendungen integrieren.

Die finale Ausführung des Gesamtsystems ist somit mechanisch stabil, vibrationssicher und erheblich benutzerfreundlicher als der ursprüngliche Aufbau. Eine Abbildung der fertigen Platine findet sich in Abbildung 12.

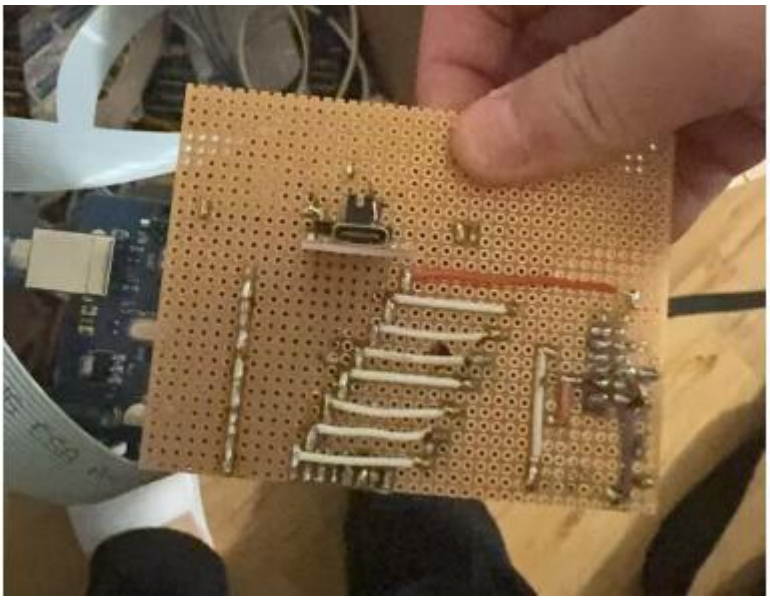


Abbildung 18 vollständige Systemintegration auf einer Lötplatine

3.5 Programmierung und Visualisierungssystem

Obwohl der elektronische Fußsensoraufbau aus acht äußerlich identischen FSR-Sensoren besteht, wurde zur Sicherstellung wissenschaftlicher Genauigkeit jeder einzelne Sensor separat getestet. Ziel war es, die individuelle Widerstands-Masse-Kennlinie experimentell zu überprüfen.

Da der Arduino direkt die Sensorspannung erfassen kann, wurde im Rahmen der Voruntersuchungen eine einfache, aber effektive Methode angewandt: Durch mehrfaches Aufbringen definierter Gewichte konnten Spannungswerte erfasst und über das Spannungsteilerprinzip in Widerstandswerte umgerechnet werden. Auf diese Weise ließ sich prüfen, ob alle Sensoren ein vergleichbares elektrisches Verhalten aufwiesen.

	2kg	2.654kg	5.129kg	7.78kg	10.2kg	15.4kg
sensor1	3.91V	3.68V	3.28V	2.85V	2.67V	1.83V
sensor2	3.92V	3.65V	3.27V	2.86V	2.68V	1.82V
sensor3	3.93V	3.66V	3.27V	2.85V	2.67V	1.80V
sensor4	3.90v	3.67V	3.25V	2.83V	2.68V	1.82V
sensor5	3.91V	3.69V	3.26V	2.84V	2.69V	1.85V
sensor6	3.89V	3.68V	3.28V	2.85V	2.68V	1.83V
sensor7	3.97V	3.66V	3.27V	2.86V	2.67V	1.84V
sensor8	3.93V	3.37V	3.28V	2.86V	2.67V	1.84V

Abbildung 19 Alle Sensorspannungsaufzeichnungen in unterschiedlicher Qualität

Die Versuchsergebnisse zeigten, dass alle Sensoren eine nahezu identische Druck-Widerstands-Kennlinie aufweisen. Daher wurde im Rahmen der Implementierung auf eine einheitliche Kalibrierformel für sämtliche Sensoren zurückgegriffen.

3.5.1 Kalibrierung und Modellbildung in MATLAB

Um die gemessenen Analogwerte der FSR-Sensoren in physikalisch interpretierbare Größen wie Masse (bzw. Kraft) umzuwandeln, wurde zunächst eine experimentelle Kalibrierung durchgeführt. Da FSR-Sensoren ein deutlich nichtlineares Widerstandsverhalten aufweisen, ist eine direkte lineare Zuordnung nicht praktikabel. Ziel war es daher, eine mathematische Modellierung zu finden, die den Zusammenhang zwischen dem gemessenen Widerstand und der tatsächlich aufgebrachten Masse möglichst genau abbildet.

Experimenteller Versuchsaufbau

Zur Kalibrierung wurde ein einzelner FSR-Sensor auf einer stabilen Oberfläche fixiert und schrittweise mit bekannten Gewichten (2 kg – 15,4 kg, in Form von Hantelscheiben) belastet. Währenddessen wurde der Widerstandswert des Sensors mithilfe eines Arduino-Mikrocontrollers gemessen und an den PC übertragen. Jeder Messwert wurde mehrfach aufgenommen, um Messfehler durch Schwankungen oder Kontaktprobleme zu minimieren.

Zur Bestimmung des jeweiligen Sensorwiderstands wurde die bekannte Spannungsteilerformel verwendet. Da der FSR in Serie mit einem 10 kΩ-Widerstand geschaltet war und die am FSR abfallende Spannung $\mathrm{V_{aut}}$ durch den Arduino ausgelesen werden konnte, ergibt sich die Formel zur Berechnung von $\mathrm{R_{FSR}}$ wie folgt:

$$\mathrm{R_{FSR}} = \left(\frac{5 - \mathrm{V_{aut}}}{\mathrm{V_{aut}}}\right) * 10000$$

Der Wer t$\mathrm{V_{aut}}$ für wurde dabei über die Funktion $\mathrm{analogRead()}$ in Kombination mit dem bekannten Referenzwert $\mathrm{V_{in}} = 5{,}0$ V bestimmt.

Der Wert für $\mathrm{V_{aut}}$ wurde dabei aus dem Analogwert mit folgender Gleichung berechnet:

$$V_{aut} = \text{analogRead}() * \frac{5.0}{1023}$$

Messdatenübersicht

Die folgende Tabelle zeigt die gemessenen Widerstandswerte bei den eingesetzten Gewichten. Zu jedem Gewicht wurden 6 bis 8 Wiederholungen durchgeführt:

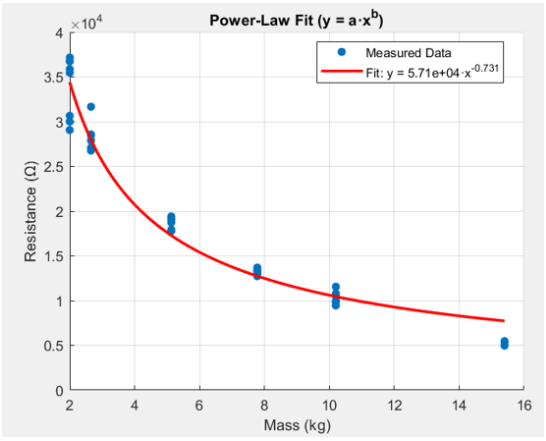
	2kg	2.654kg	5.129kg	7.78kg	10.2kg	15.4kg
1.Versuch	35871.56Ω	27878.79Ω	19069.77Ω	13255.81Ω	11551.72Ω	5749.88Ω
2.Versuch	36728.97Ω	27878.79Ω	19411.76Ω	13696.86Ω	10000.0Ω	4970.06Ω
3.Versuch	37169.81Ω	31666.67Ω	17777.78Ω	12727.27Ω	9455.25Ω	5015.02Ω
4.Versuch	29062.5Ω	28461.54Ω	17932.96Ω	13696.68Ω	10576.13Ω	4970.06Ω
5.Versuch	30650.41Ω	26764.71Ω	18735.63Ω	13041.47Ω	9841.27Ω	5015.02Ω
6.Versuch	35454.55Ω	27090.91Ω	19230.77Ω	13425.15Ω	10816.9Ω	4970.06Ω
7.Versuch	30000Ω	28571.43Ω	19047.62Ω	13223.74Ω	10576.13Ω	5375.93Ω

Abbildung 20 Messdatenübersicht der Beziehung zwischen Masse und Widerstand

Datenauswertung und Kurvenanpassung

Die aufgenommenen Wertepaare (Masse – Widerstand) wurden in MATLAB importiert und dort mittels des Curve Fitting Tool (cftool) analysiert. Zur Erhöhung der Genauigkeit der Auswertung wurden zwei unterschiedliche Funktionen zur nichtlinearen Kurvenanpassung getestet: eine Potenzfunktion sowie ein invers proportionales Modell. Die finale Potenzfunktion Modell lautet:

$$M = \left(\frac{R}{a}\right)^{\frac{1}{b}}$$



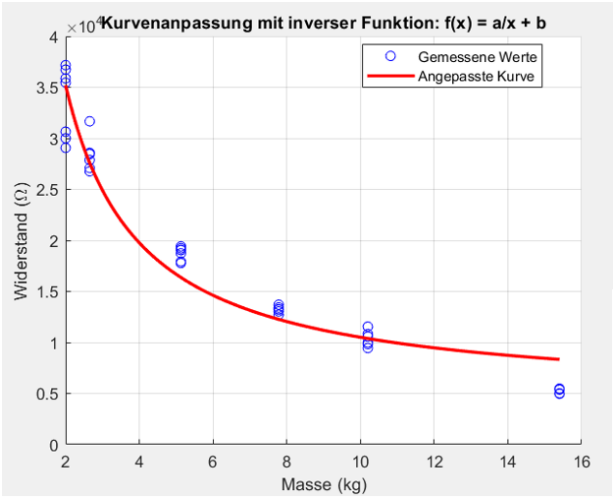

```
Power-law fit result (power1:  y = a * x^b)
     General model Power1:
     f_obj(x) = a*x^b
     Coefficients (with 95% confidence bounds):
       a =   5.711e+04  (5.343e+04, 6.079e+04)
       b =    -0.7311  (-0.7833, -0.6788)
Goodness of Fit:
          sse: 1.7018e+08
      rsquare: 0.9618
          dfe: 44
    adjrsquare: 0.9609
         rmse: 1.9666e+03
```

Abbildung 21 Anpassen der Daten an einen Potenzfunktion Modell in Matlab

Dabei ist R der gemessene Widerstand, a ein Skalierungsfaktor und b der Exponent, der die Krümmung der Funktion bestimmt. Für einen Beispiel-Sensor ergaben sich die Werte:

$$a = 5{,}711 \times 10^4, b = -0{,}7311$$

Das finale invers proportionale Modell lautet:




```
Anpassung mit inverser Funktion (a/x + b):
     General model:
     f_obj(x) = a./x + b
     Coefficients (with 95% confidence bounds):
       a =   6.173e+04  (5.718e+04, 6.627e+04)
       b =        4331  (3091, 5572)
Güte der Anpassung:
          sse: 2.4683e+08
      rsquare: 0.9446
          dfe: 44
    adjrsquare: 0.9433
         rmse: 2.3685e+03
```

Abbildung 22 Anpassen der Daten an einem invers proportionalen Modell in Matlab

$$a = 6.173 \times 10^4, b = 4331$$

Durch die Analyse der in MATLAB erhaltenen Fit-Ergebnisse lässt sich feststellen, dass das Potenzfunktionsmodell eine deutlich bessere Anpassungsgüte aufweist als das Modell der umgekehrt proportionalen Funktion. Unter dem Potenzfunktionsmodell ist der SSE-Wert geringer, der R^2-Wert liegt näher bei 1, ebenso wie das adjustierte

R^2. Darüber hinaus ist auch der RMSE-Wert im Vergleich zum Modell der umgekehrt proportionalen Funktion kleiner.

Daher habe ich mich abschließend dafür entschieden, das Potenzfunktionsmodell in meinem Code als Grundlage für die Datenerfassung zu verwenden.

Export der Ergebnisse für Mikrocontroller

Die finalen Kalibrierfunktionen wurden in C++-kompatible Form umgewandelt und in die Arduino-Umgebung integriert. Dort übernimmt eine Funktion $\mathrm{calculateMass(float\ resistance)}$ die Auswahl des passenden Funktionssegments und die Berechnung des Gewichts für jeden Sensorwert in Echtzeit.

3.5.2 Implementierung auf dem Mikrocontroller (Arduino)

Die auf Grundlage der MATLAB-Analyse entwickelten Kalibrierfunktionen wurden vollständig in die Arduino-Umgebung übertragen, um eine direkte und echtzeitfähige Umrechnung der Sensordaten zu ermöglichen. Die zentrale Aufgabe des Mikrocontrollers besteht darin, die analogen Eingangssignale in Widerstandswerte umzuwandeln, daraus Massen zu berechnen und diese per serieller Schnittstelle an den PC zu übertragen.

Spannungserfassung und Widerstandsberechnung

Die acht analogen Eingangssignale der Sensoren werden zyklisch über die Funktion $\mathrm{analogRead()}$ eingelesen. Die gemessenen Werte (0–1023) werden in Spannung umgerechnet:

$$\mathrm{V_{aut}} = \mathrm{analogRead()} * \frac{5.0}{1023}$$

Anschließend wird auf Basis der bekannten Spannungsteiler-Schaltung mit 10 kΩ Serienwiderstand der Sensorwiderstand wie folgt berechnet:

$$\mathrm{R_{FSR}} = \left(\frac{5 - \mathrm{V_{aut}}}{\mathrm{V_{aut}}}\right) * 10000$$

Massenberechnung und Segmentlogik

Basierend auf der in Kapitel 3.5.1 durchgeführten MATLAB-Kurvenanpassung wurde eine kontinuierliche Kalibrierformel der Form implementiert. Die Konstanten wurden wie folgt übernommen:

- $\mathrm{a} = 5{,}711 \times 10^4$
- $\mathrm{b} = -0{,}7311$

Die Umrechnung erfolgt direkt auf dem Mikrocontroller über eine einfache Potenzfunktion:

$$\begin{gathered}\text{float caculateMasse(float resistance)\{} \\ \text{const float a} = 57110.0; \\ \text{const float b} = -0.7311; \\ \text{return pow (resistance /a, 1.0f/b)}\end{gathered}$$

Ein typischer Ablauf im Hauptprogramm könnte wie folgt aussehen:

$$\begin{gathered}\text{float voltage} = \text{analogRead(A0)} * \frac{5.0}{1023.0}; \\ \text{float resistance } = \frac{5.0-\text{voltage}}{\text{voltage}} * 10000; \\ \text{float mass} = \text{caculateMasse(resistance)};\end{gathered}$$

Alle Sensorwerte werden in einem Array $masses[]$ gespeichert und zur seriellen Ausgabe vorbereitet.

Serielle Ausgabeformatierung

Die berechneten Massewerte aller acht Sensoren werden als formatierter String über die serielle Schnittstelle (Serial1) an den PC übermittelt. Die Ausgabe erfolgt in folgender Struktur:

$$\text{Mass1: } 3.24, \text{Mass2: } 4.15, \ldots\ldots, \text{Mass8: } 2.88$$

Das System aktualisiert diese Daten kontinuierlich in festen Intervallen (~200 ms), was einer Übertragungsfrequenz von etwa 5 Hz entspricht. Somit entsteht die technische Grundlage für die nachfolgende Visualisierung am PC.

3.5.3 Visualisierung in Visual Studio

3.5.3.1 Ziel und Aufbau der Software

Um die in Echtzeit erfassten Massewerte der FSR-Sensoren anschaulich und intuitiv darzustellen, wurde eine benutzerdefinierte PC-Anwendung in **C# mittels Visual Studio** entwickelt. Die Software dient als **Schnittstelle zur Visualisierung**, zur Live-Datenanalyse und zur Protokollierung der seriell übertragenen Sensordaten.

Die Oberfläche basiert auf dem Windows-Forms-Framework und gliedert sich in mehrere zentrale Bereiche:

- Ein **Grafikbereich** (panel1) zur Darstellung der Fußmatte in Form von acht Ellipsen, die jeweils einem FSR-Sensor zugeordnet sind;
- Ein **Tabellenbereich** (DataGridView) zur synchronen Anzeige der numerischen Massewerte;
- Eine **Steuerleiste** mit Auswahl- und Bedienmöglichkeiten für den COM-Port, Verbindungsstatus, Start/Stop und Dateispeicherung;
- Ein **Protokollfeld** zur Anzeige aller eingehenden Rohdaten.

Diese modulare Struktur ermöglicht es dem Benutzer, **Daten zu empfangen, visuell auszuwerten und langfristig zu speichern**, ohne zusätzliche externe Tools oder Libraries.

3.5.3.2 COM-Kommunikation und Datenempfang

Die Kommunikation zwischen dem Arduino-Mikrocontroller und der Visualisierungssoftware erfolgt über eine serielle USB-Schnittstelle. In der C#-Anwendung wird diese Verbindung mithilfe der Klasse SerialPort aus dem .NET-Framework realisiert.

Portauswahl und Initialisierung

Beim Start der Anwendung wird über die Methode SerialPort.GetPortNames() eine Liste aller verfügbaren COM-Ports abgerufen und im Auswahlmenü angezeigt. Der Benutzer kann über das Drop-Down-Menü den gewünschten Port auswählen. Durch Klicken auf den **„Verbinden“-Button** wird der Port mit einer festen Baudrate von **9600 Baud** geöffnet.

$$
\begin{aligned}
&\{\\
&serialPort.PortName = selectedPort;\\
&serialPort.BaudRate = 9600;\\
&serialPort.Open(\ \ )\\
&\}
\end{aligned}
$$

Fehlerhafte Portverbindungen werden durch entsprechende Ausnahmebehandlungen abgefangen und über MessageBox gemeldet.

Empfang der Sensordaten

Die Sensordaten werden vom Arduino im Format einer ASCII-codierten Zeichenkette übertragen, beispielsweise:

```yaml
Mass1: 2.35, Mass2: 3.21, Mass3: 1.87, ...
```

Abbildung 23 Zeichenkette der Masse

Sobald neue Daten empfangen werden, löst der Port das DataReceived-Event aus. Die Methode seria lPort1_DataReceived liest die Zeichenkette ein und übergibt sie zur Verarbeitung an die Methode UpdateData().

Datenverarbeitung und Synchronisierung

Da der DataReceived-Event in einem separaten Thread läuft, erfolgt die Übergabe an das GUI mittels BeginInvoke(), um Threadkonflikte zu vermeiden. Die eingelesenen

Werte werden mithilfe von Split() und Replace() aus dem String extrahiert und in ein float[] -Array mit acht Elementen überführt. Diese Werte dienen anschließend sowohl der tabellarischen Darstellung als auch der grafischen Visualisierung im Panel.

3.5.3.3 Darstellung der Druckverteilung (Panel1)

Die visuelle Repräsentation der Druckverteilung erfolgt im zentralen Grafikbereich der Anwendung, welcher durch das Windows-Forms-Steuerelement panel1 realisiert wird. Jedes der acht Messfelder wird durch eine **Ellipsenform** symbolisiert, die einer physischen Zone auf der Fußmatte entspricht. Die Farbintensität jeder Ellipse repräsentiert dabei den Druckwert bzw. die berechnete Masse.

Zeichenroutine und Initialisierung

Die Zeichenroutine ist im Paint-Event des Panels verankert und wird durch folgende Initialisierung aktiviert:

$$this.panel1.Paint \mathrel{+}= new\ System.Windows.Forms.PaintEventHandler(this.DrawEllipses);$$

Beim erstmaligen Laden wird eine leere Ellipsenstruktur erzeugt und gespeichert. Die acht Ellipsen sind in der Methode GenerateEllipses() vordefiniert, wobei jede eine bestimmte Position und Größe im Koordinatensystem erhält. Dies geschieht über ein RectangleF[]-Array mit festen Werten zur geometrischen Platzierung.

Farbkodierung und Aktualisierung

Die Methode DrawEllipses(Graphics g) zeichnet bei jedem Update-Zyklus die acht Ellipsen neu. Die Farbkodierung basiert auf den aktuellen Massenwerten aus dem Array float[] masses. Für die Farbdarstellung wird ausschließlich der **grüne Farbkanal (RGB)** verwendet, wobei gilt:

$$\text{Farbwert} = 255 - (\text{Masse} \cdot Skalierungsfaktor)$$

Dieser Wert wird dynamisch berechnet und zur Füllfarbe der jeweiligen Ellipse verwendet:

$$intensity = Math.Max\big(0, Math.Min(25, intensity)\big);$$
$$int\ green = (int)(minGreen + (intensity\ /\ 25.0f) * (255 - minGreen));$$

je höher die Masse, desto **dunkler** die Farbe. Druckfreie Bereiche bleiben in leuchtendem Grün.

Synchronisierung mit Daten

Nach jedem erfolgreichen Dateneingang über die serielle Schnittstelle wird `panel1.Invalidate()` aufgerufen. Dies triggert die `Paint`-Methode erneut, wodurch das Bild in Echtzeit aktualisiert wird – in synchronem Takt mit den übermittelten Werten.

3.5.3.4 Tabellenanzeige der Rohdaten

Neben der grafischen Visualisierung der Druckverteilung bietet die Anwendung eine tabellarische Darstellung der Sensordaten. Diese erfolgt über das Windows-Forms-Steuerelement `DataGridView`, das im rechten Bereich des Fensters positioniert ist. Es dient primär der **numerischen Kontrolle** der von den Sensoren erfassten Massewerte und unterstützt die Nachvollziehbarkeit der Farbzuordnung im Panel.

Initialisierung und Struktur

Die Tabelle wird beim Start der Anwendung durch die Methode `InitTable()` konfiguriert. Dabei werden acht Zeilen erzeugt, die jeweils einem FSR-Sensor (A0 bis A7) entsprechen. Die erste Spalte enthält die Sensorbezeichnung, die zweite Spalte zeigt den jeweils aktuellen Massewert an:

$$dataGridView1.Rows[i].Cells[0].Value = \$\text{Sensor A\{i\}}";$$

Laufende Aktualisierung

Nach jedem Eingang eines neuen Datenpakets über die serielle Schnittstelle wird nicht nur das Panel aktualisiert, sondern auch die Tabelle mit den neuesten Messwerten befüllt:

$$dataGridView1.Rows[i].Cells[0].Value = masses[i].ToString("F2");$$

Dabei wird der Wert auf zwei Nachkommastellen gerundet dargestellt. Die Aktualisierung erfolgt synchron zur grafischen Darstellung, sodass Tabelle und SVG-Ansicht stets den gleichen Datenstand repräsentieren.

Vorteile der tabellarischen Ansicht

Die tabellarische Anzeige bietet eine schnelle Möglichkeit zur Fehlerdiagnose und zur quantitativen Auswertung. Insbesondere bei der Validierung der Kalibrierung, der Erkennung von Ausreißern oder bei der Entwicklung alternativer Farbschemata liefert die numerische Darstellung wertvolle Hinweise.

3.5.3.5 Logging-Funktion

Zur langfristigen Datensicherung und zur nachträglichen Analyse der Messwerte bietet die Anwendung eine automatische Protokollierungsfunktion. Alle über die serielle Schnittstelle empfangenen Datenpakete werden dabei zeilenweise in eine Logdatei geschrieben, ohne die Echtzeitdarstellung zu beeinträchtigen.

Ablauf der Protokollierung

Die Funktion wird beim Klick auf den **„Start Logging“**-Button aktiviert. Dabei wird ein neuer StreamWriter geöffnet und eine neue Textdatei erstellt. Der Dateiname basiert auf dem aktuellen Zeitstempel, sodass jede Sitzung eindeutig identifiziert werden kann :

$$string\ timestamp\ =\ DateTime.Now.ToString("yyyy - MM - dd\ HH{:}mm{:}ss");$$

Anschließend wird jede empfangene Zeile innerhalb der Methode `serialPort1_DataReceived()` zusätzlich zum Visualisierungsupdate auch in die Datei geschrieben.

Speicherort und Dateistruktur

Die Logdateien werden im Unterordner `Logs` im Arbeitsverzeichnis der Anwendung abgelegt.

Diese Dateien lassen sich anschließend mit MATLAB, Excel oder Python weiterverarbeiten, um z. B. **Langzeitverläufe, Mittelwerte oder Standardabweichungen** zu berechnen.

Abschluss und Dateischluss

Beim Beenden der Messung (z. B. durch Klick auf „Stop Logging“) wird der StreamWriter sauber geschlossen, um Dateibeschädigungen zu vermeiden. Die Anwendung prüft außerdem automatisch, ob ein aktiver Logger vorhanden ist, bevor Schreibvorgänge ausgeführt werden.

3.5.3.6 Benutzerinteraktion und Steuerung

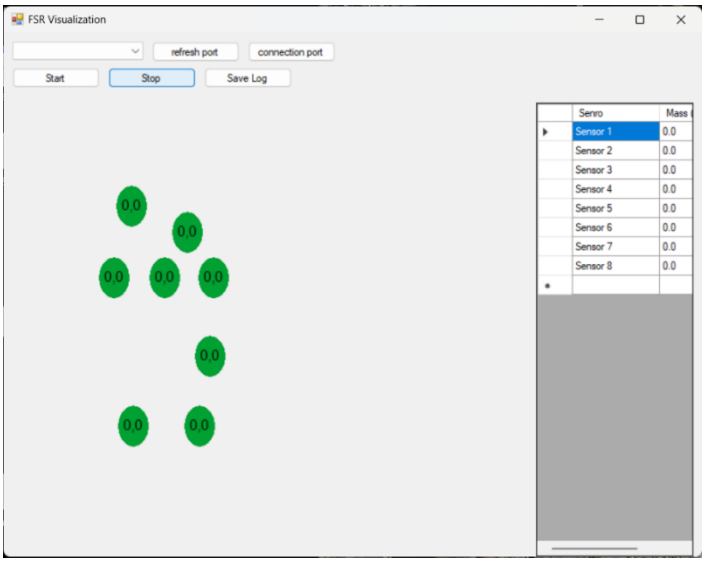


Abbildung 24 endgültige Modell des Host-PCs

Die Software wurde so konzipiert, dass sie eine möglichst einfache und intuitive Interaktion mit dem Benutzer ermöglicht. Alle wichtigen Funktionen der Anwendung lassen sich über eine zentrale Steuereinheit am oberen Fensterrand bedienen. Diese umfasst sowohl die Steuerung der Datenverbindung als auch die Kontrolle der Visualisierung und der Protokollierung.

Verbindungsaufbau und Portwahl

Der Benutzer kann über ein Dropdown-Menü den gewünschten seriellen COM-Port auswählen. Die verfügbaren Ports werden automatisch durch einen **„Refresh"-Port** neu geladen. Nach Auswahl des Ports kann die Verbindung durch einen Klick auf **„Connection port"** hergestellt werden. Ist die Verbindung aktiv, wechselt der Button zu **„Trennen"**.

Start/Stopp der Datenerfassung

Nach erfolgreicher Verbindung kann die Datenerfassung über einen **„Start"-Button** gestartet werden. Die Software beginnt daraufhin automatisch mit dem Empfang, der Verarbeitung und der Darstellung der Daten. Die laufende Verbindung kann jederzeit durch **„Stopp"** unterbrochen werden, um das System anzuhalten oder neue Einstellungen vorzunehmen.

Logging-Kontrolle

Die Speicherung der eingehenden Daten kann optional durch den Button **„Start"** aktiviert werden. Ein erneuter Klick auf **„Stop"** beendet die Aufzeichnung. Der Status der Protokollierung wird visuell angezeigt.

Fehlerbehandlung und Benutzerrückmeldung

Alle relevanten Aktionen sind mit Fehlerabfragen versehen, um etwaige Probleme (z. B. fehlender COM-Port, Verbindungsfehler, Datei-Zugriffsprobleme) über MessageBox klar und benutzerfreundlich mitzuteilen. Dies erhöht die Robustheit der Software und verhindert unbeabsichtigte Programmabbrüche.

Zusammenfassung der Benutzeroberfläche

Die vollständige Kontrolle über die Messung und Visualisierung erfolgt mit nur wenigen Mausklicks. Dadurch ist das System nicht nur funktional, sondern auch für nicht-technische Nutzer einfach bedienbar – ein wichtiger Aspekt für mögliche spätere Einsatzbereiche z. B. in der Fahrerprobung oder im Labor.

3.6 Technische Varianten und Erweiterungsmöglichkeiten

Im Rahmen dieser Arbeit wurde ein funktionsfähiges Basissystem zur Druckmessung mit acht FSR-Sensoren und einer kabelgebundenen Visualisierungseinheit erfolgreich realisiert. Darauf aufbauend lassen sich verschiedene technische Erweiterungen und Alternativen skizzieren, die im Rahmen zukünftiger Projekte oder bei industrieller Weiterentwicklung des Systems eine Rolle spielen könnten.

Skalierung der Sensoranzahl

Die aktuelle Architektur ist auf acht Messkanäle begrenzt. Durch den Einsatz von analogen Multiplexern (z. B. CD74HC4067) könnte die Anzahl der erfassbaren Sensorzonen auf 16, 32 oder mehr erweitert werden. Dies würde insbesondere für Anwendungen mit großflächiger Druckverteilung (z. B. Sitz- oder Mattenanalyse) zusätzliche Einblicke bieten. Die Arduino MEGA Plattform bietet dafür ausreichend digitale Steuerpins.

Alternative Mikrocontroller

Während der Arduino MEGA in dieser Arbeit ausreichend war, könnten zukünftige Versionen auf leistungsfähigere Mikrocontroller wie den ESP32 oder STM32 umgestellt werden. Diese bieten deutlich höhere Rechenleistung, integrierte Kommunikationsschnittstellen (WLAN/BLE) und ermöglichen komplexere Algorithmen direkt auf dem Gerät (Edge Computing). Dadurch ließen sich erweiterte Datenanalysen (z. B. Glättung, Mustererkennung) bereits vor der Datenübertragung durchführen.

Drahtlose Datenübertragung

Die serielle USB-Verbindung wurde in dieser Arbeit aus Gründen der Einfachheit gewählt. In einer mobilen oder praxisnahen Umgebung wäre eine drahtlose Lösung

jedoch deutlich vorteilhafter. Denkbar wären Bluetooth Low Energy (BLE) oder WiFi-Module zur Übertragung der Sensordaten an ein Tablet oder Smartphone. Dies würde die Bewegungsfreiheit erhöhen und Kabelsalat im Fahrerfußraum vermeiden.

Integration von Displays und Benutzerinteraktion

Das derzeitige System ist auf die Visualisierung am PC angewiesen. In zukünftigen Ausbaustufen könnten kleine OLED- oder TFT-Displays direkt auf der Sensoreinheit verbaut werden, um die Druckwerte lokal in Echtzeit anzuzeigen. Dadurch könnte das System als eigenständige Diagnoseeinheit ohne PC funktionieren. Auch Touch-Interfaces zur Modifikation von Anzeigeparametern wären denkbar.

Systemmodularität und Plattformtransfer

Langfristig könnte die Systemarchitektur modular gestaltet werden, sodass einzelne Komponenten (Datenerfassung, Übertragung, Visualisierung) unabhängig austauschbar sind. Dies würde eine Anpassung des Systems an unterschiedliche Einsatzbereiche ermöglichen, z. B. in der Rehabilitationsmedizin, im Sportbereich oder in der Ergonomieforschung.

Alle hier skizzierten Varianten wurden im Rahmen dieser Arbeit **nicht implementiert**, erscheinen jedoch aus heutiger Sicht technisch umsetzbar und könnten in weiterführenden Entwicklungsphasen konkrete Gestalt annehmen.

4. Ortsaufgelöste Kraftmessung am Beispiel Bremspedal

4.1 Messszenario

Um die Praxistauglichkeit und Reaktionsfähigkeit des entwickelten Druckmesssystems unter realitätsnahen Bedingungen zu überprüfen, wurde ein gezieltes Testszenario im Rahmen einer realen Fahrzeugfahrt durchgeführt. Dabei lag der Fokus auf der Untersuchung des Kraftverhaltens auf dem Bremspedal während eines dynamischen Notbremsvorgangs. Zusätzlich sollte analysiert werden, inwiefern ergonomische Unterschiede in der Sitzposition messbare Auswirkungen auf die Druckverteilung haben.

Motivation für die Sitzpositionen

Die Auswahl von drei unterschiedlichen Fahrpositionen basiert auf den Prinzipien der ergonomischen Sitzanpassung im Fahrzeug. Die Interaktion zwischen Mensch und Pedal wird stark durch die Sitzdistanz beeinflusst, insbesondere bei plötzlichen Reaktionshandlungen wie einer Notbremsung. Um die Messvariabilität realistisch einzufangen, wurden daher folgende Positionen definiert:

Abbildung 25 3 unterschiedliche versuchende Sitzposition

1. Der Sitz wurde maximal nach vorne verschoben, wodurch die Beine stark angewinkelt waren. Diese Haltung tritt häufig bei kleineren Fahrern auf oder bei falscher Sitzeinstellung;

2. Der Sitz befand sich in einer mittleren Position, die einer standardisierten ergonomischen Haltung entspricht. Diese dient als Referenzposition;

3. Der Sitz wurde vollständig zurückgestellt, was zu nahezu ausgestreckten Beinen führte. Diese Haltung wird in der Praxis häufig bei langen Fahrten oder aus Komfortgründen gewählt.

Ziel dieser drei Szenarien war es, den Einfluss der Beinwinkelung und Körperposition auf die Pedalkraftübertragung und deren Verteilung zu untersuchen.

Durchführung der Notbremsung

Die Testperson nahm im Fahrerbereich eines Fahrzeugs Platz und passte die Sitzposition gemäß der jeweiligen Konfiguration an. Im Anschluss wurde das Fahrzeug auf eine Geschwindigkeit von ca. **50–60 km/h** beschleunigt. Sobald die Zielgeschwindigkeit erreicht war, erfolgte eine **plötzliche Notbremsung**, wobei die vollständige Kraft auf das Bremspedal ausgeübt wurde.

Die speziell entwickelte Sensorsohle war unter dem Schuh positioniert und zeichnete während des Bremsvorgangs die Druckverteilungen in den acht Zonen des Fußes auf.

Datenerfassung und Synchronisierung

Die von den FSR-Sensoren ermittelten Widerstandswerte wurden in Echtzeit über den Arduino-Mikrocontroller verarbeitet, in Masseschätzungen umgerechnet und über die serielle USB-Schnittstelle an die Visualisierungssoftware übertragen. Diese zeigte synchron zur Bewegung die entsprechenden Druckwerte sowohl tabellarisch als auch grafisch an. Zusätzlich wurden die Originaldaten zeilenweise in einer Logdatei gespeichert, um eine nachträgliche Analyse zu ermöglichen.

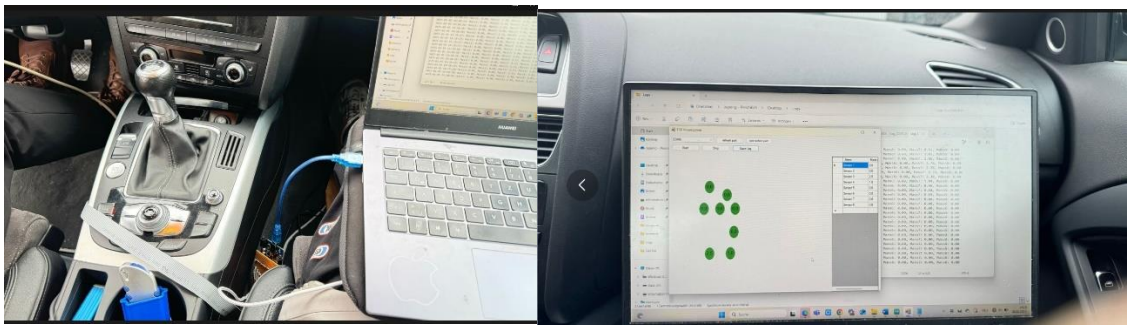


Abbildung 26 Datenerfassung in Experiment

Wiederholbarkeit und Versuchsplanung

Jedes der drei Sitzszenarien wurde **mindestens dreimal** mit identischer Testperson durchgeführt, um Schwankungen durch individuelle Körpermaße auszuschließen. Die Fahrbedingungen (Straßenbelag, Wetter, Reifendruck) wurden konstant gehalten. Jede Bremsung wurde klar signalisiert, um Reaktionszeitverzerrung zu minimieren.

Mit dieser Methodik konnte gewährleistet werden, dass die Daten vergleichbar, repräsentativ und für eine spätere statistische Auswertung geeignet waren.

4.2 Aufbau und Durchführung

Zur Durchführung der Kraftmessung am Bremspedal wurde das in Kapitel 3 entwickelte System vollständig in ein reales Fahrzeuginneres integriert. Ziel dieser Phase war es, eine präzise und verlustfreie Erfassung der Druckverteilung während des Bremsvorgangs sicherzustellen – sowohl hardware- als auch softwareseitig.

Montage der Sensormatte im Fahrzeug

Die Sensorsohle mit acht Force-Sensing-Resistoren (FSRs) wurde in den Schuh der Testperson integriert und während der Fahrt getragen. Durch diese Lösung konnte eine direkte und verlustarme Übertragung der Pedalkräfte auf die Sensorik gewährleistet werden, ohne das Bremspedal physisch zu verändern oder die Sicherheit zu beeinträchtigen. Die flexible Bauweise der Sensorfolie erlaubte eine ergonomische Anpassung an die Fußform und verschiedene Sitzhaltungen.

Abbildung 27 Montage der Sensormatte mit Schuh

Elektronische Verkabelung und Signalfluss

Die acht FSR-Sensoren wurden gemäß dem in Kapitel 3.3 beschriebenen Schaltplan mit einem **Arduino Mega 2560** verbunden. Jeder Sensor war über einen Spannungsteiler (10 kΩ Widerstand) mit einem analogen Eingang (A0–A7) des Mikrocontrollers verbunden. Zwei GND-Leitungen – jeweils für den vorderen und hinteren Fußbereich – wurden über das Breadboard verbunden und gemeinsam auf GND-Pin des Arduino geführt.

Der Arduino wurde über einen USB-TTL-Adapter (FT232) mit dem Laptop verbunden, wodurch sowohl die Stromversorgung als auch die serielle Datenübertragung gesichert waren. Zur vollständigen Visualisierung wurde zusätzlich die serielle Schnittstelle im Code aktiviert und mit einer festen Baudrate von **9600 Baud** betrieben

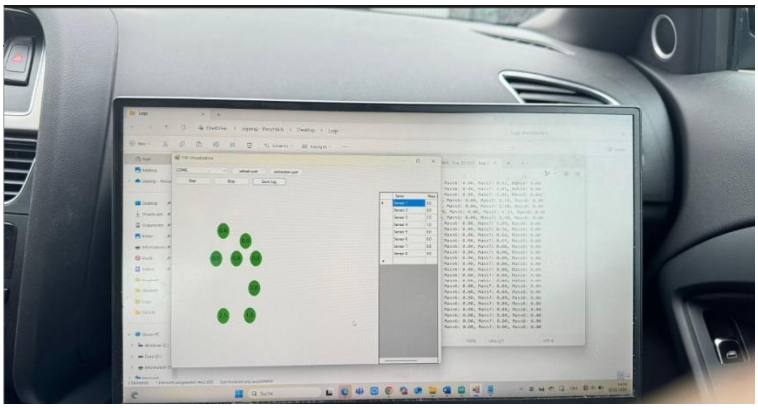


Abbildung 28 Elektronische Verkabelung und Signalfluss

Datenerfassung mit definierter Samplingfrequenz

Während jeder Notbremsung begann der Arduino mit der kontinuierlichen Messung der Spannung an den acht analogen Eingängen. Die Werte wurden mit einer Frequenz von ca. **20 Hz** (entspricht 20 Messungen pro Sekunde) erfasst und über die serielle Schnittstelle an die Visual Studio Anwendung gesendet.

Die Software übernahm im Hintergrund:

- **Umwandlung der Rohdaten** in Massewerte (mithilfe der inversen Kalibrierungsfunktion);
- **Echtzeit-Darstellung** in der SVG-Fläche (Panel);
- **Parallel-Protokollierung** in eine Logdatei mit Zeitstempel.

Synchronität und Echtzeitverhalten

Die Kommunikation und Darstellung erfolgte praktisch verzögerungsfrei. Die visuelle Aktualisierung lag bei etwa **8–10 FPS**, mit einer durchschnittlichen Latenzzeit von unter **150 ms** zwischen dem physischen Tritt auf das Pedal und der Darstellung im System.

Testdurchführung

Für jede der drei definierten Sitzpositionen (kompakt, komfortabel, maximaler Beinraum) wurde der Aufbau erneut überprüft und sichergestellt, dass keine Leitungen verrutscht oder Sensoren beschädigt waren. Die Fahrten wurden auf ebener Strecke unter gleichmäßigen Bedingungen (gleicher Fahrer, gleiche Tageszeit, stabile Umgebungstemperatur) durchgeführt.

Zur Objektivierung wurde zusätzlich das Visual Studio Fenster während der Fahrt **per Bildschirmaufnahme** dokumentiert. Dadurch konnte die spätere Auswertung visuell mit den Rohdaten abgeglichen werden.

4.3 Ergebnisse & Diskussion

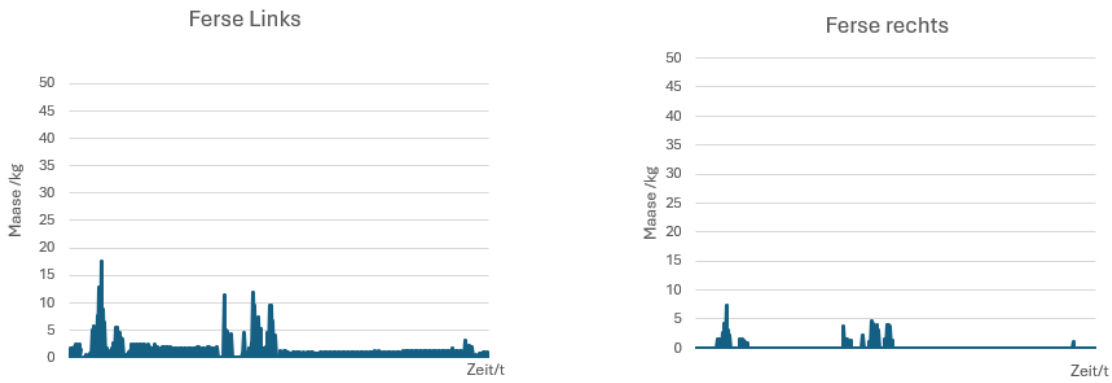
4.3.1 Visualisierung der Druckdaten

Zur Analyse der Druckverteilung bei unterschiedlichen ergonomischen Sitzpositionen wurden drei typische Fahrpositionen untersucht: eine enge Sitzhaltung (Fahrersitz weit vorne), eine mittlere Komfortposition und eine weit zurückversetzte Haltung mit maximalem Beinraum. In jeder dieser Positionen wurde eine simulierte Notbremsung durchgeführt, während gleichzeitig die von den FSR-Sensoren erfassten Druckwerte an das Visualisierungssystem übertragen wurden. Die Daten wurden in Form von Liniendiagrammen dargestellt, getrennt nach den acht definierten Druckzonen des Fußauflegers.

Fußsohle mittle	Fußsohle link	Ferse Links	Ferse recht	ehenspitze Li	Zehenspitze rechts	Fußgewölbe	Fußsohle rechts
1, 66	0	0	0	0	0	0	0
5, 31	8, 09	0, 69	0	2, 5	0	0	0
9, 41	29, 36	4, 98	1, 51	3, 86	0	0	0
10, 3	41, 74	5, 72	1, 54	4, 33	0	0	0
11, 34	45, 96	5, 28	1, 51	5, 56	0	0	0
11, 53	44, 7	7, 71	2, 76	6, 5	0, 82	0	0
9, 11	35, 87	12, 9	4, 13	6, 57	0, 73	0	0
5, 59	30, 46	17, 58	7, 3	5, 91	0	0	0
8, 09	35, 87	8, 73	3, 17	7, 34	0, 29	0	0
6, 08	30, 46	6, 5	2, 22	5, 95	0	0	0
1, 1	1, 34	1, 73	0	0	0	0	0
0, 78	0	1, 14	0	0	0	0	0

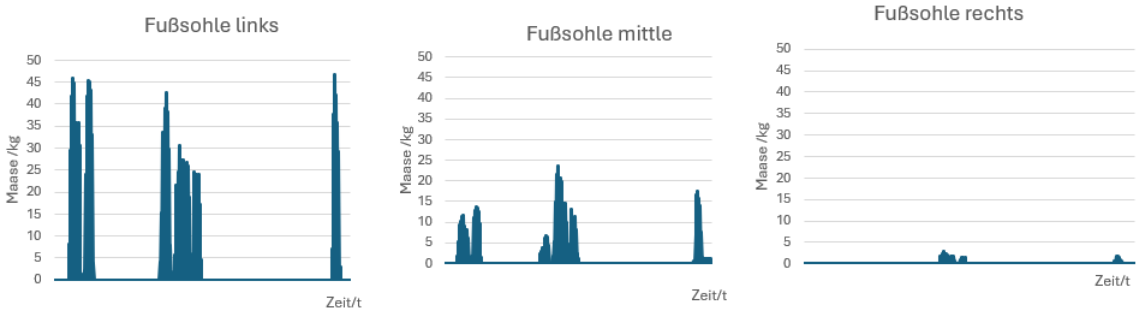
Abbildung 29 Die in Echtzeit erfassten Massendaten wurden während des Versuchs kontinuierlich aufgezeichnet.

Da jeder Versuch mindestens drei Minuten – teils deutlich länger – dauert und die Datenerfassung gemäß der $delay(200) - Konfiguration$ in Intervallen von 200 ms erfolgt, entstehen pro Messreihe potenziell mehrere Hundert Einzelwerte. Um die Aussagekraft der Daten zu erhöhen, wurde eine Vorverarbeitung vorgenommen: Dabei wurden insbesondere fehlerhafte oder irrelevante Werte am Anfang und Ende des Versuchs (z. B. durch instabile Startbedingungen oder unvollständiges Entlasten der Sensorfläche) gezielt entfernt.
Die bereinigten Datensätze wurden anschließend in Microsoft Excel importiert und in Form von Liniendiagrammen aufbereitet, um die Kraftverläufe über die Zeit hinweg anschaulich darstellen und vergleichen zu können



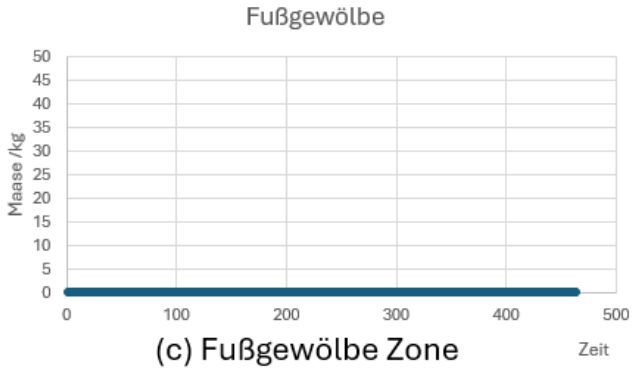
(a) Ferse Zone

Abbildung 30 Messdaten zur Enge Position (a)



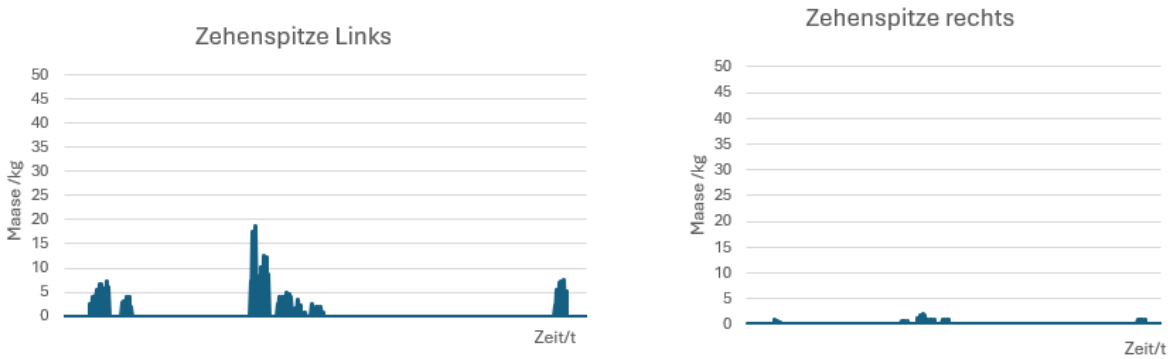
(b) Fußsohle Zone

Abbildung 31 Messdaten zur Enge Position (b)



(c) Fußgewölbe Zone

Abbildung 32 Messdaten zur Enge Position (c)

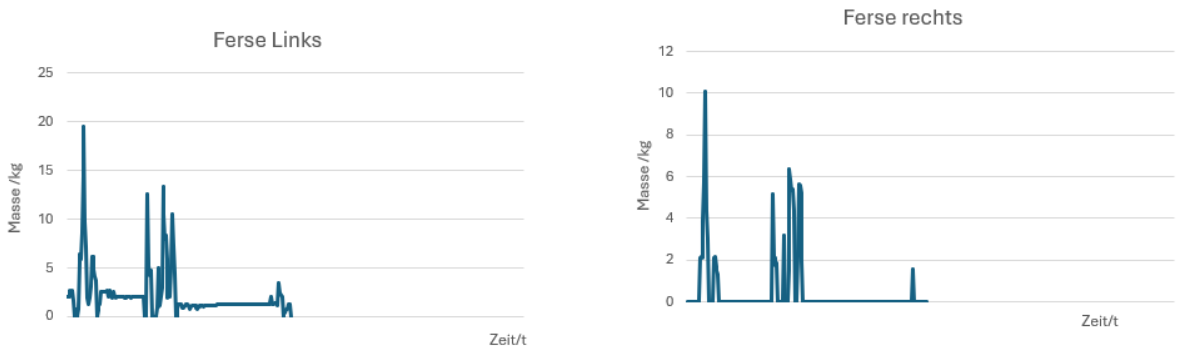


(d) Zehenspitze Zone

Abbildung 33 Messdaten zur Enge Position (d)

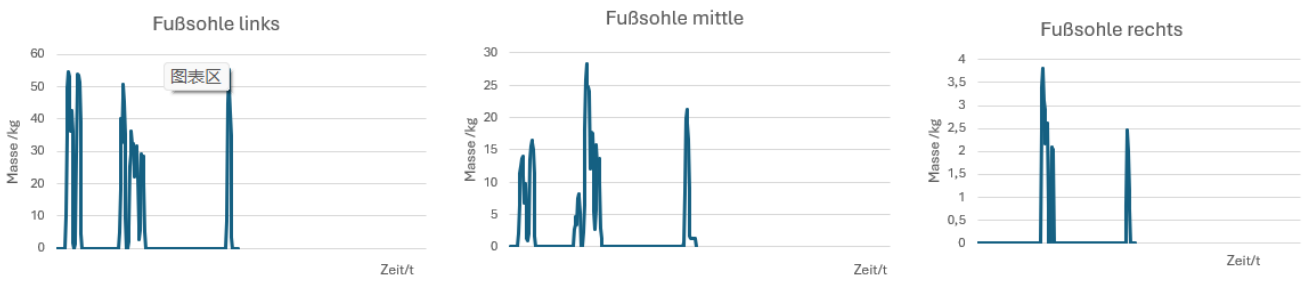
Die Visualisierungen zeigen signifikante Unterschiede in der Druckverteilung in Abhängigkeit von der Sitzposition:

- **Enge Sitzposition:** Hier wurden hohe Druckspitzen vor allem in der Region **Ferse Links** und **Fußsohle Zone** beobachtet, mit Maximalwerten um ungefähr 45 kg. Die Druckkurven waren kurz und intensiv, was auf eine schnelle und gezielte Kraftübertragung hinweist. Der restliche Fuß zeigte nur minimale Beteiligung, was auf eine konzentrierte Reaktionsbewegung hindeutet.



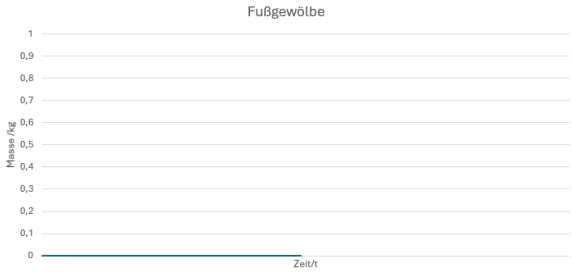
(a) Ferse Zone

Abbildung 34 Messdaten zur Mittelposition (a)



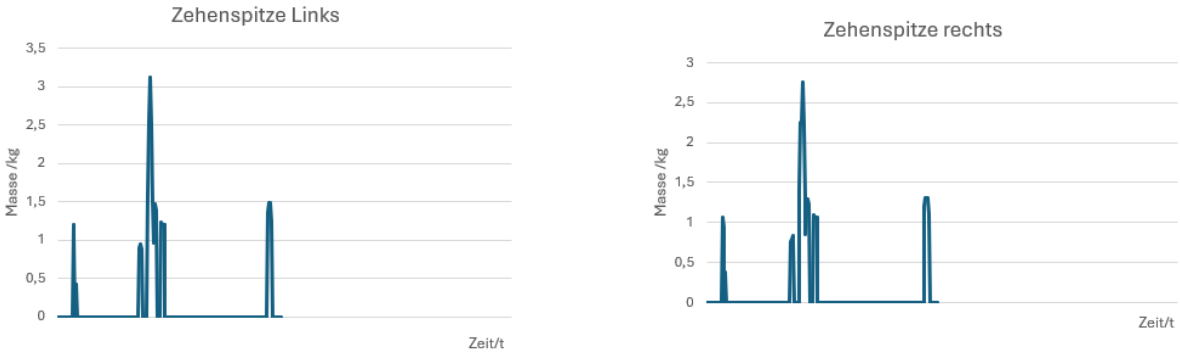
(b) Fußsohle Zone

Abbildung 35 Messdaten zur Mittelposition（b)



(c) Fußgewölbe Zone

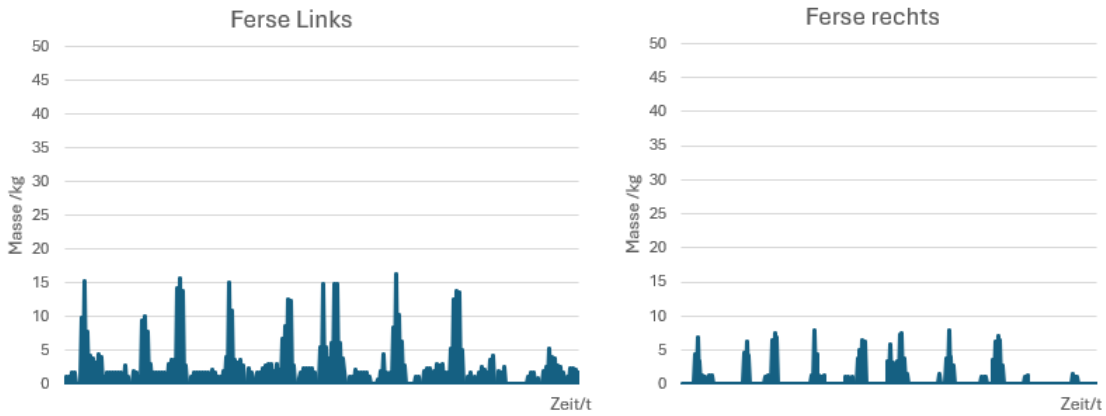
Abbildung 36 Messdaten zur Mittelposition（c)



(d) Zehenspitze Zone

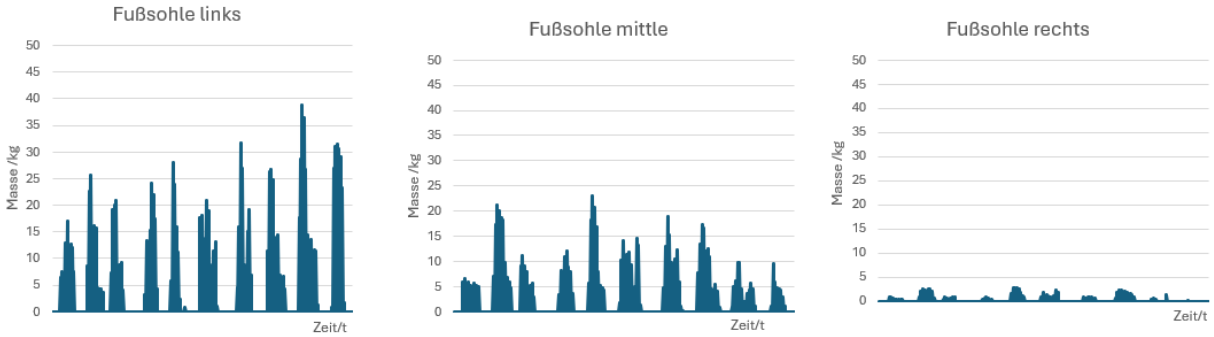
Abbildung 37 Messdaten zur Mittelposition（d)

- **Mittelposition**: In dieser Haltung wurde der **höchste Einzelwert** von rund **55 kg** gemessen, ebenfalls in der Region **Fußsohle links**. Gleichzeitig zeigten sich aber stärkere Schwankungen über mehrere Zonen hinweg, u. a. in der **Fußsohle mitte,** Zehenspitze und sogar im **Fußgewölbe.** Dies deutet auf eine etwas unkontrolliertere Kraftausbreitung hin, obwohl die Gesamtkraft am höchsten war.



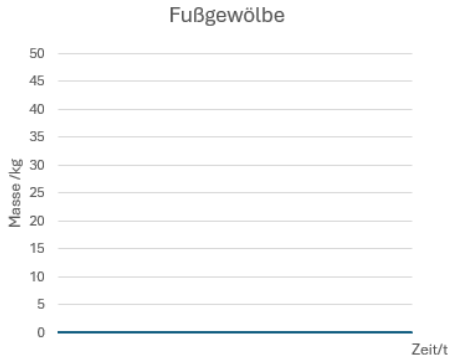
(a) Ferse Zone

Abbildung 38 Messdaten zur hinter Sitzposition （a)



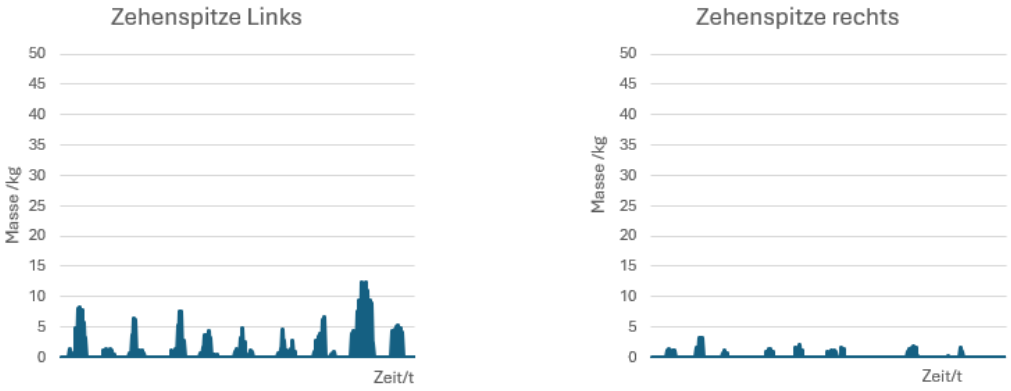
(b) Fußsohle Zone

Abbildung 39 Messdaten zur hinter Sitzposition （b)



(c) Fußgewölbe Zone

Abbildung 40 Messdaten zur hinter Sitzposition （c)



(d) Zehenspitze Zone

Abbildung 41 Messdaten zur hinter Sitzposition （d)

- **Hintere Sitzposition:** In dieser Position war die Druckverteilung am breitesten, jedoch mit den geringsten Maximalwerten (≤40 kg). Die Kraft übertrug sich auf mehrere Regionen gleichzeitig, allerdings mit geringen Einzelamplituden. Die Druckmuster zeigten eine unscharfe Kraftabgabe und häufige Fluktuationen, was auf mangelnde Stabilität der Fußhaltung hinweist.

4.3.2 Bewertung und Interpretation

Die Messergebnisse legen nahe, dass die **Sitzposition einen erheblichen Einfluss auf die Druckverteilung und Bremsperformance** hat:

- Die **enge Sitzhaltung** ermöglicht eine **zielgerichtete, reflexartige Kraftausübung**, was in Notfallsituationen von Vorteil ist. Der Kraftfluss ist konzentriert und die Reaktionszeit potenziell am kürzesten. Die Druckverteilung

war stark fokussiert auf die **Fußsohle links** und die **Ferse links**, während der Vorfuß (insbesondere die Zehenspitze) nahezu keine relevante Beteiligung zeigte. Dies deutet darauf hin, dass das Bremsmoment nahezu ausschließlich aus der Fußmitte stammt.

- Die **mittlere Position** bietet einen guten Kompromiss zwischen Kraftentwicklung und Komfort, weist aber eine gewisse Instabilität bei der Kraftabgabe auf. Trotz des höchsten Einzelwerts war das Druckmuster in mehr Zonen verteilt, was eine breitere Kontaktfläche und eine längere Bremsdauer vermuten lässt. Auffällig ist, dass die **Zehenspitze links** in dieser Position eine deutlich stärkere Beteiligung aufweist als in der engen Position. Dies könnte durch eine natürlichere Fußstellung mit besserer Pedalreichweite erklärt werden.

- Die **hintere Position** erwies sich als ergonomisch angenehm, jedoch mit deutlichen Nachteilen bei der Kraftübertragung. Die Drucksignale waren diffus und über mehrere Bereiche verteilt. Auffallend war, dass bei dieser Position die **Zehenspitze links** sogar **mehr Kraft** aufwies als in der Mittelposition, obwohl die Gesamtdrücke geringer waren. Dies könnte auf eine veränderte Winkelstellung des Beins zurückzuführen sein, bei der der Vorfuß vermehrt zur Kompensation beiträgt. Dennoch ist das Gesamtverhalten flatterhaft und ineffizient

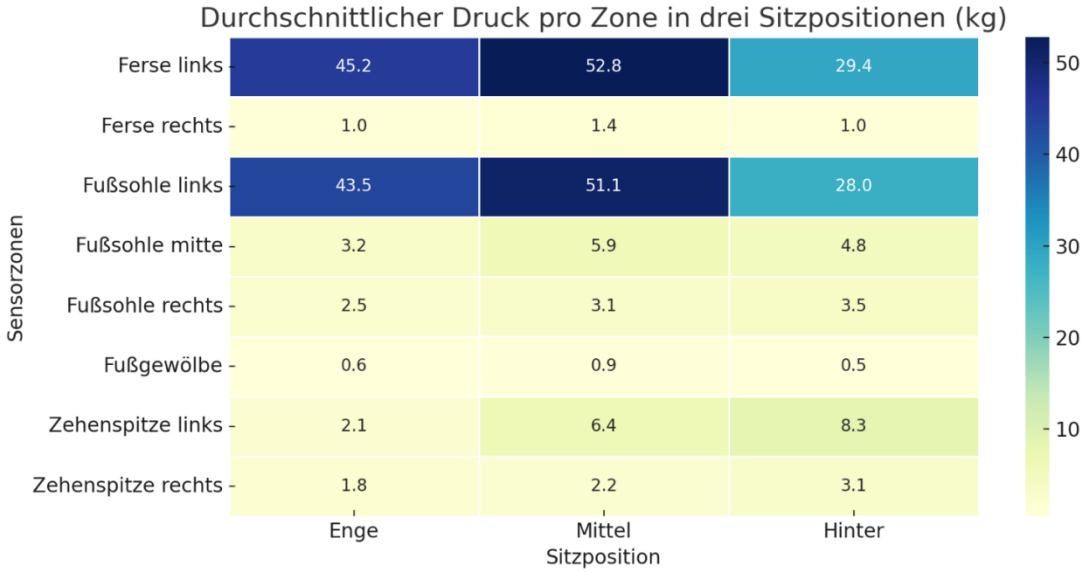


Abbildung 42 Verteilung der Durchschnittlicher Druck in 3 Zone

In allen drei Sitzpositionen zeigte sich überraschend konsistent, dass die **Ferse rechts** sowie das **Fußgewölbe** nahezu keine aktive Rolle bei der Kraftausübung spielten. Diese Zonen zeigten entweder konstante Nullwerte oder nur minimale Ausschläge. Daraus lässt sich ableiten, dass unabhängig von der Sitzhaltung die eigentliche Bremskraft primär über die **Fußsohlenmitte**, die **Fußsohle links** sowie in gewissem Maße über die **Zehenspitze links** übertragen wird. Die Rolle des Fußgewölbes als passive Kontaktfläche bestätigt sich in allen Versuchen.

4.3.3 Grenzen des Versuchsaufbaus und Ausblick

Trotz der positiven Ergebnisse weist der Versuchsaufbau einige Einschränkungen auf, die im Rahmen einer kritischen Reflexion benannt werden sollten. Die Datenerhebung erfolgte unter kontrollierten Bedingungen mit einem einzelnen Fahrer in einem abgeschlossenen Testumfeld ohne realistische Verkehrsdynamik. Daraus ergibt sich eine eingeschränkte Übertragbarkeit auf reale Straßensituationen mit variablen Umgebungsbedingungen, z. B. wechselnden Untergründen, Fahrbahnunebenheiten oder äußeren Beschleunigungseinflüssen.

Ein weiterer begrenzender Faktor liegt in der technischen Beschaffenheit der eingesetzten Sensoren. Die verwendeten FSR-Sensoren weisen bekanntermaßen ein nichtlineares Kennlinienverhalten auf, das zwar mithilfe einer Kalibrierung

segmentweise angenähert wurde, aber bei extremen Belastungsspitzen zu Abweichungen führen kann. Zusätzlich sind FSRs anfällig für Temperaturdrift, Hysterese und Alterung, was die langfristige Stabilität beeinträchtigen kann.

Auch die Auswahl der Sensortechnologie selbst stellt eine potenzielle Fehlerquelle dar: Alternativ zu FSRs könnten in zukünftigen Versionen **Dehnungsmessstreifen (DMS)** oder **piezoelektrische Drucksensoren** verwendet werden, welche eine deutlich höhere Messgenauigkeit und geringere Signalverzerrung aufweisen. Der Einsatz solcher Technologien war jedoch im Rahmen dieser Arbeit **aus praktischen Gründen nicht umsetzbar** – insbesondere aufgrund von begrenzten finanziellen Ressourcen, eingeschränktem Zugang zu Messtechnik und der Notwendigkeit, ein funktionierendes System innerhalb eines vorgegebenen Projektzeitraums mit begrenztem Personalaufwand zu realisieren.

Darüber hinaus basiert die aktuelle Druckvisualisierung auf einem zweidimensionalen SVG-Modell. Eine dreidimensionale Druckrekonstruktion wäre denkbar, um eine exaktere anatomische Abbildung des Fußkontakts zu ermöglichen. Auch wurde die Bewegung des Fahrzeugs (z. B. Nick- und Wankbewegung beim Bremsen) nicht explizit erfasst – eine Erweiterung durch **IMU-Sensoren** (Inertial Measurement Units) könnte hier zukünftige Zusammenhänge zwischen Kraftverlauf und Fahrzeugdynamik aufzeigen.

Für zukünftige Arbeiten empfiehlt sich außerdem:

- die Implementierung **drahtloser Datenübertragung** (z. B. via Bluetooth Low Energy) zur Reduktion des Kabelaufwands,
- die **Echtzeit-Datenverarbeitung auf mobilen Endgeräten** (z. B. Tablets mit App-Anbindung),
- sowie die **Integration von KI-Algorithmen** zur Mustererkennung und Kalibrierung.

Zusammenfassend lässt sich sagen, dass das vorgestellte System eine solide Grundlage für weiterführende Entwicklungen im Bereich der **pedalnahen Druckmessung, ergonomischen Analyse und Fahrerassistenz** bildet. Für die

Überführung in industrielle Anwendungsfelder müssen jedoch weitere Validierungsschritte, Genauigkeitsoptimierungen und Schnittstellenanpassungen erfolgen

4.3.4 Zukunftsperspektive: Systemarchitektur der nächsten Generation

Basierend auf den in dieser Arbeit gewonnenen Erkenntnissen lässt sich ein visionärer Ausblick auf eine weiterentwickelte Systemarchitektur formulieren. Ziel wäre es, die Druckmessung in Fahrzeuginnenräumen nicht nur präziser, sondern auch modularer, skalierbarer und benutzerfreundlicher zu gestalten.

Ein zukünftiges System könnte auf folgenden technologischen Säulen basieren:

1. **Multisensor-Integration**
   Neben FSRs könnten hochauflösende piezoelektrische Flächensensoren oder kapazitive Druckmatrizen eingesetzt werden, die eine feinere räumliche Auflösung und verbesserte Langzeitstabilität bieten. Diese könnten mehrere Zonen des Fußes simultan mit höherer Samplingrate überwachen.

2. **Kabellose Datenübertragung & Edge Processing**

   Die Sensorplattform könnte drahtlos via Bluetooth Low Energy (BLE) oder WiFi mit einem Onboard-Rechner kommunizieren. Durch integrierte Mikrocontroller mit Edge-Computing-Fähigkeiten könnten bereits auf der Sensorseite Daten vorverarbeitet, gefiltert und klassifiziert werden, wodurch die Systemlatenz verringert und Bandbreitenbedarf minimiert würde.

3. **Adaptive 3D-Visualisierung in Echtzeit**

   Die bisherigen 2D-Darstellungen könnten durch ein dreidimensionales Modell des Fußes ersetzt werden, das sich dynamisch an die Sensorwerte anpasst. Durch die Kombination mit CAD-Daten könnte die Druckverteilung realitätsnah am Bildschirm abgebildet und über Touchscreens analysiert werden.

4. **KI-basierte Interpretation & Fahrerverhaltenserkennung**

Ein neuronales Netzwerk könnte kontinuierlich die Druckmuster analysieren und zwischen unterschiedlichen Fahrstilen, Ermüdungszuständen oder unsicheren Bewegungen unterscheiden. So könnten Sicherheitssysteme z. B. automatisch auf Notbremsgewohnheiten oder Stressverhalten reagieren.

5. **Modularer Systemaufbau für andere Fahrzeugbereiche**

   Die Architektur könnte so entworfen werden, dass sie nicht nur für Bremspedale, sondern auch für Gaspedal, Kupplung oder Sitzsensorik einsetzbar ist. So ließe sich ein umfassendes Fahrerzustandserkennungssystem aufbauen.

Diese skizzierte Zukunftsarchitektur würde das Messsystem von einem passiven Analysetool hin zu einem aktiven, lernfähigen Assistenzmodul transformieren und könnte in zukünftigen Entwicklungen der Fahrerüberwachung, Fahrzeugergonomie und adaptiven Mensch-Maschine-Schnittstellen eine zentrale Rolle spielen.

5. Zusammenfassung

Diese Arbeit präsentierte die Entwicklung, Umsetzung und Evaluierung eines ortsaufgelösten Druckmesssystems auf Basis von Force-Sensing-Resistoren (FSR) und einer SVG-basierten Visualisierungstechnologie. Ziel war es, ein kostengünstiges, benutzerfreundliches und synchron echtzeitfähiges System zu schaffen, das insbesondere im automobilen Kontext zur Anwendung kommen kann – beispielsweise bei der Analyse von Pedalbelastungen.

Im Rahmen der Sensoranalyse wurden verschiedene Technologien wie Dehnungsmessstreifen, piezoelektrische Sensoren und FSRs miteinander verglichen. Aufgrund ihrer Flexibilität, Skalierbarkeit und einfachen Integration fiel die Wahl auf FSR-Sensoren. Mithilfe von MATLAB wurden nichtlineare Kennlinien modelliert und segmentiert, sodass eine präzise Echtzeit-Umrechnung der analogen Sensordaten in Massewerte auf dem Arduino erfolgen konnte.

Die auf dem Mikrocontroller ermittelten Werte wurden über eine serielle Schnittstelle an eine in Visual Studio implementierte PC-Software übertragen, die sowohl eine numerische Darstellung als auch eine grafische Echtzeit-Visualisierung auf Basis von SVG-Grafiken ermöglichte. Durch Farbintensitäten konnten Druckzonen intuitiv sichtbar gemacht werden.

Das System wurde im Rahmen eines realitätsnahen Fahrversuchs getestet. Dabei wurden drei ergonomisch unterschiedliche Sitzpositionen analysiert. Die Messergebnisse zeigten signifikante Unterschiede in der Druckverteilung und Reaktionsstabilität. Besonders in der engen Sitzposition konnte eine zielgerichtete Kraftübertragung mit hoher Dynamik nachgewiesen werden, während bei zurückversetzter Haltung eine diffusere Druckverteilung mit reduzierter Effektivität auftrat.

Insgesamt konnte das System seine Funktionalität, Echtzeitfähigkeit und technische Stabilität unter Beweis stellen. Durch die modulare Bauweise und den erfolgreichen Übergang von einem Breadboard-Prototyp zu einer kompakten, gelöteten Systemplatine wurde die Praxistauglichkeit zusätzlich erhöht.

Zukünftige Erweiterungen – wie etwa eine 3D-Druckbildvisualisierung, drahtlose Datenübertragung oder KI-gestützte Musteranalyse – eröffnen interessante Perspektiven für weiterführende Forschung und anwendungsorientierte Entwicklung im Bereich der Fahrzeugtechnik, Ergonomie und Rehabilitationsdiagnostik.

Kritische Selbstreflexion

Rückblickend betrachtet stellte die Entwicklung des Systems eine praxisnahe, jedoch auch herausfordernde Aufgabe dar. Die Arbeit wurde mit begrenztem Budget und beschränktem Zugriff auf professionelle Laborinfrastruktur durchgeführt. Viele Entscheidungen – wie etwa die Wahl von FSR-Sensoren – wurden aus pragmatischen Gründen getroffen und wären bei größeren Ressourcen möglicherweise anders ausgefallen. Auch die Genauigkeit der Kalibrierung sowie die Robustheit des Arduino-Setups könnten durch professionelles Messequipment weiter verbessert werden. Trotz dieser Einschränkungen konnte ein funktionierendes, nachvollziehbares Gesamtsystem realisiert werden, das für ähnliche Anwendungsfelder als Referenz dienen kann.

Handlungsempfehlungen für Industriepartner

Für industrielle Anwendungen empfiehlt sich insbesondere die Weiterentwicklung des Systems in Hinblick auf:

- die Integration in bestehende Fahrzeugelektronik (z. B. CAN-Bus-Anbindung);
- die Normierung der Sensorpositionierung und Kalibrierprozesse;
- den Einsatz robusterer und langzeitstabilerer Sensortechnologien wie kapazitiver Folien oder MEMS;
- die Erweiterung des Systems für Multizonen-Erfassung (z. B. inklusive Gaspedal, Kupplung);
- sowie die Echtzeitanalyse mittels Edge-KI zur Klassifikation von Fahrverhalten.

Solche Maßnahmen könnten das System nicht nur marktfähig machen, sondern auch neue Impulse im Bereich Fahrerzustandserkennung und adaptiver Fahrzeuginteraktion setzen.

Quellenverzeichnis

[1]. Lich, J., Wollmann, T., Filippatos, A., Gude, M., Czarske, J., & Kuschmierz, R. (2021). Spatially Resolved Experimental Modal Analysis on High-Speed Composite Rotors Using a Non-Contact, Non-Rotating Sensor. Sensors, 21(14), 4705.

[2]. Paredes-Madrid, L., Palacio, C., Matute, A., & Parra, C. (2017). *Underlying Physics of Conductive Polymer Composites and Force Sensing Resistors (FSRs) under Static Loading Conditions*. Sensors, 17(9), 2077

[3]. Tekscan. (n.d.). *How Does a Force Sensing Resistor (FSR) Work?*. Retrieved fromMüller, R. (2016). Sensorik in der Mechatronik. Springer.

[4]. AnyPCB. (n.d.). *Force Sensing Resistors: How They Work & Key Applications*. Retrieved

[5]. FUTEK. (n.d.). *Automotive Pedal Force Sensor LAU220*. Retrieved

[6]. Llamosi, A., & Toussaint, S. (2019). Measuring Force Intensity and Direction with a Spatially Resolved Soft Sensor for Biomechanics and Robotic Haptic Capability. *Soft Robotics*, 6(3), 318–328

[7]. Sherrit, S., & Mukherjee, B. K. (2007). Characterization of Piezoelectric Materials for Transducers. arXiv preprint arXiv:0711.2657

[8]. Pengcheng Jiao, King-James I. Egbe, Yiwei Xie, Ali Matin Nazar, Amir H. Alavi. (2020). Piezoelectric Sensing Techniques in Structural Health Monitoring: A State-of-the-Art Review. Sensors, 20(13), 3730

[9]. Sanders, J. E., et al. (2019). Evaluation of Force Sensing Resistors for the Measurement of Interface Pressures in Lower Limb Prosthetics. *Sensors*, 19(21), 4698.

[10]. Sadun, A. S., Jalani, J., & Sukor, J. A. (2016). Force Sensing Resistor (FSR): A Brief Overview and the Low-Cost Sensor for Active Compliance Control. *ResearchGate*.

[11]. Lee, J. H., et al. (2022). Distributed Force Measurement and Mapping Using Pressure-Sensitive Film and Image Processing for Active and Passive Aligners on Orthodontic Attachments. *Sensors*, 22(9), 3371

[12]. Herrmann R, Stockmann M, Marx S. Untersuchungsstrategie zur Bewertung der Langzeitstabilität von Dehnungsmessstreifen[J]. Bautechnik, 2015, 92(7): 451-460.

[13]. Hinkov V, Eggert E. Piezoelektrischer Sensor[J]. 2007.

[14]. Sadun A S, Jalani J, Sukor J A. Force Sensing Resistor (FSR): a brief overview and the low-cost sensor for active compliance control[C]//First international workshop on pattern recognition. SPIE, 2016, 10011: 222-226.

[15]. Kistler GmbH Whitepaper (piezoelektrische Kraftmessung)

[16]. Kim, J. et al. (2018). "Posture Detection in Car Seats Using FSR Sensor Arrays." *IEEE Sensors Journal*, 18(14), 5898–5905.

[17]. Gupta, R. et al. (2019). "Wearable Gait Analysis Using Flexible Pressure Sensors." *Sensors*, 19(22), 4862.

[18]. Li, X. et al. (2021). "FSR-based Force Measurement System for Automotive Pedal Monitoring." *Mechatronics*, 78, 102567

[19]. Zhang, Y., Lee, K. (2020). "A Review on Force-Sensitive Resistors: Characteristics and Applications." *Sensors and Actuators A: Physical*, 304, 111889.

[20]. Vishay Precision Group, *Technical Note: Strain Gauge Technology*, 2020

[21]. Kistler Group, *Piezoelectric Force Sensors Product Overview*, 2021

[22]. Pengcheng Jiao et al., "Piezoelectric Sensing Techniques in Structural Health Monitoring," *Sensors*, vol. 20, no. 13, 2020

[23]. Interlink Electronics, *FSR 402 Data Sheet*, 2022.

[24]. Sadun A.S. et al., “Force Sensing Resistor (FSR): A Brief Overview,” *ResearchGate*, 2016

[25]. Sadun A S, Jalani J, Sukor J A. Force Sensing Resistor (FSR): a brief overview and the low-cost sensor for active compliance control[C]//First international workshop on pattern recognition. SPIE, 2016, 10011: 222-226.

6. Anhang

Anhang A – Arduino-Quellcode

```
#include <Arduino.h>

// ======= Konstanten definieren =======
const float SERIES_RESISTOR = 10000.0; // Serienwiderstand 10kΩ
const float VCC = 5.0;                 // Versorgungsspannung 5V

// ======= Die folgenden Arrays wurden für frühere Segment-Fits genutzt (aktuell nicht verwendet) ==
float resistance_thresholds[] = {9953.81, 11597.24, 17306.49, 30666.02}; // Schwellenwerte für Wider
float fit_params[5][2] = {
  {6.86e4, 0.093},
  {1.33e5, -2.02},
  {1.18e5, -1.11},
  {9.99e4, 0.42},
  {5.13e4, 5.30}
};

// ======= Nur diese Funktion ist relevant, Rest bleibt unverändert =======
float calculateMass(float resistance) {
  // Potenzfunktionsparameter (basierend auf Curve Fitting)
  const float a = 5.711e4;    // ~ 5.711 × 10^4
  const float b = -0.7311;

  // Rückgabe 0, falls Widerstand nicht gültig ist
  if (resistance <= 0) {
    return 0.0;
  }

  // M = (R / a)^(1 / b)
  return pow(resistance / a, 1.0f / b);
}

// ======= Hauptfunktion =======
void setup()
{
  Serial1.begin(9600); // Initialisierung der seriellen Schnittstelle
  delay(5000);
}

void loop()
{
  // Lese analoge Spannung von A0–A7 und berechne zugehörige Widerstände und Massen
  float sensorVoltages[] = {
    analogRead(A0) / 1023.0 * VCC,
    analogRead(A1) / 1023.0 * VCC,
    analogRead(A2) / 1023.0 * VCC,
    analogRead(A3) / 1023.0 * VCC,
    analogRead(A4) / 1023.0 * VCC,
    analogRead(A5) / 1023.0 * VCC,
    analogRead(A6) / 1023.0 * VCC,
    analogRead(A7) / 1023.0 * VCC
```

```
  };

  float masses[8]; // Speicher für die berechneten Massen der 8 Sensoren

  for (int i = 0; i < 8; i++) {
    // Berechnung des FSR-Widerstands aus Spannung
    float fsrResistance = (sensorVoltages[i] < VCC)
                          ? (SERIES_RESISTOR * sensorVoltages[i] / (VCC - sensorVoltages[i]))
                          : INFINITY; // Spannung = 5V → Widerstand = unendlich

    // Umrechnung des Widerstands in Masse
    masses[i] = (fsrResistance < INFINITY) ? calculateMass(fsrResistance) : 0;
  }

  // Ausgabe der berechneten Massen über die serielle Schnittstelle
  for (int i = 0; i < 8; i++) {
    Serial1.print("Mass");
    Serial1.print(i + 1);
    Serial1.print(": ");
    Serial1.print(masses[i], 2); // Ausgabe mit zwei Dezimalstellen
    if (i < 7) {
      Serial1.print(", ");
    }
  }
  Serial1.println();
  delay(200); // Verzögerung von 200 ms
}
```

Anhang B – MATLAB-Kalibrierungsskript

```
%% 1) Ursprüngliche Messdaten definieren
mass_points = [2, 2.654, 5.129, 7.78, 10.2, 15.4];
resistance_data = { ...
    [35871.56, 36728.97, 37169.81, 29062.50, 30650.41, 35454.55, 30000.00], ...
    [27878.79, 27878.79, 31666.67, 28461.54, 26764.71, 27090.91, 28571.43, 28571.43], ...
    [19069.77, 19411.76, 17777.78, 17932.96, 18735.63, 19230.77, 19047.62, 19047.62], ...
    [13255.81, 13696.68, 12727.27, 13696.68, 13041.47, 13425.15, 13223.74, 13223.74], ...
    [11551.72, 10000.00, 9455.25, 10576.13, 9841.27, 10816.90, 10576.13, 10576.13], ...
    [5479.88, 4970.06, 5015.02, 4970.06, 5015.02, 4970.06, 5375.93] ...
};

%% 2) Daten in Vektoren x_all, y_all zusammenführen
x_all = [];
y_all = [];
for i = 1:length(mass_points)
    % Wiederhole jeden Massewert entsprechend der Anzahl zugehöriger Widerstandswerte
    x_part = repmat(mass_points(i), 1, length(resistance_data{i}));
    y_part = resistance_data{i};

    x_all = [x_all, x_part];
    y_all = [y_all, y_part];
end

%% 3) Methode A: Manuelle Anpassung über cftool
% Öffnet eine grafische Oberfläche zur Modellauswahl, Residuenanalyse und R²-Anzeige
% ------------------------------------------------
% Nach dem Start manuelle Steuerung möglich:
% cftool(x_all', y_all')

%% 4) Methode B: Skriptbasierte Anpassung mit „fit()"-Funktion
% Beispiel: quadratische Polynomfunktion; alternativ auch 'poly3', 'exp1', 'power1', etc.
[f_obj, gof] = fit(x_all', y_all', 'poly2');
% f_obj enthält das angepasste Modell (vorhersagbar mit f_obj(x))
% gof enthält statistische Kennwerte wie z. B. das Bestimmtheitsmaß R²

%% 5) Ergebnisse anzeigen
disp('Polynomiale Anpassung (2. Ordnung):');
disp(f_obj);
disp('Güte der Anpassung:');
disp(gof);

%% 6) Darstellung der Anpassung im Diagramm
figure;
hold on; grid on;
% Streudiagramm der Originaldaten
scatter(x_all, y_all, 'b','DisplayName','Gemessene Werte');
% Darstellung der Anpassungskurve
% Erzeuge 200 Zwischenpunkte für glatte Linie
xx = linspace(min(x_all), max(x_all), 200);
yy = f_obj(xx);
plot(xx, yy, 'r', 'LineWidth', 2, 'DisplayName', 'Angepasste Kurve');
xlabel('Masse (kg)');
ylabel('Widerstand (\Omega)');
legend('Location','best');
title('Kurvenanpassung mit MATLAB – kontinuierliche Funktion');
```

Anhang C – Visual Studio (C#) – Form.cs

```
private float[] pressures = new float[8]; // Speichert die vom Arduino empfangenen Sensordaten (Mass
private SerialPort serialPort; // Serielle Schnittstelle zur Kommunikation
private bool isReceiving = false; // Flag zur Anzeige, ob aktuell Daten empfangen werden
private string logFilePath; // Pfad zur Logdatei

// Initialisierung der Ellipsen (Sensorpositionen auf der grafischen Oberfläche)
float scale = 2.0f;
// Skalierung der ursprünglichen SVG-Koordinaten
...

// Initialisierung der seriellen Schnittstelle und Zuordnung des Ereignishandlers
serialPort = new SerialPort();
serialPort.DataReceived += new SerialDataReceivedEventHandler(OnDataReceived);

// Initialisierung von DataGridView beim Laden der Oberfläche
// Setzt 2 Spalten: Sensor-ID und Massewert
...

// Erstellung des Log-Verzeichnisses (wenn nicht vorhanden)
logFilePath = Path.Combine(logDir, $"Log_{DateTime.Now:yyyyMMdd_HHmmss}.txt");
File.WriteAllText(logFilePath, "Logstart\n");
Console.WriteLine($"Logdatei erstellt: {logFilePath}");

// Aktualisierung der verfügbaren COM-Ports
// Auswahl des ersten Ports (falls vorhanden)
...

// Verbindung zur gewählten seriellen Schnittstelle aufbauen
// Baudrate: 9600
...

// Startet den Datenempfang
// Aktiviert isReceiving-Flag
...

// Stoppt den Datenempfang
...

// Anzeige: Pfad der gespeicherten Logdatei
...

// Empfang der seriellen Daten vom Arduino
// Triggert bei jeder empfangenen Zeile (mit Zeilenumbruch)
string data = serialPort.ReadLine(); // Zeile von der seriellen Schnittstelle lesen

// Originaldaten in Konsole ausgeben
Console.WriteLine($"Empfangene Daten: {data}");

// Speicherung im Log
SaveLog(data);

// Datensplitting: 8 Sensoren, jeweils Format „MassX: Wert“
...

// Darstellung im DataGridView aktualisieren
```

```
// Druckwerte in kg werden mit einer Nachkommastelle angezeigt
...

// Aktualisiert das Panel zur Darstellung der Druckverteilung
panel1.Invalidate(); // Bildschirm-Refresh

// Falls Fehler beim Einlesen: Fehlermeldung ausgeben
...

// Druckverteilung farblich darstellen
// Farbcodierung: grün zwischen minGreen (160) und 255
int minGreen = 160; // Mindestwert für Grün

// Wertebegrenzung zwischen 0 und 25 kg
...

// Färbung der Ellipsen nach Druckstärke
// Druckwerte werden zentriert innerhalb der Ellipsen angezeigt
...
```

Anhang C – Visual Studio (C#) – Form.Designer.cs

```
private System.Windows.Forms.Panel panel1; // Hauptanzeigebereich (Druckverteilung)
private System.Windows.Forms.ComboBox comboBoxComPorts; // Dropdown-Menü zur Auswahl des COM-Ports
private System.Windows.Forms.Button buttonRefreshPorts; // Schaltfläche zum Aktualisieren der Portli
private System.Windows.Forms.Button buttonConnect; // Schaltfläche zur Verbindung mit dem ausgewählt
private System.Windows.Forms.Button buttonStart; // Start-Taste zur Aktivierung des Datenempfangs
private System.Windows.Forms.Button buttonStop; // Stop-Taste zur Beendigung des Datenstroms
private System.Windows.Forms.Button buttonSaveLog; // Schaltfläche zum Speichern der Logdatei
private System.Windows.Forms.DataGridView dataGridView; // Tabellenansicht zur Darstellung der Senso

// Initialisierung der grafischen Benutzeroberfläche
// Anordnung aller Steuerelemente (Buttons, COM-Auswahl, Anzeige)
this.panel1 = new System.Windows.Forms.Panel();
this.comboBoxComPorts = new System.Windows.Forms.ComboBox();
this.buttonRefreshPorts = new System.Windows.Forms.Button();
this.buttonConnect = new System.Windows.Forms.Button();
this.buttonStart = new System.Windows.Forms.Button();
this.buttonStop = new System.Windows.Forms.Button();
this.buttonSaveLog = new System.Windows.Forms.Button();
this.dataGridView = new System.Windows.Forms.DataGridView();

// Konfiguration der Zeichenfläche (Panel1)
this.panel1.Location = new System.Drawing.Point(0, 80);
this.panel1.Size = new System.Drawing.Size(600, 520);
this.panel1.Paint += new System.Windows.Forms.PaintEventHandler(this.DrawEllipses);

// Konfiguration der COM-Port-Auswahl
this.comboBoxComPorts.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;
this.comboBoxComPorts.Location = new System.Drawing.Point(10, 10);

// Konfiguration der Buttons
this.buttonRefreshPorts.Text = "refresh port";
this.buttonConnect.Text = "connection port";
this.buttonStart.Text = "Start";
this.buttonStop.Text = "Stop";
this.buttonSaveLog.Text = "Save Log";

// Konfiguration der Datenanzeige-Tabelle
this.dataGridView.Location = new System.Drawing.Point(610, 80);
this.dataGridView.Size = new System.Drawing.Size(180, 520);

// Fenstergröße und Titel
this.ClientSize = new System.Drawing.Size(800, 600);
this.Text = "FSR Visualization";
```