TECHNISCHE HOCHSCHULE MITTELHESSEN

THM | CAMPUS GIESSEN | MNI
Mathematik, Naturwissenschaften und Informatik

Bachelorarbeit

Realisierung und Evaluation einer Touch-basierten Benutzerschnittstelle für Remote-Labor-Versuchseinheiten

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

vorgelegt dem

Fachbereich Mathematik, Naturwissenschaften und Informatik
der Technischen Hochschule Mittelhessen

Kevin Klauser

im August 2025

Referent: Prof. Dr. Martin Weigel
Korreferent: Ing. Jakob Czekansky

Eidesstattliche Erklärung

In Übereinstimmung mit den Empfehlungen der Deutschen Forschungsgemeinschaft (DFG)[1] und denen der Zeitschrift Theoretical Computer Science[2] erkläre ich (Kevin Klauser) hiermit den Einsatz von generativer KI.

Bei der Vorbereitung dieser Arbeit habe ich ChatGPT 4 und Perplexity AI verwendet, um ausschließlich die Lesbarkeit und Sprache zu verbessern. Nach der Verwendung von ChatGPT 4 und Perplexity AI habe ich den Inhalt überprüft und nach Bedarf bearbeitet und übernehme die volle Verantwortung für den Inhalt dieser Arbeit.

In dieser Arbeit wird aus Gründen der besseren Lesbarkeit das generische Maskulinum verwendet. Weibliche und anderweitige Geschlechteridentitäten werden dabei ausdrücklich mitgemeint, soweit es für die Aussage erforderlich ist.

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Gießen, 25. August 2025 Kevin Klauser

[1] DFG formuliert Richtlinien für den Umgang mit generativen Modellen für Text- und Bild: https://www.dfg.de/resource/blob/289676/89c03e7a7a8a024093602995974832f9/230921-statement-executive-committee-ki-ai-data.pdf

[2] Erklärung zur Verwendung von generativer KI in wissenschaftlichen Arbeiten: https://www.sciencedirect.com/journal/theoretical-computer-science/publish/guide-for-authors

Kurzfassung

Kurzfassung

Titel

Realisierung und Evaluation einer Touch-basierten Benutzerschnittstelle für Remote-Labor-Versuchseinheiten

Zusammenfassung

Gegenstand dieser Arbeit ist die Entwicklung und Evaluation eines Touch-basierten Human-Machine-Interface (HMI), das im Kontext des hybriden Remote-Labors MICRO zum Einsatz kommt. Ziel ist es, ein eigenständiges, intuitiv bedienbares Interface zu entwerfen, das den Zugriff auf Laborstationen erleichtert und zentrale Steuerungs- und Überwachungsfunktionen bereitstellt. Die Umsetzung basiert auf einem Raspberry Pi mit angeschlossenem Touch-Display und ermöglicht unter anderem die Darstellung von Statusinformationen, die Bearbeitung von Nutzer-Reports sowie die Steuerung einer RGB-Beleuchtung zur visuellen Rückmeldung am Laboraufbau. Technisch wurde das HMI als webbasierte Anwendung mit Vue.js im Kiosk-Modus realisiert. Der Betrieb erfolgt containerisiert mithilfe von Docker und wird durch eine automatisierte CI/CD-Pipeline unterstützt. Die Arbeit umfasst neben der Softwarearchitektur und Systemintegration auch eine detaillierte Anforderungsanalyse sowie eine Evaluation verschiedener Touch-Displays hinsichtlich Benutzerfreundlichkeit und technischer Eignung sowie die Evaluation der entwickelten HMI-Anwendung. Das entwickelte HMI trägt zur besseren Bedienbarkeit technischer Lehrsysteme bei und ist flexibel erweiterbar. Damit leistet es einen konkreten Beitrag zur Weiterentwicklung digital gestützter Laborsysteme im Hochschulkontext.

Abstract

Title

Design and Evaluation of a Touch-Based Interface for Remote Lab Stations

Summary

The subject of this thesis is the development and evaluation of a touch-based Human-Machine Interface (HMI) designed for use in the context of the hybrid remote laboratory MICRO. The objective is to design a standalone, intuitively operable interface that facilitates access to laboratory stations while providing central control and monitoring functions. The implementation is based on a Raspberry Pi with an attached touch display and enables, among other features, the visualization of status information, the processing of user reports, and the control of RGB lighting for visual feedback within the laboratory setup. From a technical perspective, the HMI was realized as a web-based application using Vue.js in kiosk mode. Operation is containerized with Docker and supported by an automated CI/CD pipeline. In addition to the software architecture and system integration, the thesis includes a detailed requirements analysis as well as an evaluation of different touch displays with respect to usability and technical suitability, alongside an evaluation of the developed HMI application. The resulting HMI contributes to improved usability of technical teaching systems and offers flexible expandability. In this way, it makes a concrete contribution to the further development of digitally supported laboratory systems in the higher education context.

Inhaltsverzeichnis

Danksagung

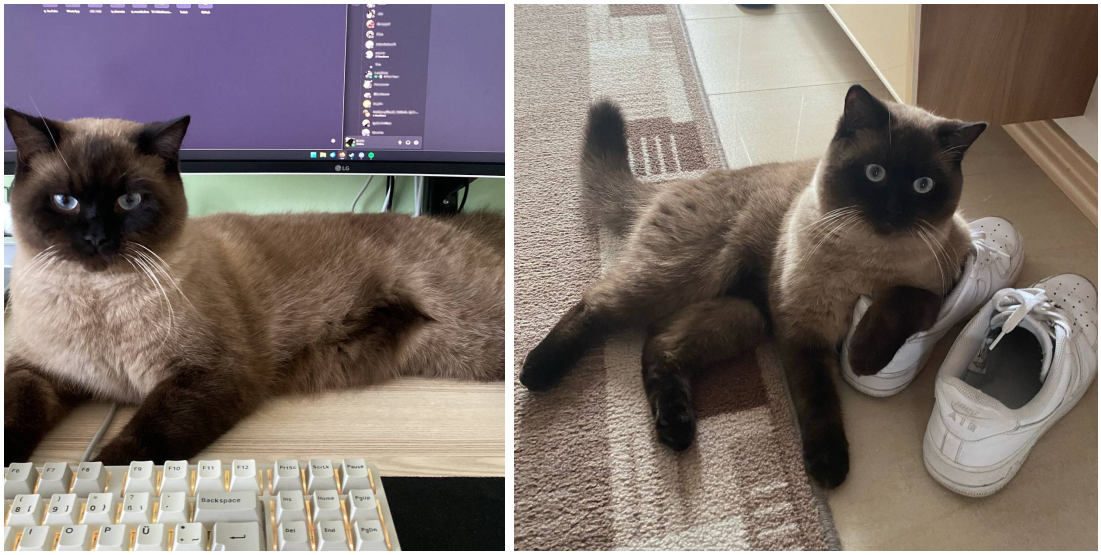


Abbildung 1: Mein Kater Milo

Ich möchte mich bei allen bedanken, die mich während der Entstehung dieser Arbeit unterstützt haben.

Ein großes Dankeschön geht an André, Nicolai, Justin, Ronny, Noah, Alex und Selma, die sich die Zeit genommen haben, meine Arbeit zu lesen und mir hilfreiches Feedback zu geben.

Für die Hilfe bei den Bildern und beim 3D-Druck möchte ich mich besonders bei Lia, Lukas und André bedanken.

Ein spezieller Dank geht an meinen Kater Milo, der mich bei unzähligen Schreibsessions begleitet und für gute Laune gesorgt hat.

Und natürlich auch danke an meine Familie und an Selma, die mich in dieser Zeit immer unterstützt und motiviert haben.

Nicht zuletzt danke ich allen Teilnehmerinnen und Teilnehmern der Evaluation. Ohne deren Rückmeldungen wäre diese Arbeit nicht in dieser Form möglich gewesen.

1 Einleitung

Mit fortschreitender Digitalisierung in der Hochschullehre und Forschung gewinnen sogenannte Remote-Labore zunehmend an Bedeutung [113]. Dabei handelt es sich um physische Laborumgebungen, auf die Nutzer über das Internet zugreifen können, um Experimente aus der Ferne durchzuführen [55]. Die Vorteile liegen in der orts- und zeitunabhängigen Nutzung, der besseren Auslastung teurer Geräte und der Möglichkeit, einer größeren Zahl von Teilnehmenden den Zugang zu praxisnahen Versuchen zu ermöglichen [4, 55].

Gleichzeitig steigt mit dem Wachsen solcher Labore der Bedarf an zuverlässigen Werkzeugen für die Verwaltung und Überwachung der zugrunde liegenden technischen Infrastruktur. Gerade in komplexen Laborszenarien mit mehreren Versuchsaufbauten und diversen Hardwarekomponenten sind benutzerfreundliche Systeme zur Steuerung, Überwachung und Konfiguration unerlässlich. Das im Rahmen dieser Arbeit entwickelte Human-Machine-Interface (HMI) richtet sich daher explizit an Administratoren eines Remote-Labors. Es unterstützt diese bei zentralen Aufgaben wie Statusüberwachung, Wartungsarbeiten, Zugriffssteuerung und Konfigurationsänderungen.

1.1 Motivation

Die Corona-Pandemie sowie die Energiekrise in den Jahren 2022 und 2023 hatten erhebliche Auswirkungen auf den Lehrbetrieb an Hochschulen [113]. Insbesondere praxisorientierte Fächer, wie die Arbeit mit eingebetteten Systemen, litten unter eingeschränkten Raumkapazitäten und strengen Kontaktbeschränkungen. Studierende waren teilweise nicht mehr in der Lage, physisch an Übungen mit Hardwarekomponenten teilzunehmen, obwohl gerade diese praktische Erfahrung essenziell für das Verständnis eingebetteter Systeme ist [26].

Als Antwort auf diese Herausforderungen entstand im Rahmen einer hochschulübergreifenden Initiative die Idee eines hybriden Remote-Labors namens MICRO, das Studierenden ortsunabhängig Zugriff auf reale Versuchsaufbauten ermöglichen soll. Die-

ses Konzept trägt dazu bei, die Widerstandsfähigkeit des Lehrbetriebs in Krisenzeiten zu erhöhen und eröffnet zudem neue didaktische Möglichkeiten für die technische Ausbildung. [12, 14, 26]

1.2 Problemstellung

Trotz der grundsätzlichen Zugänglichkeit des MICRO-Labors bestehen in der aktuellen Umsetzung wesentliche Einschränkungen, die den effizienten und einfachen Betrieb und die Wartung des Systems behindern:

Monitoring

- *Stationsübersicht*: Es existiert kein unkomplizierter Weg, vor Ort unmittelbar den Status der einzelnen Stationen einzusehen. Die Einsicht über die Stationen ist lediglich über die Website von MICRO möglich und benötigt dafür einen Computer. Eine Einsicht des Systems vor Ort ist somit nicht ohne Umwege möglich.

- *Einsicht auf Fehlermeldungen von MICRO-Nutzer über das System*: Das MICRO-System besitzt bereits für Nutzer die Möglichkeit, Fehler des Systems über die Webseite zu melden. Für die Auswertung der Fehlermeldungen gibt es bislang jedoch nur eine Ansichtsmaske in der Admin-Ansicht der Webseite. Somit ist es nicht einfach und komfortabel für MICRO-Administratoren möglich, Hardwarefehler einer Station direkt auslesen und beheben zu können.

Steuerung

- *Deaktivierung der Stationen*: Für Arbeiten an den Stationen ist stets eine manuelle Deaktivierung über die Website notwendig. Dieser Vorgang ist umständlich, da er mehrere Schritte und den Zugriff auf einen Computer voraussetzt.

- *Robotiksteuerung*: Es existiert aktuell kein Interface, um den im Labor vorgesehenen Portalroboter direkt zu steuern oder dessen Status einzusehen. Die Interaktion ist bislang nicht möglich.

- *RGB-LED-Steuerung*: Um die RGB-LEDs am MACRO-Schrank einzustellen, ist derzeit eine direkte SSH-Verbindung auf den steuernden Raspberry Pi notwendig. Anschließend muss ein spezifisches Skript manuell gestartet werden. Dieses Verfahren ist fehleranfällig, wenig komfortabel und setzt tiefgreifende Systemkenntnisse voraus.

Diese Einschränkungen führen insgesamt zu einem fragmentierten System, das weder eine ganzheitliche Übersicht noch eine konsistente und intuitive Steuerung der Laborressourcen ermöglicht. Insbesondere für wechselnde Nutzergruppen und für die Erprobung neuer Szenarien oder Erweiterungen hemmt diese Situation die Bedienbarkeit und Weiterentwicklung des Gesamtsystems.

1.3 Ziele der Arbeit

Diese Arbeit beinhaltet die Konzeption, Entwicklung und die Evaluation eines Touch-basierten HMIs, das die Überwachung und Bedienung der Stationen des MICRO-Labors erheblich vereinfachen und zentralisieren soll. Das HMI ermöglicht die:

- Abfrage von Statusdaten der Stationen und des Gesamtsystems.
- Interaktion mit den Stationen des Systems.
- Entwicklung einer Steuerung für den Portalroboter.
- intuitive Steuerung der RGB-Beleuchtung zur visuellen Rückmeldung direkt über eine grafische Oberfläche am Laboraufbau.
- Einsicht über MICRO-Nutzer-Fehlermeldungen einzelner Stationen und des Gesamtsystems.

Ziel ist ein benutzerfreundliches, zuverlässiges und erweiterbares Interface, das langfristig in den Laborbetrieb integriert werden kann. Die Evaluation schließt sich an die Entwicklungsphase an und umfasst sowohl die Überprüfung der HMI-Software im Hinblick auf die Anforderungen, als auch die Analyse der Benutzerfreundlichkeit und Bedienbarkeit mit unterschiedlichen Touch-Displays.

1.3.1 Relevanz und Einordnung der Arbeit

Vor dem Hintergrund der fortschreitenden Digitalisierung in der Hochschullehre und der zunehmenden Komplexität technischer Laborinfrastrukturen gewinnt die sichere und effiziente Verwaltung entsprechender Systeme immer mehr an Bedeutung. Gerade moderne Remote-Labore, wie das MICRO-Labor, vereinen zahlreiche Versuchsaufbauten, Hardwarekomponenten und verteilte Dienste, deren reibungsloser Betrieb und Wartung entscheidend von zentralen, benutzerfreundlichen Administrationswerkzeugen abhängen.

Das im Rahmen dieser Arbeit entwickelte HMI adressiert exakt diese Anforderungen. Im Gegensatz zu traditionellen Nutzeroberflächen, die meist auf die Bedienung durch Studierende ausgelegt sind, fokussiert sich dieses HMI explizit auf die Bedürfnisse und Aufgaben von Administratoren und Laborverantwortlichen.

Das HMI unterstützt Laborverantwortliche darin, Störungen frühzeitig zu erkennen und gezielt zu beheben, Konfigurationsänderungen ohne Umwege vorzunehmen und neue Komponenten flexibel in das System zu integrieren.

Die Übertragbarkeit des Ansatzes ist ein weiterer relevanter Aspekt: Der Einsatz von kostengünstiger, energieeffizienter Hardware (z. B. Raspberry Pi) und modernen Frameworks wie Vue.js [129] erlaubt eine unkomplizierte, skalierbare Anpassung für verschiedenste Labore und organisatorische Kontexte im Hochschulumfeld und darüber hinaus.

Zusammenfassend leistet diese Arbeit nicht nur einen Beitrag zur Modernisierung des konkreten Laborbetriebs am Beispiel des MICRO-Labors, sondern schafft generell eine Blaupause für administrative Schnittstellen in ähnlich komplexen technischen Infrastrukturen. Sie antwortet auf aktuelle Herausforderungen im Rahmen der Digitalisierung des Hochschulbetriebs und ist sowohl im Hinblick auf Effizienz und Sicherheit als auch auf Zukunftsfähigkeit relevant.

1.4 Struktur der Arbeit

Die vorliegende Arbeit gliedert sich in sieben Kapitel, die systematisch von den theoretischen Grundlagen bis hin zur Evaluation des entwickelten Systems führen. Nach dem Abstract und der Einleitung, in der Motivation, Problemstellung, Zielsetzung sowie die Einordnung der Arbeit erläutert werden, folgt in Kapitel 2 ein Überblick über

den Stand der Technik. Dabei wird ein Einblick in bestehende Remote-Labore gewährt, sowie auch der Projektkontext des Remote-Labors MICRO erläutert. Zudem werden sowohl bestehende HMI-Lösungen im industriellen und bildungsnahen Umfeld als auch relevante Evaluationsmethoden betrachtet.

Kapitel 3 behandelt die technologischen und theoretischen Grundlagen, welche für das Verständnis der Arbeit und des entwickelten HMI erforderlich sind.

In Kapitel 4 werden die konzeptionellen Überlegungen zur Entwicklung des HMIs vorgestellt. Es umfasst die Anforderungsanalyse, die Auswahl geeigneter Technologien sowie das Designkonzept für die Systemarchitektur und das Benutzerinterface.

Die praktische Umsetzung wird in Kapitel 5 beschrieben. Dazu zählen der Aufbau der Raspberry-Pi-basierten Umgebung, die Entwicklung der grafischen Oberfläche, die Integration ins Gesamtsystem sowie die Einrichtung der CI/CD-Pipeline.

Kapitel 6 beinhaltet die Evaluation des entwickelten Systems. Es werden die Methodik, die erhobenen Daten sowie die Ergebnisse der Untersuchung dargestellt und diskutiert. Dabei liegt ein besonderer Fokus auf der Bewertung unterschiedlicher Touch-Displays und der Usability der Software.

Abschließend fasst Kapitel 7 die wichtigsten Erkenntnisse zusammen und gibt einen Ausblick auf mögliche Weiterentwicklungen und zukünftige Anwendungsszenarien.

2 Stand der Technik

Dieses Kapitel stellt den aktuellen Stand der Technik sowie zentrale wissenschaftliche und praxisrelevante Arbeiten im Umfeld von Remote-Laboren und Human-Machine-Interface (HMI) vor. Ziel ist es, wesentliche Entwicklungen, etablierte Technologien und maßgebliche Trends zu beleuchten, die für die Gestaltung und Bewertung moderner benutzerzentrierter Steuer- und Verwaltungsschnittstellen im Remote-Labor-Kontext relevant sind. Die systematische Betrachtung bestehender Ansätze und Lösungen dient dazu, den in dieser Arbeit gewählten methodischen und technologischen Ansatz zu motivieren und einzuordnen. Zudem werden anerkannte Evaluationsmethoden für die Bewertung der Benutzerfreundlichkeit und User Experience (UX) aufgezeigt, um die wissenschaftliche Fundierung der abschließenden Evaluation im Rahmen dieser Arbeit zu gewährleisten.

2.1 Remote-Labore

Die Digitalisierung der Hochschullehre hat durch die COVID-19-Pandemie erheblich an Bedeutung gewonnen [12, 14]. In diesem Kontext haben sich Remote-Labore als eine vielversprechende Lösung erwiesen, um Studierenden auch außerhalb traditioneller Präsenzlehre einen Zugang zu realen Experimenten und Systemen zu ermöglichen. Besonders im Bereich der eingebetteten Systeme bieten Remote-Labore heute vielfältige Ansätze, um Lernenden hardware- und softwarenahe Kompetenzen zu vermitteln.

Remote-Labore sind in der Regel software- und hardwarebasierte Systeme, die den Zugriff auf reale Laborgeräte über das Internet ermöglichen [55]. Die zentrale Schnittstelle hierfür bildet eine Benutzerschnittstelle (engl. User Interface (UI)), welche in moderner Ausgestaltung Web- oder Cloud-basiert ist und Funktionen wie Gerätesteuerung, Code-Uploads, Datenvisualisierung sowie Gruppenarbeitsfeatures integriert [11, 45, 98, 99].

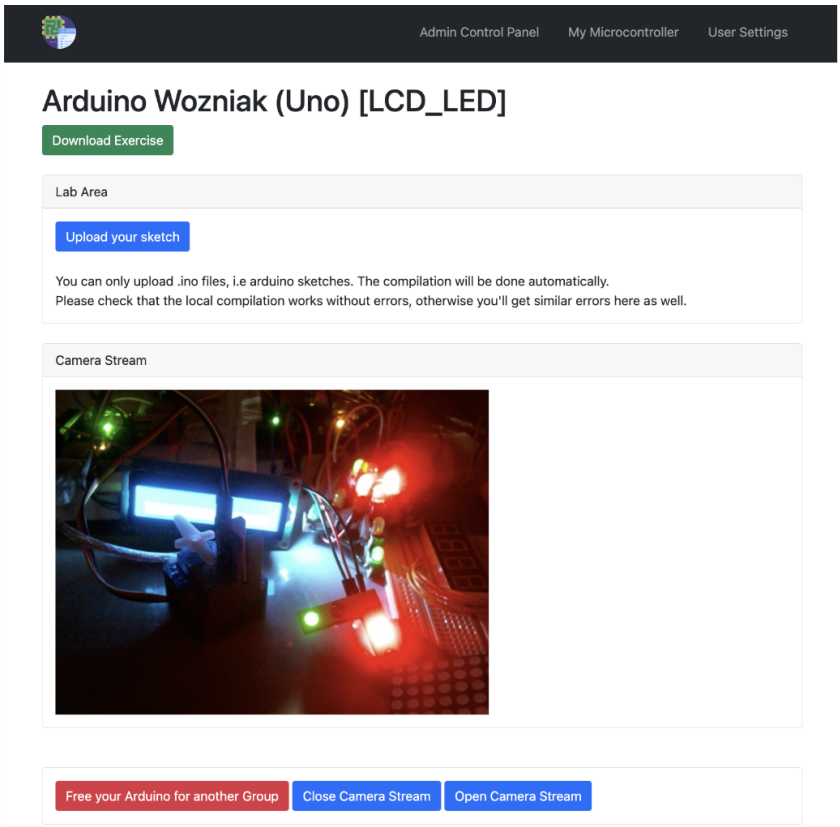


Abbildung 2.1: Webansicht eines Ampelsimulationsaufbaus für Admin-Nutzer des digital-labs [11]

2.1.1 digital-lab

Ein aktuelles Beispiel ist das Gruppen-basierte Remote-Labor digital-lab von Beck et al., bei dem der kollaborative Aspekt im Vordergrund steht [11]. Die Plattform ist darauf ausgelegt, Gruppenarbeiten an Embedded-Projekten ortsunabhängig zu ermöglichen und gleichzeitig einen realitätsnahen Laborbetrieb zu simulieren. Eine Darstellung der Benutzeroberfläche von digital-lab ist in Abbildung 2.1 dargestellt. Das Remote-Labor setzt bei der Implementierung der Aufbauten auf den ADDIE-Ansatz (Analyze, Design, Develop, Implement, Evaluate) [106]. Der ADDIE-Ansatz gibt den Studenten eine Arbeitsweise vor, wie sie ein Problem angehen sollten. Dabei soll man zuerst das Problem analysieren, dann ein Konzept zur Lösung entwerfen, gefolgt von dessen Implementation [106]. Am Ende einer Iteration soll eine Evaluation aller Schritte erfolgen, um in einer weiteren Iteration Fehler und Probleme ausbessern zu können [106].

2.1.2 LabsLand

Plattformen wie LabsLand bieten eine Infrastruktur für den Austausch und die gemeinsame Nutzung von Remote-Laboren zwischen verschiedenen Bildungseinrichtungen [98]. Die Plattform fungiert als Vermittlungs- und Buchungssystem, über welches Nutzer auf reale Laboranlagen weltweit zugreifen können [98]. LabsLand setzt dabei auf externe Laboranlagen, welche ihre Aufbauten über LabsLand für Menschen weltweit zur Verfügung stellen [98]. Die Benutzerschnittstellen dieser Anbieter sind typischerweise webbasiert und oft in Lernmanagementsystemen (z. B. Moodle) integriert [98]. LabsLand ist dabei ein Spin-Off des WebLab-Deusto Projekts [98, 99].

2.1.3 WebLab-Deusto

WebLab-Deusto ist ein an der Universität Deusto entwickeltes Open-Source-Remote-Labor-Management-System, das seit den 2000er-Jahren international bei Hochschulen im Einsatz ist. Es ermöglicht Nutzern über das Internet den sicheren Zugang zu realen, physikalischen Experimenten in Bereichen wie Elektronik, Physik, Chemie oder Nukleartechnik. WebLab-Deusto unterstützt wesentliche Funktionen wie Authentifizierung, Zeitbuchung, Nutzerverwaltung und Gruppenmanagement. Die Plattform ist modular aufgebaut, lässt sich mit Lernmanagementsystemen wie Moodle koppeln und wurde so konzipiert, dass Labore zwischen verschiedenen Bildungsinstitutionen geteilt und gemeinsam genutzt werden können. [99]

2.1.4 VISIR

Eine der international am häufigsten eingesetzten Remote-Labor-Plattformen für die elektrische Messtechnik ist VISIR (Virtual Instrument Systems in Reality). VISIR bietet Nutzern die Möglichkeit, reale elektronische Messgeräte und Experimentieraufbauten über eine webbasierte, virtuelle Benutzeroberfläche ortsunabhängig zu bedienen und auszuwerten. Das System zeichnet sich durch eine originalgetreue Abbildung von Laborprozessen aus, indem es virtuelle Frontpanels bereitstellt, die den Bedienelementen realer Instrumente nachempfunden sind. Ein Beispiel dazu findet sich in Abbildung 2.2. Das Experimentieren erfolgt dabei in Echtzeit mit physikalischen Geräten, was sich vorteilhaft auf die Akzeptanz und die Lerneffektivität im Vergleich zu rein simulierten Umgebungen auswirkt. [45, 57]

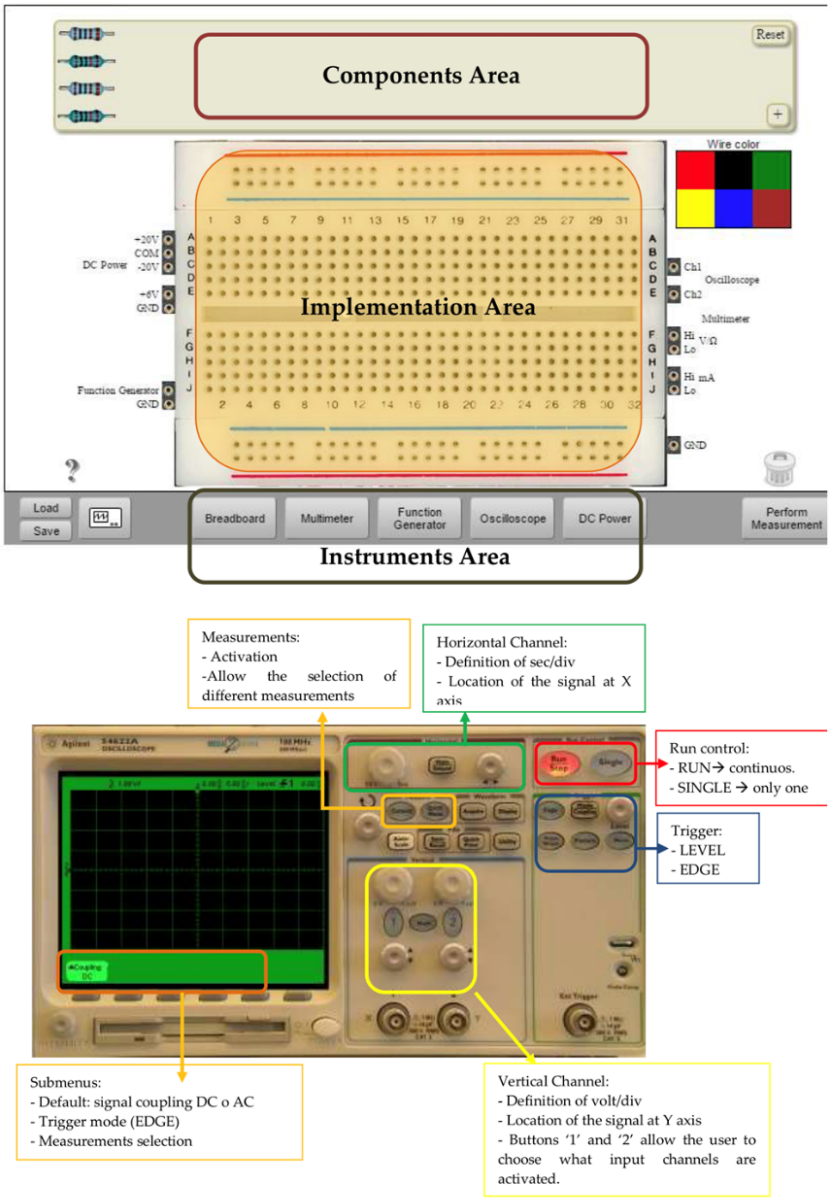


Abbildung 2.2: Web-Interface eines Breadboardaufbaus und eines Oszilloskops des VISIR Remote-Labors [84]

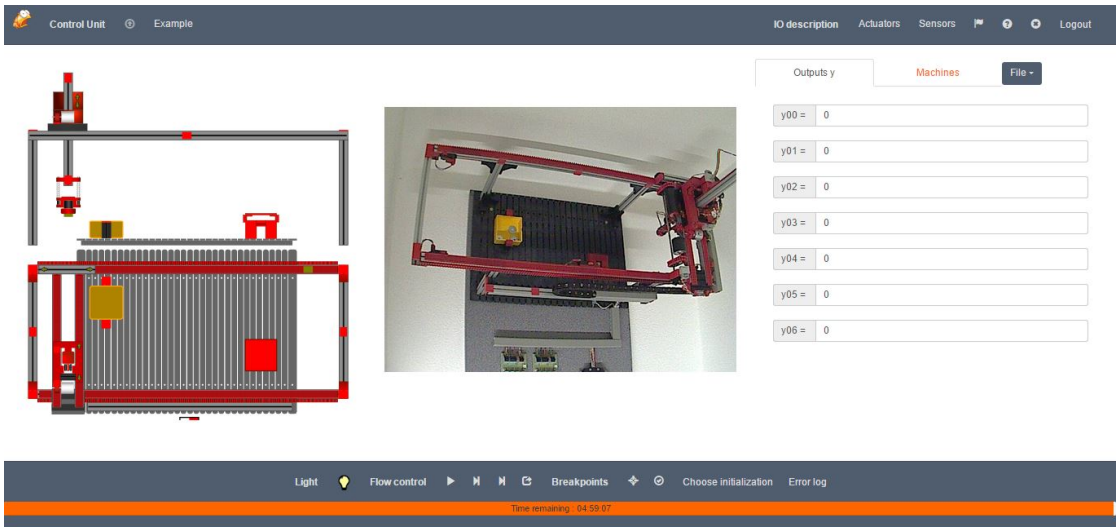


Abbildung 2.3: Ansicht eines Experimentes in GOLDi [121]

2.1.5 GOLDi

Ein weiteres Beispiel eines modernen Remote-Labors ist das GOLDi-System („Grid of Online Lab Devices Ilmenau“), das an der Technischen Universität Ilmenau entwickelt wurde [59]. GOLDi verfolgt einen grid-basierten Ansatz zur flexiblen und robusten Vernetzung verschiedener Laborgeräte und Steuerungseinheiten innerhalb eines gemeinsamen Online-Labornetzwerks [59]. Kernidee ist eine serielle Bus-Infrastruktur, über die beliebige Kombinationen aus Web-Steuerungseinheiten (etwa Mikrocontroller, FPGA, SPS) und physischen Hardwaremodellen für Experimente verschaltet werden können [59].

Für Anwender stellt GOLDi eine webbasierte Experiment-Oberfläche zur Verfügung, das vollständig als HTML5-Anwendung umgesetzt wurde. Hier können Nutzer eigene Steueralgorithmen hochladen, Versuche starten, stoppen oder zurücksetzen, Umgebungsparameter anpassen, Debugging-Funktionen nutzen und per Live-Webcam den Experimenten zuschauen. Die Weboberfläche eines Experimentes ist in Abbildung 2.3 dargestellt. [59]

Mit der Entwicklung der GOLDi Cloud wurde die bisher dezentrale Struktur mit mehreren Einzelinstanzen an verschiedenen Standorten in eine zentral verwaltete Cloud-Infrastruktur überführt [59]. Alle Instanzen sind über einen zentralen Zugang vernetzt, sodass Universitäten weltweit neue Funktionen synchron und sofort zur Verfügung stehen [59]. Nutzer können mit nur einem Login alle Partnerlabore nutzen, sich das Labor mit der schnellsten Verbindung auswählen und haben einen Überblick über alle

laufenden Experimente [59]. Datenschutzrelevante Informationen werden dabei ausschließlich zentral in Ilmenau verwaltet, der Laborzugriff selbst erfolgt jeweils direkt zwischen Client und lokalem Server [59].

GOLDi illustriert, wie durch modulare, grid-basierte und cloudfähige Remote-Labor-Infrastrukturen institutionsübergreifende, sichere und benutzerfreundliche Online-Laborumgebungen geschaffen werden können, die individuelle Anforderungen verschiedener Hochschulstandorte adaptiv abbilden und gleichzeitig zentrale Wartungsvorteile bieten.

2.1.6 CrossLab

Das CrossLab-Projekt verfolgt das Ziel, flexibel anpassbare digitale Laborumgebungen für die Hochschullehre bereitzustellen. Dabei ist CrossLab kein eigenes Remote-Labor, sondern vielmehr ein übergeordnetes Konzept, welches von anderen Laboren adaptiert werden kann. Im Mittelpunkt steht dabei die Integration verschiedener Labore (z. B. Remote-Labs, Simulationen oder VR/AR-Labore) in eine gemeinsame, interoperable Plattform [4]. CrossLab adressiert Herausforderungen wie eingeschränkten Zugang, hohe Kosten und starre Laborkonfigurationen, indem es eine Kombination von Labor-Komponenten anbietet, die sowohl fachübergreifend als auch hochschulübergreifend eingesetzt und geteilt werden können [4].

Das CrossLab-Projekt stellt kein monolithisches, eigenes Remote-Labor dar, sondern bietet Rahmenbedingungen, technische Standards und organisatorische Modelle, um bestehende und neue Laborinfrastrukturen flexibel, adaptiv und institutionenübergreifend einzubinden. CrossLab liefert damit Konzepte, Werkzeuge und Umsetzungsempfehlungen, mit denen einzelne Remote- oder Präsenzlabore zu einem vernetzten, adaptiven Gesamtsystem verbunden werden können. [4]

Beispielhaft werden im CrossLab-Projekt verschiedene bereits bestehende Laborlösungen (wie GOLDi und VISIR) eingebunden, weiterentwickelt und miteinander verknüpft. Die Partneruniversitäten behalten die Hoheit über ihre jeweiligen Labore, ermöglichen aber eine gegenseitige Nutzung der Systeme [4].

2.1.7 Zusammenfassung

Abschließend zeigt sich, dass Remote-Labore und ihre Benutzerschnittstellen einen essenziellen Beitrag zur Digitalisierung der Hochschullehre leisten. Die Bandbreite reicht

Tabelle 2.1: Vergleich ausgewählter Remote-Labore

Remote-Labor	**Erweiterbarkeit**	**Anwendungsgebiet**	**Schnittstellen**	**Verfügbarkeit**
digital-lab	Gering (noch nicht weit ausgebaut)	Embedded Systems	Web, Integration in Moodle möglich	Institutionsintern nutzbar
LabsLand	Hoch (Anbindung externer Labore, Vielfalt an Disziplinen)	Elektronik, Elektrotechnik, Physik, Informatik etc.	Web, Integration von Dritt-Laboren	Kommerziell
WebLab-Deusto	Sehr hoch (APIs, Einbindung beliebiger Fachlabore)	Vielzahl (Elektronik, Physik, Chemie, Informatik)	Web, Integration von Dritt-Laboren	Open Source
VISIR	Hoch (modular, verschiedene Geräte und Curricula)	Elektronische Messtechnik, Schaltungen	Web	Open Source
GOLDi	Hoch (gridbasiert, cloudfähig, institutionsübergreifend)	Automatisierungstechnik, Robotik, Steuerungstechnik	Web, Integration von Dritt-Laboren	Open Source, Partnernetzwerk weltweit
CrossLab	Hoch (Framework für Zusammenschluss vieler Labore und Simulationssysteme)	Disziplinübergreifend; Integration von Präsenz-, Remote-, Simulations-, VR/AR-Laboren	API-Frameworks	Konzeptuell
MICRO	Hoch (modular mit Docker/Services, Aufbauten erweiterbar)	Embedded Systems	Web	Nur für die THM verfügbar

von spezialisierten Plattformen wie digital-lab, die kollaboratives und problemorientiertes Lernen im Bereich der eingebetteten Systeme ermöglichen, über offene Infrastrukturen wie LabsLand und WebLab-Deusto, die einen institutionenübergreifenden Zugriff auf reale Laborressourcen bieten. Etablierte Systeme wie VISIR verknüpfen schließlich originalgetreu digitale mit realen Experimentierumgebungen. Ebenso stellt das GOLDi-System einen innovativen Ansatz dar, indem es durch seine grid-basierte und cloudfähige Architektur eine flexible und zentral wartbare Infrastruktur über mehrere Standorte hinweg ermöglicht und damit eine hohe Skalierbarkeit sowie eine effiziente Nutzung von Ressourcen realisiert. Insbesondere plattformübergreifende Konzepte wie das CrossLab-Projekt vereinen bestehende Laborlösungen, technische Standards und organisatorische Modelle zu einer adaptiven, teilbaren Laborlandschaft. Eine tabellarische Gegenüberstellung der Lösungen findet sich in Tabelle 2.1inklusive des MICRO Remote-Labors, auf welches in Abschnitt 2.2 genauer eingegangen wird. Gemeinsam schaffen diese unterschiedlichen Ansätze die Grundlage, Nutzern unabhängig von Ort und Institution einen nachhaltigen, kompetenzorientierten Zugang zu experimenteller Lehre zu ermöglichen und die universitäre Laborlandschaft zukunftsfähig weiterzuentwickeln.

2.2 Projektkontext: Remote-Labor MICRO

MICRO ist ein Remote-Labor, das an der Technischen Hochschule Mittelhessen (THM) entwickelt wurde. Ziel ist es, den Zugang zu hardwarebasierten Experimenten über das Internet zu ermöglichen. Studierende sollen damit Versuchsaufbauten aus der Ferne steuern, Daten auslesen und ihre Ergebnisse dokumentieren können. Das Labor basiert auf einem modularen Aufbau: Stationen sind über eine zentrale Plattform zugänglich und lassen sich durch digitale Schnittstellen ansteuern. Das Konzept wurde aus der Notwendigkeit heraus geboren, Labore auch bei eingeschränktem Präsenzbetrieb während der COVID-19 Pandemie nutzbar zu machen, und stellt heute eine zukunftsweisende Ergänzung klassischer Präsenzlehre dar. [26]

2.2.1 Systemüberblick

Das MICRO-System stellt durch eine verteilte Client-Server-Architektur reale Hardware-Experimente über eine browserbasierte Benutzeroberfläche bereit. Im Gegensatz zu rein simulationsbasierten Ansätzen arbeitet MICRO mit physischen Mikrocontroller-Setups, die über das Internet ferngesteuert werden können. Diese Herangehensweise

gewährleistet eine authentische Lernerfahrung, bei der Studierende mit realer Hardware interagieren und dabei praktische Kompetenzen in der Programmierung eingebetteter Systeme entwickeln. [24, 26]

Abbildung 2.4 zeigt einen Überblick über die essenziellen technischen Systeme von MICRO. Die Systemarchitektur umfasst mehrere Schlüsselkomponenten: ein zentraler Server innerhalb des universitären VPN-Netzwerks hostet die Homepage und das Front- und Backend von MICRO. Jede experimentelle Station operiert autonom mit einem eigenen Host-Computer, genannt Station-Controller, der verschiedene Dienste für Kamera-Streaming, Code-Kompilierung und Hardware-Steuerung bereitstellt. Die genauere Erläuterung der einzelnen Komponente des MICRO-Systems erfolgt in den folgenden Kapiteln.

2.2.2 MICRO

MICRO bezeichnet zum einen den Projektnamen, zum anderen jedoch auch die einzelnen hardwarebasierten Stationsaufbauten (MICRO-Cubes) sowie das Web-Interface (Vuefrontend), das für Steuerung, Monitoring und Interaktion mit den Versuchsstationen zuständig ist. [26]

MICRO-Cubes

Die MICRO-Cubes repräsentieren die grundlegenden experimentellen Einheiten des Systems. Jeder Cube ist als autonome, in sich geschlossene Laborstation konzipiert [24]. Die modulare Bauweise basiert auf industriellen Aluminium-Profilen und speziell angefertigten 3D-gedruckten Komponenten [26].

Jeder MICRO-Cube besteht aus drei Hauptkomponenten: einem Station-Controller (ein Host-Computer, typischerweise ein Raspberry Pi), einer standardisierten Webcam für visuelles Feedback und einer austauschbaren Plattform mit dem jeweiligen Experimentaufbau („Assembly“ genannt) [26]. Der Station-Controller fungiert als lokaler Server und führt verschiedene Services aus, die für die Kommunikation, Steuerung und Überwachung der Experimente erforderlich sind. Dieser dient somit als Schnittstelle zwischen einer MICRO-Station und dem restlichen MICRO-System [26, 27].

Die stapelbare Bauweise der Cubes ermöglicht eine platzsparende Anordnung und erleichtert die Skalierung des Systems. Diese Designentscheidung trägt erheblich zur Kosteneffizienz und Wartbarkeit der gesamten Infrastruktur bei. [26]

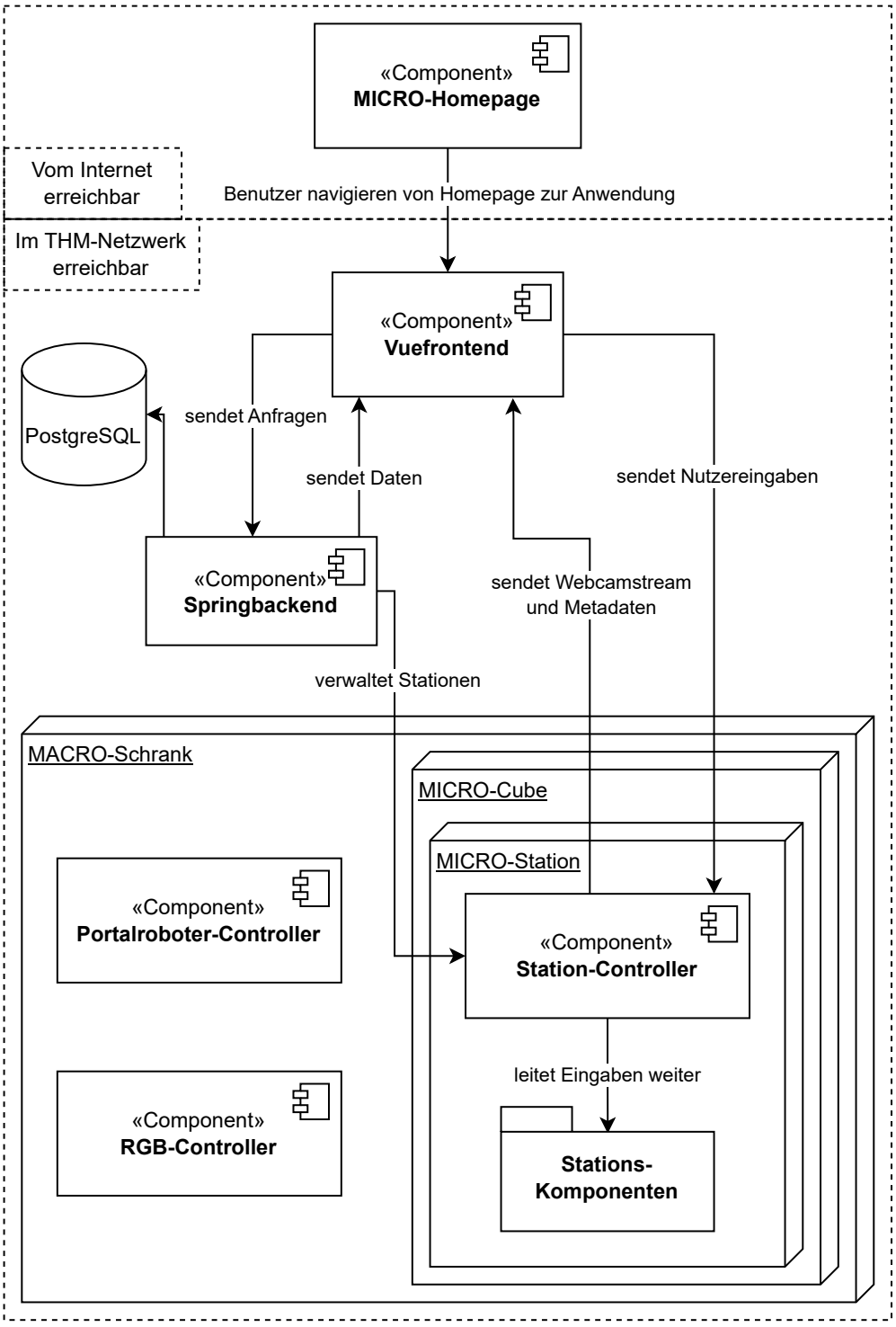


Abbildung 2.4: Essenzielle Systemkomponenten des Remote-Labors MICRO

Abbildung 2.5: Die Stationsansicht einer FPGA-Station mit Ampelsteuerungssimulation in MICRO

Stationstypen

Das MICRO-System unterstützt verschiedene Arten von Experimentierstationen, die jeweils auf spezifische Lehrmodule und Lernziele ausgerichtet sind [26].

Mikrocontroller-basierte Stationen bilden den ursprünglichen Schwerpunkt des Systems und werden primär im Modul „Mikroprozessortechnik“ eingesetzt [26]. Diese Stationen basieren auf einem AVR-Mikrocontroller und umfassen Peripherie-Komponenten wie LEDs, Taster, LC-Displays, Sensoren und Servomotoren [26]. Die Interaktion erfolgt über kamerabasiertes visuelles Feedback in Kombination mit Analog Discovery-Geräten von Digilent für die Signalgenerierung und -analyse [26]. Die Oberfläche für eine Mikrocontroller-basierte Station findet sich in Abbildung 2.10.

FPGA-basierte Stationen wurden für das Modul „Digitaltechnik“ entwickelt und nutzen Lattice basierte FPGAs (icoBoard). Im Gegensatz zu den mikrocontroller-basierten Stationen verwenden diese Einheiten FTDI-Breakout-Boards für die GPIO-Anbindung und ersetzen das kamerabasierte Feedback durch Echtzeit-Simulationen. Typische Experimente umfassen Wassertank-Finite-State-Machines, Siebensegmentanzeigen-Zähler

Abbildung 2.6: Der MACRO-Schrank des MICRO Remote-Labors

und Ampelsteuerungen. Die Simulationsoberfläche einer Ampelsteuerung findet sich in Abbildung 2.5. [24]

2.2.3 MACRO

Das MACRO-System stellt eine Erweiterung der MICRO-Infrastruktur dar und adressiert die dynamische Verwaltung und automatisierte Umkonfiguration von Experimentierstationen. Der Name MACRO steht außerdem für einen Schrank im Labor (siehe Abbildung 2.6), welcher die einzelnen MICRO-Stationen beherbergt. Darüber hinaus sind in dem MACRO-Schrank noch ein Portalroboter und RGB-LEDs verbaut.

Der MACRO-Schrank kann bis zu 24 MICRO-Cubes aufnehmen, wobei einige Cubes als Speicher für verschiedene Experimentaufbauten und andere als aktive Stationen fungieren.

Portalroboter

Der Portalroboter soll gesammelte Nutzungsdaten zur Vorhersage des Bedarfs an spezifischen Experimentaufbauten nutzen und diese proaktiv austauschen, um Wartezeiten zu minimieren [26]. Das System soll Experimentaufbauten wie Bücher aus einem

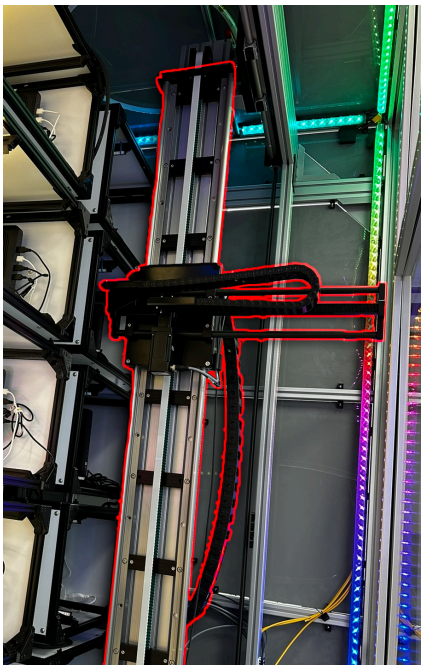


Abbildung 2.7: Portalroboter im MACRO-Schrank

Abbildung 2.8: Rendering des MACRO-Schrankes samt Portalroboter

Regal entnehmen und in aktive Stationen einsetzen, wodurch eine erhebliche Platzersparnis und Effizienzsteigerung erreicht wird. Der Portalroboter besitzt bislang noch kein Steuergerät und genau dazu soll auch das in dieser Arbeit entwickelte HMI dienen. Der Portalroboter ist in Abbildung 2.7 rot umrandet, und als Rendering innerhalb des MACRO-Schrankes in Abbildung 2.8 zu sehen.

MACRO RGB-LEDs

Das RGB-LED-System ist als visueller Indikator für den Status des Portalroboters vorgesehen und wurde im Rahmen dieser Arbeit um eine REST-Schnittstelle ergänzt. Die Steuerung erfolgt über einen dedizierten RGB-Controller, der MICRO-Administratoren eine Anzeige der verschiedenen Systemzustände ermöglicht. In Abbildung 2.6 und Abbildung 2.7 ist das System als regenbogenfarbenes Licht in der Umrandung des MACRO-Schranks dargestellt.

2.2.4 Frontend

Das Frontend (*Vuefrontend* in Abbildung 2.4) des MICRO-Systems basiert auf Vue.js [129] in Verbindung mit dem Vuetify-Framework [128] für ein einfaches und konsistentes Design [26]. Diese Technologiekombination wurde aufgrund der Vertrautheit des studentischen Entwicklungsteams gewählt und ermöglicht eine responsive, benutzerfreundliche Web-Oberfläche [26].

Die Benutzeroberfläche integriert mehrere Kernfunktionalitäten: einen Kamera-Stream für visuelles Feedback, virtuelle Bedienelemente zur Emulation physischer Komponenten (Taster, Potentiometer), eine Code-Upload-Funktionalität und ein UART Kommunikationsinterface für die direkte Kommunikation mit den Mikrocontrollern [25, 26]. Auf diese Komponenten wird in Unterunterabschnitt 2.2.6 weiter eingegangen.

Zu den existenten Features der Stationsreservierung ist zudem eine Implementierung von Shared Sessions geplant, die es mehreren Nutzern ermöglichen soll, gleichzeitig an demselben Experiment zu arbeiten [25]. Diese Funktionalität soll kollaboratives Lernen, Pair Programming und gemeinsame Fehlersuche durch Session-Key-Sharing oder ein Beitrittsanfrage-System unterstützen [25].

2.2.5 Backend

Das Backend (*Springbackend* in Abbildung 2.4) wurde in Kotlin unter Verwendung des Spring Boot-Frameworks implementiert und stellt eine RESTful API zur Verfügung. Die Architektur folgt modernen Microservice-Prinzipien und gewährleistet eine saubere Trennung zwischen verschiedenen Systemkomponenten. [26]

Der Backend-Service ist für die Benutzerauthentifizierung über das universitäre LDAP-System, die Stationsverwaltung, das Reservierungssystem und vielen weiteren Features zuständig. Nach erfolgreicher Reservierung einer Station stellt das Backend die notwendigen Verbindungsdetails bereit und ermöglicht dem Frontend direkte Kommunikation zwischen Client und Station, wodurch die Serverlast reduziert wird. [26]

Die Datenbank-Integration erfolgt über PostgreSQL und speichert umfassende Informationen zu Benutzerprofilen, Stationsdetails, Reservierungsprotokollen, Kursinformationen und Aufgabenspezifikationen. [26]

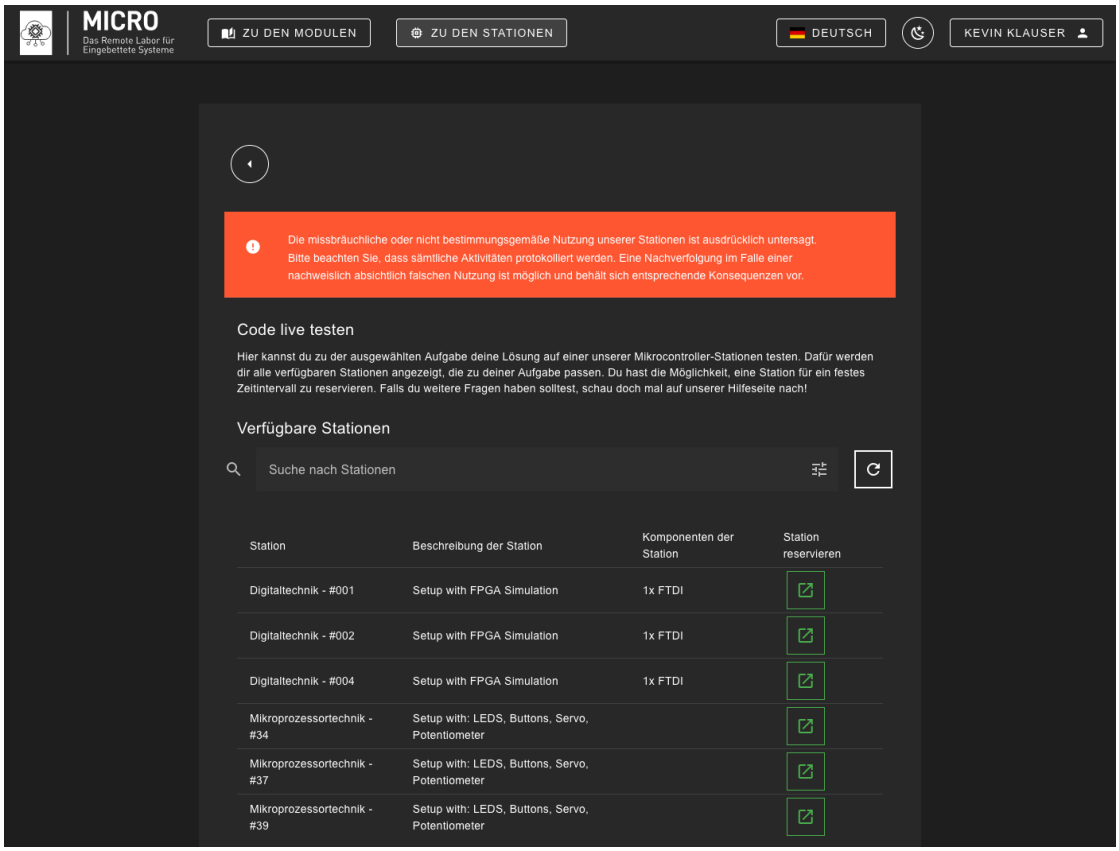


Abbildung 2.9: Liste der reservierbaren Stationen in MICRO

2.2.6 Nutzerinteraktion

Die Nutzerinteraktion im MICRO-System ist darauf ausgelegt, eine möglichst authentische Laborerfahrung zu bieten, während gleichzeitig die Vorteile des Remote-Zugriffs genutzt werden.

Stationsreservierung

Der Reservierungsprozess beginnt mit dem Zugriff auf die MICRO-Webanwendung (Vue-frontend) und ist nur über das universitäre VPN möglich. MICRO-Nutzer können verfügbare Stationen durchsuchen und reservieren (siehe Abbildung 2.9), wobei das zentrale Springbackend die Reservierungsverwaltung übernimmt. [26]

Nach erfolgreicher Reservierung der Station wird eine Direktverbindung zwischen dem MICRO-Nutzer und der zugewiesenen MICRO-Station etabliert. Alle weiteren Interaktionen werden über den lokalen Raspberry Pi der Station abgewickelt, wodurch nied-

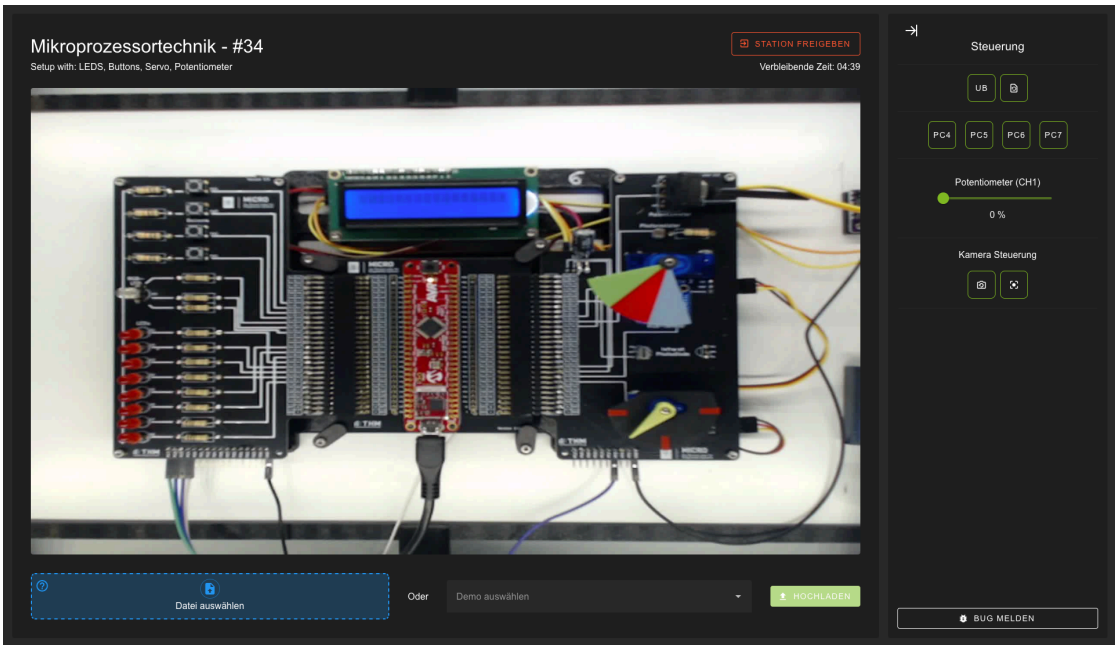


Abbildung 2.10: Die Stationsansicht einer Mikroprozessorstation in MICRO

rige Latenz und hohe Responsivität gewährleistet werden. Anschließend findet sich der MICRO-Nutzer auf der Stationsreservierungs-Ansicht wieder, dargestellt in Abbildung 2.10. Hier kann der MICRO-Nutzer den Webcam-Livestream der Station sehen, seinen Programmcode hochladen und mit der Station über die Interaktionsflächen auf der rechten Seite interagieren. Zudem gibt es die Möglichkeit per Universal Asynchronous Receiver / Transmitter (UART) über eine Konsolenschnittstelle mit der Station zu kommunizieren. In Abbildung 2.11 sieht man auf der rechten Seite die UART-Konsole und ein Modell des Stationsaufbaus. [26]

Am Ende der Reservierungszeit wird die Station automatisch zurückgesetzt und in einen standardisierten Ausgangszustand versetzt, um für den nächsten Nutzer bereit zu sein. Der Nutzer findet sich dann wieder in der Reservierungsliste, dargestellt in Abbildung 2.9. Dieses dezentrale Modell ermöglicht eine effiziente Skalierung und reduziert die Abhängigkeit von zentralisierten Services während der Experimentdurchführung. [26]

2.3 Übersicht bestehender HMI-Lösungen

Die HMI Technologie hat sich zu einem fundamentalen Baustein der modernen industriellen Automatisierung entwickelt. Diese Analyse untersucht die aktuellen HMI-

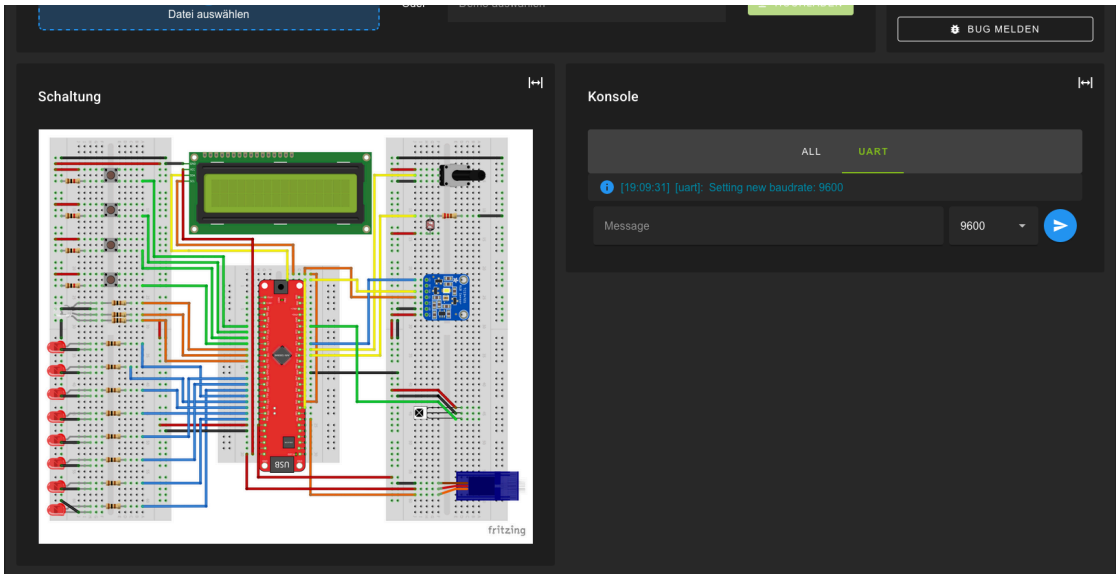


Abbildung 2.11: UART-Konsole und Stationsaufbau einer MICRO-Station

Systeme, die sowohl in industriellen Kontexten als auch in Kombination mit Remote-Laboren eingesetzt werden.

2.3.1 Human-Machine-Interfaces (HMIs)

Human-Machine-Interfaces (HMIs) sind zentrale technologische Komponenten, die als Schnittstelle zwischen Mensch und Maschine dienen. Ihr Hauptzweck ist es, Benutzern den Zugriff auf technische Systeme, Produktionsanlagen oder Prozessleittechnik zu ermöglichen, um Zustände zu überwachen und gezielt auf Prozesse einzuwirken [101]. Dabei stehen meist die grafische Darstellung, intuitive Steuerung und die verständliche Rückmeldung von Systemzuständen im Vordergrund [89, 101]. HMIs reichen von einfachen Bedientafeln mit Tastern und Displays bis hin zu komplexen, softwarebasierten Visualisierungslösungen auf Touchpanels, Computern oder mobilen Endgeräten [101].

Ein HMI erfüllt drei Kernaufgaben: [65, 101]

- Visualisierung von Prozesszuständen, Alarmen und Trenddaten
- Interaktion, also Eingabe von Soll-Werten, Parametern oder Sequenzen
- Feedback, beispielsweise Bestätigung, Fehlermeldung oder Handlungsanweisungen

Entscheidend ist, dass HMIs kognitive und physiologische Eigenschaften ihrer Nutzerinnen und Nutzer berücksichtigen und dadurch eine verständliche, fehlerarme und sichere Prozessführung ermöglichen [89].

2.3.2 Zusammenhang von HMI und SCADA in industriellen Systemen

Supervisory Control and Data Acquisition (SCADA) Systeme und HMIs sind zentrale Komponenten moderner Automatisierung und industrieller Leittechnik. Sie sind eng miteinander verbunden und arbeiten in einer abgestimmten Systemarchitektur, um Bedienern die Überwachung, Steuerung und Optimierung von komplexen, oft verteilten technischen Prozessen zu ermöglichen. [28]

SCADA-Systeme verwalten das gesamte Bild eines Betriebs: Sie sammeln kontinuierlich Sensordaten und steuern Geräte über industrielle Netzwerke und Protokolle. Die dabei generierten Informationen werden zentral verarbeitet und zum Beispiel für Analysen, Visualisierungen und Alarmmanagement bereitgestellt. [28, 107]

Das HMI ist aus Anwendersicht die direkte Schnittstelle zu diesen Daten und Funktionen. Es bietet grafische Darstellungen, Live-Visualisierung von Anlagenzuständen, Protokoll- und Trendanzeigen, sowie Steuerelemente zur aktiven Eingabe von Befehlen. Dadurch wird das HMI zur „Bedienzentrale“, die alle wesentlichen Funktionen des SCADA-Systems zugänglich macht. [28, 107]

2.3.3 Kommerzielle HMI-Systeme

Die industrielle HMI-Landschaft wird von mehreren etablierten Technologieplattformen dominiert, die sich durch ihre Zuverlässigkeit, Skalierbarkeit und Integrationsfähigkeiten auszeichnen.

SIMATIC HMI von Siemens

Siemens bietet eine große Auswahl an HMI- und SCADA-Lösungen [5]. Die HMI Lösungen von Siemens (SIMATIC HMI) zeichnen sich durch eine modulare, skalierbare und durchgängige Plattformstruktur aus, die für verschiedene industrielle Anforderungen konzipiert ist [5]. Eine beispielhafte Darstellung einiger SIMATIC HMI-Panels findet sich in Abbildung 2.12.

Abbildung 2.12: SIMATIC HMIs von Siemens [115]

Dabei sind Hardwarefunktionen, Software-Engineering und Visualisierungskonzepte aufeinander abgestimmt und über alle Gerätefamilien durchgängig nutzbar. Die Bedienoberflächen sind einheitlich gestaltet, bieten grafische Bedienelemente und unterstützen Multitouch-Technologie sowie klassische Tastenbedienung. Die Usability orientiert sich bewusst an modernen Smartphones/Tablets, um eine intuitive und schnelle Bedienung zu ermöglichen. [116]

Alle HMIs lassen sich problemlos in bestehende Automatisierungsnetzwerke integrieren. Schnittstellenvielfalt (z.B. USB, Ethernet) sowie erweiterte Connectivity (Industrieprotokolle) gewährleisten ein nahtloses Zusammenspiel mit Peripheriegeräten. [116]

Die Projektierung aller SIMATIC HMI Geräte erfolgt im TIA Portal, einer Software von Siemens, die zur Programmierung und Konfiguration von SIMATIC-Steuerungen dient. Projekte können einfach zwischen Gerätetypen übertragen oder migriert werden, wodurch Engineering-Aufwand, Fehleranfälligkeit und Inbetriebnahmezeiten reduziert werden [116]. Das TIA Portal basiert softwareseitig auf dem hauseigenen Windows Control Center (WinCC). Somit werden die HMIs mit Windows als unterliegendes Betriebssystem betrieben.

Siemens garantiert langfristige Produktverfügbarkeit, Ersatzteile und Migration für zukünftige Gerätegenerationen. Projekte und Daten lassen sich leicht auf Nachfolgeprodukte übertragen [116]. Zudem sind die Panels speziell für raue Industrieumgebungen

Abbildung 2.13: PanelView 5310 Graphic Terminal von Rockwell Automation [110]

entwickelt und robust gebaut [116].

Insgesamt verfolgt Siemens mit seinem HMI-Portfolio einen integrativen Ansatz: einheitliche Bedienphilosophie, durchgehende Integration, hohe Sicherheit und Skalierbarkeit, kombiniert mit robuster Industrietauglichkeit und modernen Visualisierungsmöglichkeiten für alle Anforderungen.

FactoryTalk View von Rockwell Automation

Mit FactoryTalk View stellt Rockwell Automation ein umfassendes Portfolio an HMI-Lösungen bereit, die sowohl Hardware- als auch Software-Komponenten umfassen und auf industrielle Anforderungen hinsichtlich Produktivität, Flexibilität und Skalierbarkeit ausgerichtet sind [111]. Die Geräte von Rockwell Automation bieten verschiedene Displaygrößen, robuste Bauweise sowie Touch-Eingabe [112]. Abbildung 2.13 zeigt ein beispielhaftes HMI-Modell von Rockwell Automation.

Die FactoryTalk View Reihe spaltet sich auf in zwei untergeordnete Kategorien: [111]

- *FactoryTalk View Machine Edition (ME)*: Für maschinennahe HMIs.
- *FactoryTalk View Site Edition (SE)*: Für verteilte, netzwerkbasierte Leitsysteme und große Anlagen.

Die HMI Lösungen nutzen Windows als unterliegendes Betriebssystem [112]. Für die Entwicklung vonHMI-Software verwendet Rockwell Automation das hauseigene FactoryTalk View Studio [112]. Dadurch lassen sich Anwendungen auch über die gesamte Familie von Rockwell Automation HMIs portieren. Die Entwicklung erfolgt über eine UI basierte Programmieroberfläche und kann mit Visual Basic Skripten (VBA) erweitert werden [111].

Rockwell bietet außerdem weitere Lösungen im HMI Bereich an, etwa die Optix Serie [111], welche eine cloudbasierte HMI-Lösung darstellt. Die Software läuft somit nicht lokal auf dem Gerät, sondern das HMI-Gerät dient ausschließlich als Aus- und Eingabegerät [112].

Rockwell Automation bietet mit den FactoryTalk-Lösungen skalierbare, robuste HMI-Systeme für alle Ebenen der industriellen Visualisierung. Die Entwicklung erfolgt mittels grafischer Werkzeuge und mit Skripting in VBA. Dank neuerer Plattformen wie FactoryTalk Optix steht ein moderner, offener und zukunftsfähiger HMI-Technologie-Stack zur Verfügung, der sowohl klassische Industrieanforderungen als auch aktuelle Web- und Cloud-Trends adressiert.

AVEVA InTouch

AVEVA InTouch HMI zählt zu den weltweit etablierten Softwarelösungen für industrielle HMIs und wird branchenübergreifend zur Überwachung und Steuerung komplexer Prozesse eingesetzt [7].

InTouch HMI basiert softwareseitig auf dem Windows-Betriebssystem und kann entweder als Standalone-Anwendung auf Industrie-PCs oder als Teil eines vernetzten SCADA-Systems betrieben werden. Moderne Versionen unterstützen zusätzlich webfähige Clients [6, 7]. Dadurch lassen sich Visualisierungen sicher und ortsunabhängig via HTML5-kompatible Browser aufrufen [6, 7], wie beispielhaft in Abbildung 2.14 dargestellt.

Die Benutzeroberfläche orientiert sich an einer konsequenten „Situational Awareness“-Philosophie [6]: Die umfangreiche Symbol- und Designbibliothek erleichtert die Realisierung intuitiver und konsistenter Bedienoberflächen, die speziell für die schnelle

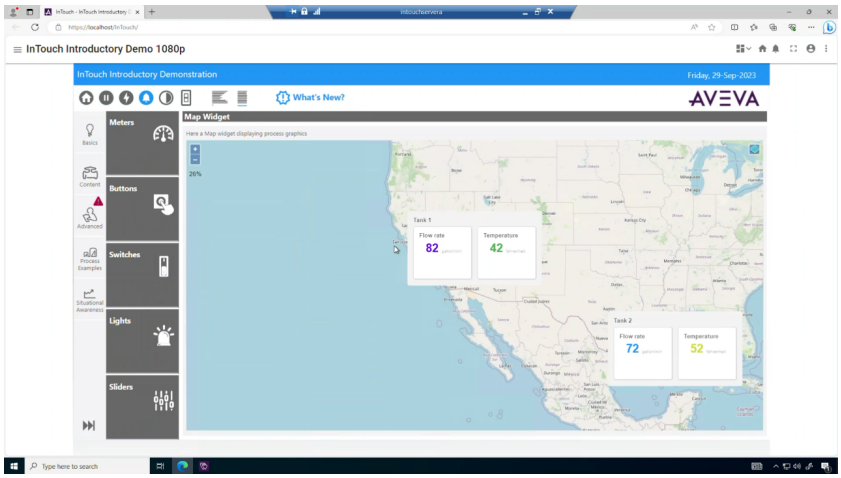


Abbildung 2.14: AVEVA InTouch HTML5 webbasierte HMI Anwendung

Informationsaufnahme und sichere Bedienung im industriellen Umfeld optimiert sind [6].

Die Projektierung der HMIs erfolgt überwiegend grafisch per Drag & Drop in der InTouch Entwicklungsumgebung [6]. Für die Programmierung von Bedienlogik und Automatisierungsfunktionen steht mit „QuickScript“ eine dedizierte Skriptsprache zur Verfügung [8].

InTouch gewährleistet herstellerunabhängige Konnektivität durch die Unterstützung offener Industriestandards wie HTTP und SQL-Datenbankanbindung [6].

Insgesamt bietet AVEVA InTouch HMI eine leistungsfähige, offene und skalierbare Visualisierungsplattform, die flexibel an unterschiedlichste Automatisierungs- und Bedienkonzepte angepasst werden kann und sowohl klassische als auch moderne Anforderungen an industrielle HMIs erfüllt.

2.3.4 Nicht-kommerzielle HMI-Lösungen

Für nicht-kommerzielle Anwendungen, Forschung, Lehre oder Low-Budget-Projekte haben sich mehrere Open-Source-HMI-Plattformen etabliert. Nachfolgend werden FUXA, CuteHMI und OSHMI vorgestellt, jeweils mit Fokus auf Technologie, Merkmale und praktische Einsatzmöglichkeiten.

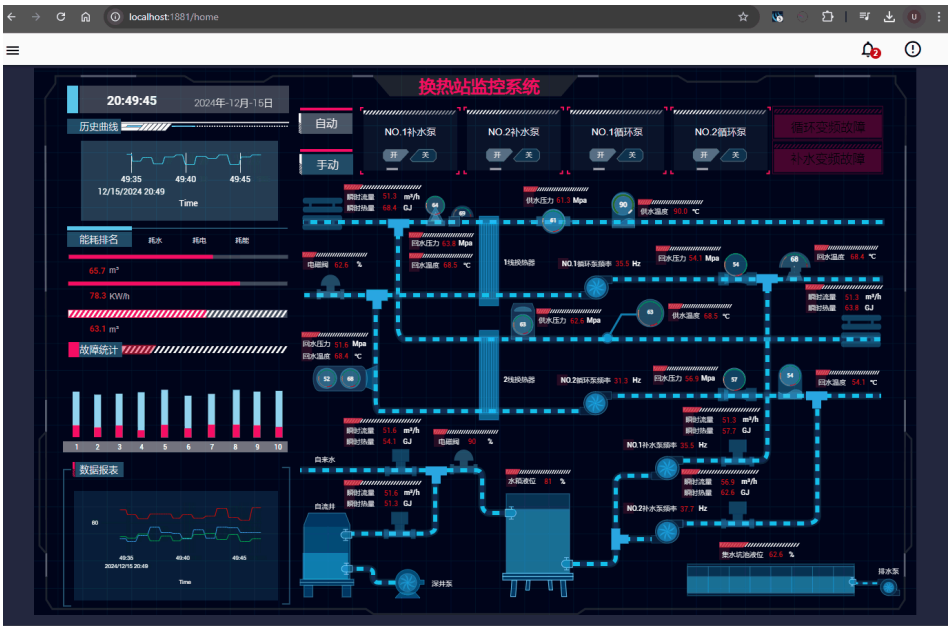


Abbildung 2.15: Beispielhafte FUXA HMI-Anwendung

FUXA

FUXA ist eine komplett webbasierte Open-Source-HMI- und SCADA-Lösung, die sich besonders für kleine bis mittlere Visualisierungsprojekte und DIY-Anwendungen eignet. Sie ist plattformunabhängig und läuft auf allen Systemen, auf denen Node.js ausgeführt werden kann (Linux, Windows, Raspberry Pi). [40, 41]

Die technische Umsetzung basiert auf Node.js und Angular. Die Bedienoberfläche ist vollständig browserbasiert und nutzt Webtechnologien. Visualisierungen werden direkt im Browser mithilfe eines grafischen Editor-Interfaces erstellt, wobei HTML5 und SVG zur Darstellung verwendet werden. Eine beispielhafte FUXA Anwendung findet sich in Abbildung 2.15. [40]

In Bezug auf die Konnektivität unterstützt FUXA gängige Industrieprotokolle wie Modbus RTU/TCP, Siemens S7 Protocol, BACnet IP, MQTT und Ethernet/IP, was eine Kompatibilität mit anderen Geräten und Systemen, auch von anderen HMI-Anbietern, ermöglicht. Viele Geräte werden bereits direkt unterstützt, und über offene Schnittstellen ist eine einfache Erweiterung möglich. [40]

Zu den zentralen Funktionen zählen ein intuitiver Drag-&-Drop-Editor (dargestellt in Abbildung 2.16), animierbare SVG-Grafiken, Benutzerverwaltung, Alarmierung, Trend-Anzeigen sowie ein Multi-User-fähiges Responsive-Design. Projekte lassen sich pro-

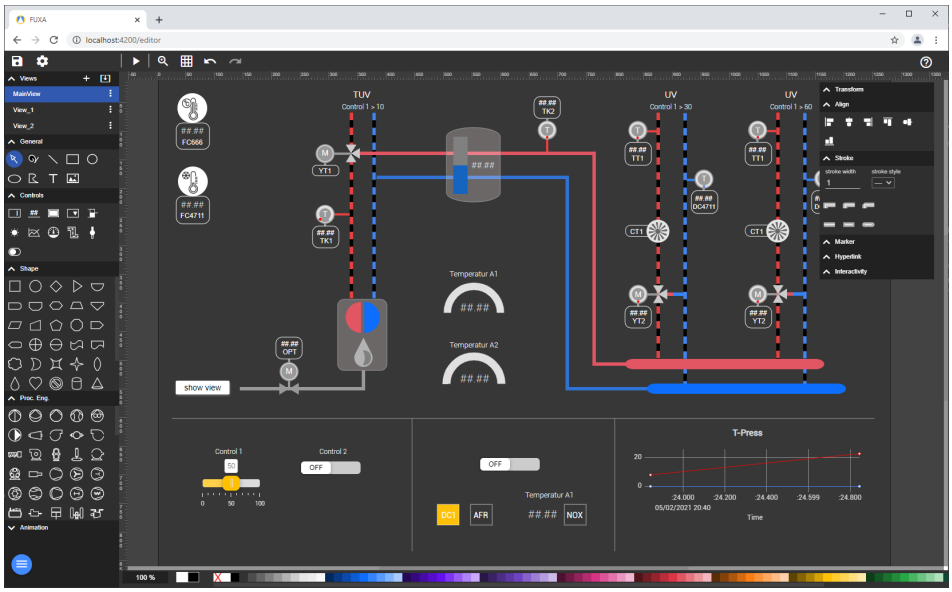


Abbildung 2.16: FUXA HMI Editor [40]

blemlos exportieren und importieren, was den Austausch und die Wiederverwendbarkeit deutlich vereinfacht. [41]

FUXA ist besonders praxistauglich für Anwendungsbereiche wie DIY-Projekte, Prototyping, kleinere industrielle Steuerungen oder als didaktische Plattform im universitären Umfeld. Die Software ist unter der MIT-Lizenz frei verfügbar, wird aktiv weiterentwickelt und steht als Open-Source-Projekt öffentlich auf GitHub [40] zur Verfügung.

CuteHMI

CuteHMI ist ein weiteres flexibles Open-Source-Framework zur Entwicklung von HMIs, mit Fokus auf Modularität, Portabilität und grafische Gestaltung. Es basiert auf Qt (vgl. Unterunterabschnitt 5.1.3) und eignet sich neben klassischen HMI-Anwendungen auch für IoT-, Gebäudemanagement- oder Raummanagement-Anwendungen. [103]

Die betrachtete HMI-Lösung basiert auf dem Qt-Framework und kombiniert C++ mit Qt Modelling Language (QML) zur Definition von Benutzeroberflächen und Logik [104]. Die Architektur ist modular aufgebaut und verwendet ein Plug-in-System, das eine flexible Erweiterbarkeit ermöglicht. Die Gestaltung der grafischen Oberfläche kann wahlweise über Qt Creator oder direkt mittels QML erfolgen [104]. Eine beispielhafte Implementierung findet sich in Abbildung 2.17.

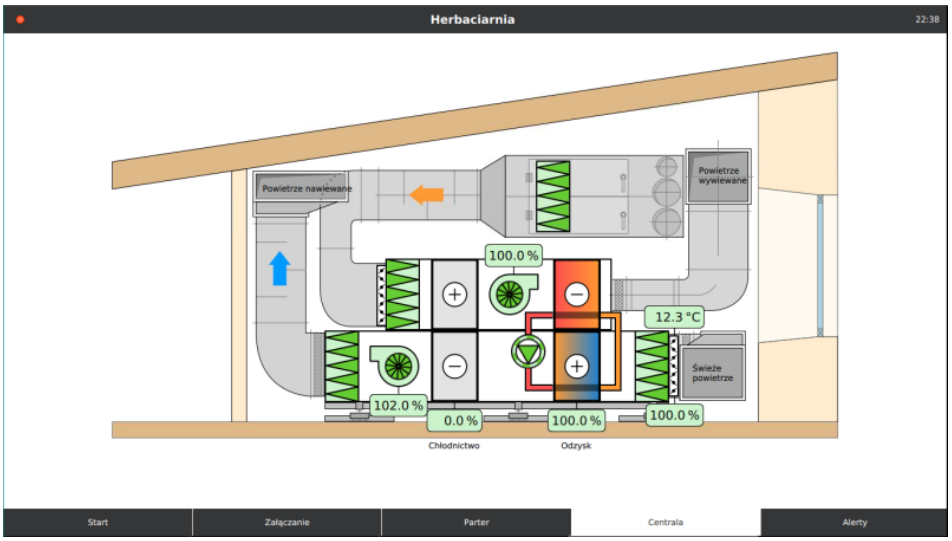


Abbildung 2.17: Beispielhafte HMI-Anwendung in CuteHMI [104]

Hinsichtlich der Konnektivität werden unter anderem Protokolle wie Modbus unterstützt [103]. Weitere gängige Schnittstellen können über zusätzliche Plugins eingebunden werden. Für individuelle Anforderungen lässt sich die Funktionalität zudem durch eigene Projekte in C++ oder QML erweitern.

Die Lösung eignet sich besonders für experimentelle, forschungsnahe oder modular aufgebaute HMI-Projekte. Sie wird kostenfrei unter der LGPL angeboten. Die Projektseite und der Quelltext sind offen zugänglich. [104]

OSHMI (Open Substation HMI)

Open Substation HMI (OSHMI) ist ein plattformübergreifendes HMI-Framework, welches vielseitig als industrielles, oder auch do it yourself (DIY) HMI-Framework verwendet werden kann [97]. Die technische Basis bildet eine vollständig webbasierte Architektur, die auf modernen Webtechnologien wie SVG, HTML5, Javascript, PHP, Lua, QT, SQLite und NGINX aufbaut [95]. OSHMI läuft sowohl unter Windows als auch unter Linux und nutzt Chromium-basierte Browser als Laufzeitumgebung [97]. Ein HMI Beispiel findet sich in Abbildung 2.18.

OSHMI bietet eine breite Protokoll-Konnektivität und unterstützt unter anderem OPC UA, IEC 61850, Modbus, MQTT und BACnet. Durch die Möglichkeit zur Anbindung von Datenbanken wie SQLite oder PostgreSQL eignet sich die Plattform auch für größere SCADA-Architekturen. [95]

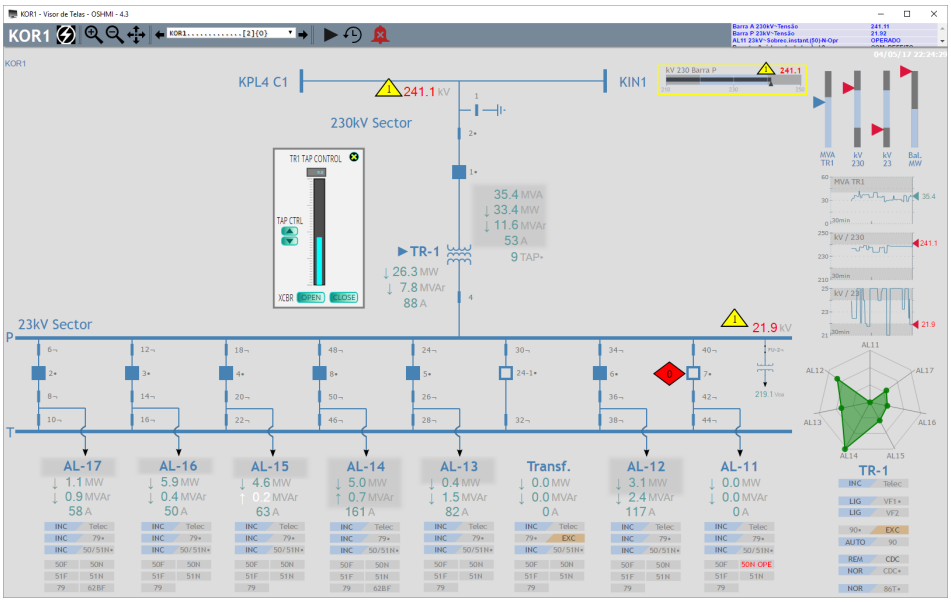


Abbildung 2.18: OSHMI implementation einer HMI-Oberfläche [96]

Als Fazit lässt sich festhalten, dass OSHMI eine leistungsfähige und vielseitig einsetzbare HMI-Lösung darstellt, die sowohl in industriellen Anwendungen als auch im DIY-Bereich überzeugt. Die webbasierte, plattformunabhängige Architektur auf Basis moderner Technologien wie HTML5, SVG und JavaScript ermöglicht eine flexible und zukunftssichere Gestaltung von Benutzeroberflächen. Durch die breite Protokollunterstützung, sowie die Anbindungsmöglichkeiten an relationale Datenbanken eignet sich OSHMI nicht nur für kleine Projekte, sondern auch für komplexe SCADA-Systeme. Die Open-Source-Lizenz, aktive Weiterentwicklung und umfangreiche Dokumentation machen OSHMI zu einer attraktiven Option für Forschung, Lehre und professionelle Anwendungen gleichermaßen.

2.3.5 Zusammenfassung

Die Analyse der aktuellen HMI-Landschaft verdeutlicht die zentrale Rolle von Mensch-Maschine-Schnittstellen in der industriellen Automatisierung. Moderne HMI-Systeme fungieren als Brücke zwischen technischen Prozessen und menschlicher Interaktion und sind oftmals eng mit SCADA-Architekturen verknüpft. In diesem Kontext ermöglichen sie eine intuitive Überwachung, Steuerung und Optimierung komplexer Systeme.

Kommerzielle Anbieter wie Siemens, Rockwell Automation und AVEVA bieten umfas-

Tabelle 2.2: Vergleich gängiger kommerzieller und Open-Source HMI-Systeme

System	**Lizenz**	**Plattform / OS**	**Eingabe**	**Engineering-Tool**	**Zielgruppe**
SIMATIC HMI (Siemens)	Kommerziell	Hardware-Panels, Windows, TIA Portal	Touch, Tasten	TIA Portal (WinCC)	Industrie, Großanlagen
FactoryTalk View (Rockwell)	Kommerziell	Hardware-Panels, Windows, Cloud, FactoryTalk Studio	Touch, Tasten	FactoryTalk Studio, VBA	Industrie, Anlagenbau
AVEVA InTouch	Kommerziell	Windows, Web/HTML5 Clients	Maus, Touch, Tastatur	InTouch Dev. Env., QuickScript	Industrie, SCADA
FUXA	Open Source (MIT)	Node.js, Linux, Windows, Web, ARM/RPi	Maus, Touch (Browser)	Webeditor (HTML5, Angular)	Bildung, DIY, KMU
CuteHMI	Open Source (LGPL)	C++/Qt, Windows, Linux, macOS	Maus, Touch (Qt, QML)	Qt Creator, QML	Forschung, Bildung, Embedded
OSHMI	Open Source (verschiedene)	HTML5, SVG, JS, PHP, Linux/Win, Webbrowser	Maus, Touch (Browser)	Webeditor (SVG/JS), Lua, Plugins	Lehre, Forschung, DIY, Industrie

sende, robuste Plattformen, die insbesondere auf industrielle Großanwendungen und langfristige Produktzyklen ausgerichtet sind. Neben diesen etablierten Industrieanbietern bieten Open-Source-Projekte wie FUXA, CuteHMI und OSHMI attraktive Alternativen, vornehmlich für Lehre, Forschung, Prototyping oder kostensensitive Anwendungen. Eine tabellarische Gegenüberstellung der HMI-Lösungen findet ich in Tabelle 2.2.

Trotz ihrer vielfältigen Funktionen, der bewährten Integrationsfähigkeit und industriellen Robustheit sprechen wesentliche Gründe gegen den Einsatz der beleuchteten kommerziellen und Open-Source-HMI-Systeme für das Remote-Labor MICRO:

Kosten, Lizenzierung und Hardwarebindung

Kommerzielle HMI-Lösungen wie Siemens SIMATIC HMI, Rockwell FactoryTalk und AVEVA InTouch sind mit erheblichen Lizenzkosten verbunden, setzen spezifische proprietäre (Industrie-)Hardware voraus und sind auf leistungsfähige Windows-basierte Plattformen ausgelegt. Damit sind sie weder wirtschaftlich noch praktikabel für Low-Cost-Lösungen wie das MICRO-Labor, dessen Grundidee auf offener, günstiger und flexibler Hardware (z.B. Raspberry Pi) und Software beruht.

Technische Überdimensionierung und Komplexität

Die genannten Systeme zielen auf Großanlagen mit hohem Automatisierungsgrad ab, bieten umfassende Funktionsumfänge (z.B. komplexes Alarmmanagement, Integration ganzer Fertigungslinien, Multiuser- und Rechteverwaltung) und verfolgen lange Produktzyklen. Diese Features gehen deutlich über die Anforderungen eines universitärpädagogischen Remote-Labors hinaus und führen zu unnötiger Komplexität, langer Einarbeitung und erhöhtem Wartungsaufwand. Die Nutzenden des HMIs werden aufgrund des Wechsels der Arbeitskräfte im Labor stetig variieren. So ist eine unnötig komplizierte Einarbeitung durch Verwendung der komplexen HMI Lösungen in das System vermeidbar.

Plattformabhängigkeit und mangelnde Offenheit

Viele Lösungen benötigen spezielle Plattformen (meist Windows) oder eigene Tools, die für die Entwicklung, das Customizing und die Wartung tiefere, oft proprietäre Kenntnisse erfordern. Dies widerspricht dem Anspruch, ein leichtgewichtes, eigenständig

anpassbares und wartbares HMI bereitzustellen, das einfach auf offener Hardware läuft und ohne komplexe Technologien anpassbar ist.

Weder klassische Industrielösungen noch allgemeine Open-Source-HMIs adressieren diese Anforderungen in Kombination. Daher ist eine eigene Entwicklung bzw. eine tiefgreifende Anpassung bestehender Open-Source-Lösungen vorzuziehen. So kann garantiert werden, dass das HMI exakt auf die spezifischen Anforderungen zugeschnitten ist. Diese Argumentationslinie begründet, warum für das Projektkonzept auf den Einsatz fertiger HMI-Systeme verzichtet und stattdessen eine spezialisierte Eigenentwicklung gewählt wird, die maximale Offenheit, Flexibilität, Integrationsfähigkeit und Kosteneffizienz sicherstellt, wie sie für Forschung und Lehre im Bereich Remote-Labore essenziell ist.

2.4 Evaluationsmethoden

Die Bewertung und Sicherstellung der Benutzerfreundlichkeit sowie der UX ist ein zentrales Element bei der Entwicklung von UIs und Human-Machine-Interfaces (HMIs). Die folgenden Abschnitte geben einen Überblick über etablierte Evaluationsmethoden und standardisierte Fragebögen zur UX-Messung, wie sie im Forschungs- und Hochschulkontext Anwendung finden.

2.4.1 Einordnung der Evaluationsmethoden

Die Bewertung von Benutzungsschnittstellen kann anhand unterschiedlicher methodischer Ansätze erfolgen. Häufig werden dabei zwei grundlegende Herangehensweisen unterschieden [29]:

- *Qualitative Methoden* zielen darauf ab, ein tiefgehendes Verständnis der Nutzererfahrung und der zugrunde liegenden Ursachen für beobachtetes Verhalten zu gewinnen. Hierzu zählen beispielsweise Interviews, Beobachtungsstudien oder offene Gruppendiskussionen. Sie liefern detaillierte, kontextbezogene Einblicke, eignen sich jedoch weniger für den direkten Vergleich zwischen Systemen, da sie meist keine statistisch verwertbare Daten erheben. [29]
- *Quantitative Methoden* hingegen erfassen Nutzerwahrnehmung strukturiert und in messbarer Form, vorwiegend durch standardisierte Fragebögen. Diese bieten den Vorteil, größere Nutzergruppen systematisch zu befragen und die Ergebnisse

statistisch auswerten zu können. Dadurch lassen sich belastbare, vergleichbare Aussagen zur Akzeptanz und Erfahrung mit unterschiedlichen Systemen treffen, welche insbesondere im Bereich der User Experience (UX) von großer Bedeutung sind. [29, 69]

Der Einsatz von standardisierten, quantitativen Fragebögen erleichtert eine effiziente Erhebung von Nutzerfeedback. Zudem ermöglichen diese Instrumente die vergleichende Bewertung unterschiedlicher Technologien oder Produktvarianten und können sowohl eigenständig als auch ergänzend zu qualitativen Methoden wie Beobachtungen oder Usability-Tests eingesetzt werden. [29, 69]

2.4.2 Standardisierte UX-Fragebögen

Zu den wichtigsten und in der Praxis am weitesten verbreiteten standardisierten UX-Fragebögen für allgemeine Anwendungen zählen: [67]

- System Usability Scale (SUS)
- AttrakDiff
- User Experience Questionnaire (UEQ)
- modular evaluation of key Components of User Experience (meCUE)

Weitere domänenspezifische Skalen wie SASSI (Subjective Assessment of Speech System Interfaces), SUISQ (Speech User Interface Service Quality) und MOS-X (Mean Opinion Scale Expanded) werden häufiger in spezialisierten Szenarien eingesetzt (z. B. Sprachinterfaces, Servicequalität) [67] und sind somit für diese Arbeit nicht von Relevanz.

System Usability Scale (SUS)

Die SUS ist ein international weitverbreiteter, standardisierter Fragebogen zur Erfassung der wahrgenommenen Gebrauchstauglichkeit (Usability) technischer Systeme [67]. Entwickelt wurde der SUS von John Brooke im Jahr 1996 [17] und besteht aus zehn kompakt gehaltenen Aussagen, die von Nutzerinnen und Nutzern auf einer fünfstufigen Likert-Skala bewertet werden [17]. Die Fragen wechseln sich zwischen positiv und negativ formulierten Items ab, um Antworttendenzen zu minimieren und die Nutzerperspektive breit abzudecken [67].

Der Fragebogen ist dabei bewusst technologie- und domänenübergreifend formuliert, sodass er sowohl für Software, Hardware, Webseiten, Apps als auch für andere interaktive Systeme eingesetzt werden kann [67].

Thematisch legt der SUS ausschließlich Wert auf Kernaspekte der Usability wie Einfachheit, Konsistenz, Vertrauen in das System und Unterstützung bei Aufgaben. Tiefergehende Dimensionen der UX, wie Emotionen, Motivation oder Erlebnisqualität, sind nicht Gegenstand der Erhebung. Daher wird der SUS in wissenschaftlicher und unternehmerischer Praxis häufig als schnelle, erste Einschätzung der Usability genutzt oder in Kombination mit anderen, umfassenderen UX-Fragebögen angewendet. [17, 67]

AttrakDiff

AttrakDiff basiert auf dem Modell von Hassenzahl et al. aus 2003 [58] und misst primär drei Dimensionen: die hedonische Qualität, die pragmatische Qualität sowie die Attraktivität. Die *hedonische Qualität* beschreibt, welche Erlebnisse und emotionalen Ziele ein Produkt ermöglicht, beispielsweise das Gefühl von Kompetenz, Zugehörigkeit, Besonderheit oder Stimulation. Die *pragmatische Qualität* erfasst, inwieweit ein Produkt die Aufgaben der Nutzer effektiv und effizient unterstützt. Die *Attraktivität* gibt an, wie angenehm und anziehend das Gesamtpaket empfunden wird. Die Bewertung erfolgt anhand von 28 semantischen Differenzialen auf einer siebenstufigen Skala. Das Instrument ist besonders etabliert zur Evaluation von Web- und Software-Oberflächen. [67]

Affektive Zustände wie Freude, Frustration oder Zufriedenheit werden durch AttrakDiff nicht direkt erfasst. Das Instrument ist bewusst darauf ausgelegt, keine momentanen Gefühle zu messen. Stattdessen werden solche Emotionen als Konsequenzen aus der Beurteilung der Produktmerkmale verstanden und nicht als eigene Items berücksichtigt. In der Literatur wird daher empfohlen, zusätzliche Skalen einzusetzen, wenn ein umfassendes Bild der UX einschließlich affektiver Aspekte entstehen soll. [67]

User Experience Questionnaire (UEQ)

Der UEQ wurde 2008 speziell von Laugwitz et al. [69] entwickelt, um ein möglichst umfassendes und gleichzeitig schnelles Bild der Nutzererfahrung zu liefern. Er besteht aus 26 bipolaren Adjektiv Paaren, welche auf einer 7-stufigen Skala bewertet werden. Die Paare verteilen sich auf die Skalen „Attraktivität“, „Verständlichkeit“, „Effizienz“, „Zuverlässigkeit“, „Stimulation“ und „Neuheit“ [69]. Der Fragebogen ist für verschiedenste

Softwareprodukte sowie industrielle Anwendungen validiert. Sowohl deutsche als auch englische Versionen sind verfügbar [69]. Für die Evaluation des HMIs in dieser Arbeit wurde die Kurzversion des UEQ verwendet.

meCUE

Das meCUE ist ein modular aufgebauter Fragebogen, der eine umfassende Bewertung der UX für interaktive Produkte und Systeme ermöglicht. Entwickelt wurde meCUE von Minge et al. in 2013 [86], um alle zentralen UX-Dimensionen abzudecken und flexibel auf unterschiedliche Anwendungsszenarien zugeschnitten werden zu können [86].

Der Fragebogen besteht aus insgesamt 34 Fragen, die sich auf vier unabhängig nutzbare Module verteilen. Diese Module decken folgende Aspekte ab [86]:

- *Produktwahrnehmung*: Erfasst Dimensionen wie Nützlichkeit, Usability, visuelle Ästhetik, Status und Bindung an das Produkt.
- *Nutzeremotionen*: Misst explizit sowohl positive als auch negative Emotionen, die während der Produktnutzung entstehen.
- *Konsequenzen*: Bewertet, inwiefern die Nutzungserfahrung zu einer erneuten Nutzungsabsicht führt.
- *Gesamteinschätzung*: Dient der Abgabe eines globalen Gesamturteils über das Produkt.

Jedes Item wird üblicherweise auf einer 7-stufigen Skala beantwortet, wobei ein dediziertes Gesamturteil als semantisches Differenzial vorliegt. Der modulare Aufbau erlaubt eine flexible Kombination der Teilbereiche. In der Praxis können, je nach Evaluationsziel, einzelne oder alle Module zum Einsatz kommen. [29]

2.4.3 Zusammenfassung

Die systematische Evaluation von User Interfaces (UIs) und Human-Machine-Interfaces (HMIs) basiert auf einer fundierten Mischung methodischer Ansätze. Besonders im Forschungs- und Entwicklungsumfeld haben sich standardisierte, quantitative Fragebögen als unverzichtbare Werkzeuge etabliert. Sie ermöglichen eine effiziente, vergleichbare und belastbare Messung sowohl der Usability als auch der User Experience (UX) bei unterschiedlichen Systemen und Nutzergruppen.

Tabelle 2.3: Vergleich der wichtigsten standardisierten UX-Fragebögen zur Evaluation von HMIs

Fragebogen	**Popularität**	**Einsatzgebiet**	**Dimensionen**
SUS	Sehr populär wegen universeller Nutzbarkeit	Produkte, Industrie, Software, Hardware, Webseiten, Apps	Usability (Einfachheit, Konsistenz, Vertrauen), keine Emotionen
AttrakDiff	Etabliert in Forschung, insb. Web/Software	Web/Software, Produktdesign	Hedonische und pragmatische Qualität, Attraktivität, keine direkten Affekte
UEQ	Weit verbreitet in Forschung und Industrie	Software, Industrie, Usability Evaluation	Attraktivität, Verständlichkeit, Effizienz, Zuverlässigkeit, Stimulation, Neuheit
meCUE	Weit verbreitet in Forschung	Forschung, interaktive Produkte	Produktwahrnehmung, Nutzeremotionen, Nutzungskonsequenzen, Gesamteinschätzung

Die wichtigsten Instrumente, wie AttrakDiff, UEQ, meCUE und SUS, decken verschiedene Aspekte der UX ab: von pragmatischen und hedonischen Qualitäten über emotionale und verhaltensbezogene Dimensionen bis hin zur globalen Gebrauchstauglichkeit. Während AttrakDiff und UEQ besonders auf die differenzierte Analyse von Erlebnisqualität und Funktionalität ausgelegt sind, bietet meCUE einen modularen Zugang mit expliziter Erfassung von Emotionen und Nutzungskonsequenzen. Die SUS wiederum überzeugt durch ihre Einfachheit und universelle Anwendbarkeit für die schnelle Erhebung der wahrgenommenen Bedienfreundlichkeit technischer Systeme. Diese Methoden sind auch nochmal tabellarisch in Tabelle 2.3 dargestellt.

Für die Evaluation des entwickelten Touch-basierten HMI im Rahmen dieser Bachelorarbeit wurde die Kurzversion des UEQ verwendet. Die Wahl fiel darauf, weil der UEQ:

- eine schnelle und zugleich umfassende Erfassung der UX ermöglicht,
- sowohl funktionale als auch emotionale Aspekte abdeckt,
- wissenschaftlich validiert ist,
- sich für unterschiedlichste Produkt- und Forschungsszenarien eignet und
- populär in Forschung und Praxis ist.

Damit können die Bedienqualität und Erlebnisdimensionen des HMI gezielt bewertet und mit bestehenden Systemen verglichen werden. Bei spezifischem Interesse an Usability- oder Emotionsaspekten wäre ein ergänzender Einsatz von SUS bzw. meCUE sinnvoll.

3 Hintergrund

Remote-Labore verbinden technische Infrastrukturen mit dem Anspruch, ortsunabhängig nutzbar zu sein. Damit ein solches System reibungslos funktioniert, bedarf es nicht nur robuster Hardware und verlässlicher Software, sondern auch einer klaren, benutzerfreundlichen Schnittstelle für seine Verwaltung. Folgendes Kapitel stellt die dafür zentralen Grundlagen vor, die für das Verständnis des entwickelten HMIs im Kontext des Remote-Labors MICRO notwendig sind. Die Entwicklung eines benutzerfreundlichen, stabilen und funktional integrierten Interfaces setzt ein breites technologisches Fundament voraus, von eingebetteten Systemen über Betriebssystemtechnologien bis hin zu modernen Webtechnologien und Softwarearchitekturen. Neben diesen technologischen Aspekten wird auch der konkrete Projektkontext erläutert, innerhalb dessen das HMI konzipiert und umgesetzt wurde. Ziel ist es, dem Leser eine fundierte Ausgangsbasis zu vermitteln, um die im späteren Verlauf beschriebenen Designentscheidungen, technischen Umsetzungen und Einsatzszenarien nachvollziehen zu können.

3.1 Technologische Grundlagen

Dieses Kapitel beleuchtet zentrale Technologien und Systeme, die für die Entwicklung und Umsetzung des HMIs im Kontext des Remote-Labors MICRO von grundlegender Bedeutung sind.

3.1.1 Eingebettete Systeme

Eingebettete Systeme sind spezialisierte Rechnersysteme, die in technische Geräte oder Anlagen integriert werden, um dort spezifische Steuerungs-, Überwachungs- oder Kommunikationsaufgaben zu übernehmen [11, 26, 60]. Im Gegensatz zu universellen Computern wie PCs sind eingebettete Systeme meist exakt auf ihre Aufgaben zugeschnitten

und zeichnen sich durch begrenzte Ressourcen hinsichtlich Rechenleistung, Energieverbrauch und Speicher aus [60].

Solche Systeme finden sich in einer Vielzahl von Anwendungen. Typische Beispiele umfassen Haushaltsgeräte wie Waschmaschinen oder Mikrowellen, medizinische Geräte, Autos, Industrieanlagen und zahlreiche weitere technische Produkte [26]. Oft agieren sie unsichtbar für die Nutzer und leisten über Sensoren, Aktoren und integrierte Software einen entscheidenden Beitrag zur Automatisierung, Sicherheit und Zuverlässigkeit moderner Technik [60].

3.1.2 Raspberry Pi

Der Raspberry Pi ist ein kostengünstiger, energieeffizienter Einplatinencomputer, der sich durch seine Vielseitigkeit und umfangreiche Community auszeichnet [122]. Besonders im Bildungsbereich erfreut er sich großer Beliebtheit, da er eine niederschwellige Möglichkeit zur Entwicklung eingebetteter Systeme bietet [9]. Der Raspberry Pi eignet sich ideal als Plattform für HMIs, da er neben USB- und GPIO-Schnittstellen auch über HDMI- oder DSI-Anschlüsse zur Displayansteuerung verfügt [122]. Zudem kann man ihn mit einem regulären Betriebssystem betreiben [122], was die Entwicklung vereinfacht. In Kombination mit Touchdisplays ermöglicht er die Entwicklung kompakter Steuergeräte mit grafischer Oberfläche. In dieser Arbeit dient ein Raspberry Pi 5 [78] als Hardware-Basis für das entwickelte Interface.

3.1.3 Kiosk-Ansicht einer Applikation

In der Entwicklung grafischer Benutzungsschnittstellen für eingebettete Systeme wird häufig das Konzept der sogenannten Kiosk-Ansicht (engl. kiosk mode), auch Kiosk-Modus genannt, eingesetzt. Dabei handelt es sich um eine speziell konfigurierte Betriebsart, bei der das Gerät ausschließlich eine fest definierte Anwendung im Vollbildmodus startet und sämtliche sonstigen Systemfunktionen für den Nutzer ausgeblendet oder gesperrt werden [94]. Ziel ist es, eine kontrollierte und störungsfreie Benutzererfahrung zu ermöglichen, bei der die Nutzerinnen und Nutzer ausschließlich mit der vorgesehenen Oberfläche interagieren können [61, 131]. Im Kontext des hier entwickelten HMIs bedeutet dies, dass beim Start des Raspberry Pi automatisch ein Webbrowser im Vollbildmodus geladen wird, der direkt die Weboberfläche des HMIs anzeigt. Zugriffe auf das Betriebssystem oder andere Anwendungen sind dabei nicht vorgesehen.

Auch klassische Navigationselemente des Browsers wie Adresszeile, Tabs oder Kontextmenüs sind deaktiviert. Nutzerinnen und Nutzer können somit nur die vorgesehenen Funktionen des HMIs verwenden.

Durch die Kiosk-Ansicht werden eine Großzahl von möglichen Fehleingaben und ungewollten Eingaben der Nutzerinnen und Nutzer vermieden, welche z. B. die eigentliche HMI-Software beeinträchtigen könnten [131]. Zudem wirkt das System durch die Verschleierung der Browser und Betriebssystem eigenen Schaltflächen dem Nutzungskontext entsprechend einfacher und minimalistischer. Nutzer können sich so besser auf ihre Aufgaben konzentrieren [61, 131].

3.1.4 Display-Server in Linux

Ein Display-Server übernimmt im Linux-Betriebssystem die zentrale Aufgabe, grafische Inhalte auf dem Monitor darzustellen und Eingaben von Geräten wie Maus oder Tastatur entgegenzunehmen. Anwendungen (auch Clients genannt) senden ihre grafischen Ausgaben nicht direkt an die Hardware, sondern übergeben diese an den Display-Server. Dieser sorgt dafür, dass die gewünschte Darstellung auf dem Bildschirm erscheint und leitet Benutzereingaben gezielt an die jeweils aktive Anwendung weiter, was eine geordnete Interaktion zwischen Programm und Benutzer ermöglicht. [15, 16]

Die Kommunikation zwischen den Anwendungen und dem Display Server erfolgt im Regelfall über standardisierte Netzwerkprotokolle [16]. Dadurch ist es selbst möglich, grafische Benutzeroberflächen von entfernten Rechnern zu nutzen [16].

Ein klassisches Beispiel für einen Display-Server unter Linux ist der X-Server, der das X11-Protokoll nutzt. Hierbei sind die Rollen von Client und Server klar getrennt: Die grafischen Anwendungen agieren als Clients, der X-Server ist die Vermittlungsinstanz, die alle Aktionen koordiniert. Durch das Netzwerkprotokoll lassen sich Anwendungen auch über das Netzwerk starten, wobei die grafische Ausgabe auf einem anderen Computer als dem, auf dem das Programm ausgeführt wird, angezeigt werden kann. [16]

Ein moderner Ansatz ist Wayland, der im Gegensatz zu X11 eine engere Kopplung von Display Server und Anwendung vorsieht [91]. Dies führt zu einer einfacheren Architektur, geringerer Latenz sowie besseren Sicherheitsmechanismen, da die Anwendungen weniger Möglichkeiten haben, gegenseitig in ihre Ausgaben oder Eingaben einzugreifen [91].

Zusammengefasst ist der Display-Server eine essenzielle Schicht im Linux-System, der die Grundlage für jede grafische Interaktion bildet und die Zusammenarbeit aller grafischen Anwendungen ermöglicht. Im Kontext des hier entwickelten HMIs wurde Wayland als Display-Server verwendet.

3.2 Eingesetzte Softwaretechnologien

Für die Realisierung des HMIs kamen verschiedene Softwaretechnologien zum Einsatz, die im Folgenden näher beschrieben werden. Die Auswahl und Kombination dieser Technologien basiert auf den spezifischen Anforderungen des Projekts: Plattformunabhängigkeit (N-R3), Benutzerfreundlichkeit (N-R5), Wartbarkeit (N-R9) und eine robuste, skalierbare Architektur (N-R10). Von Schnittstellenarchitekturen bis hin zu Containerisierung, Webserver-Management und Versionskontrolle, jede dieser Technologien trägt gezielt dazu bei, die Funktionalität und Zuverlässigkeit der Anwendung sicherzustellen. Dieses Kapitel liefert einen Überblick über die verwendeten Werkzeuge, Konzepte und Systeme und verdeutlicht, wie sie in die Gesamtarchitektur der Anwendung integriert wurden.

3.2.1 REST-Architektur

Das in dieser Arbeit entwickelte HMI setzt auf Representational State Transfer (REST) für die Kommunikation mit anderen Komponenten des Remote-Labors. REST ist ein Architekturstil für die Gestaltung verteilter Systeme, insbesondere für das Web. Er wurde 2000 von Roy Thomas Fielding im Rahmen seiner Dissertation definiert [38] und diente als grundlegendes Designmodell für die Weiterentwicklung und Standardisierung der Webarchitektur, darunter grundlegende Protokolle wie HTTP und URI [36, 38].

REST basiert auf einer Reihe von architektonischen Einschränkungen, die gemeinsam bestimmte gewünschte Eigenschaften wie Skalierbarkeit, lose Kopplung, allgemeine Schnittstellen und die Förderung von Zwischenkomponenten ermöglichen. Die wichtigsten Prinzipien sind: [38]

- *Ressourcenorientierung*: Alles Wesentliche wird als Ressource modelliert. Eine Ressource kann dabei alles sein, was referenzierbar ist, von physischen Objekten bis zu abstrakten Konzepten. Ressourcen werden eindeutig durch URIs identifiziert.

- *Repräsentationen*: Der Zustand einer Ressource wird über verschiedene Repräsentationen (z. B. JSON, XML, HTML) abgebildet und übertragen.
- *Zustandslosigkeit*: Interaktionen zwischen Client und Server sind zustandslos. Jede Anfrage beinhaltet sämtliche nötigen Informationen. Der Server speichert keinen Sitzungszustand. Dies erleichtert das horizontale Skalieren und die Ausfallsicherheit.
- *Einheitliche Schnittstelle*: Die Interaktion erfolgt über eine standardisierte, uniform definierte Schnittstelle. Für Webservices heißt das konkret: HTTP-Methoden wie GET, POST, PUT und DELETE haben festgelegte Bedeutungen.
- *Zwischenschaltung von Komponenten*: Komponenten wie Caches, Gateways und Proxys können nahtlos eingebunden werden, um z. B. die Netzwerkbelastung zu verringern oder Sicherheits- und Protokollfunktionen bereitzustellen.

Im Kontext von REST repräsentieren die HTTP-Methoden GET, POST, PUT und DELETE die vier Hauptaktionen, mit denen Clients mit Ressourcen auf einem Server interagieren können [36, 37]. Später wurde noch die PATCH Aktion eingeführt, um eine genauere Differenzierung zwischen der Erstellung neuer Ressourcen, und dem Aktualisieren einer Bestehenden zu schaffen [33]. Jede dieser Methoden hat eine klar definierte Bedeutung und Funktion: [33, 36, 37]

- *GET*: Ressource abrufen (lesen); verändert Daten nicht.
- *POST*: Neue Ressource anlegen oder untergeordnete Aktionen auslösen.
- *PUT*: Ressource ersetzen, aktualisieren oder neu anlegen.
- *DELETE*: Ressource löschen.
- *PATCH*: Ressource aktualisieren.

Die Anwendung der REST-Prinzipien führt zu einer Architektur, in der Komponenten unabhängig voneinander deployt und weiterentwickelt werden können. Das fördert Interoperabilität, Wiederverwendbarkeit und erleichtert die Integration neuer Funktionalitäten [38, 36]. RESTful Webservices sind heute Standard für die Anbindung moderner vernetzter Systeme, da sie mit bestehenden Webtechnologien harmonieren und gut skalieren [3].

Im Vergleich zu alternativen Modellen, wie Remote Procedure Call (RPC)-basierten Modellen, ist REST weniger komplex und ermöglicht eine bessere Entkopplung [36]. Durch

die feste Semantik der HTTP-Methoden und das Adressieren jeder Ressource durch einen eindeutigen URI wird die Integration verschiedener Systeme vereinfacht und die Wartbarkeit erhöht [36].

Zusammengefasst definiert REST ein universelles Modell für die Interaktion in verteilten Systemen, das Skalierbarkeit, lose Kopplung und einfache Erweiterbarkeit unterstützt. Durch die Anwendung von REST entstehen langlebige, gut wartbare und webtaugliche Systemarchitekturen, was den Stil insbesondere für moderne Webanwendungen prädestiniert.

3.2.2 Docker

Im Rahmen dieser Arbeit wird Docker zur Bereitstellung der Anwendung verwendet. Docker [62] ist eine Open-Source-Plattform für die Entwicklung, das Deployment und den Betrieb von Anwendungen in sogenannten Containern [105]. Container sind leichtgewichtige, portable und isolierte Umgebungen, in denen Anwendungen inklusive aller benötigten Abhängigkeiten gebündelt werden [105]. Dadurch kann Software konsistent auf unterschiedlichen Rechnern und Servern ausgeführt werden, unabhängig davon, welches Betriebssystem oder welche Konfiguration dort herrscht [105].

3.2.3 NGINX

NGINX [92] (ausgesprochen: „engine x“) ist ein Alternative zu bewährten Technologien wie der Apache HTTP Server [39] und ist ein leistungsfähiger, quelloffener Webserver, der häufig zum Ausliefern von Webseiten, als Reverse Proxy, Load Balancer und HTTP-Cache eingesetzt wird. Ursprünglich wurde NGINX entwickelt, um die Performance-Probleme anderer Webserver zu lösen und eine noch höhere Skalierbarkeit bei gleichzeitig geringem Ressourcenverbrauch zu erreichen. [117]

Einige relevanten Aufgaben, die NGINX erfüllt, sind folgende: [117, 93]

- *Webserver*: NGINX dient primär als Server für statische und dynamische Inhalte im Web. Durch eine effiziente, asynchrone Ereignisverarbeitung kann NGINX viele tausend gleichzeitige Anfragen mit sehr wenig Arbeitsspeicher bearbeiten.
- *Reverse Proxy*: Oft agiert NGINX als Vermittler zwischen Clients und mehreren nachgelagerten Applikationsservern (z.B. für Web-Apps in Node.js, Python, PHP).

- *Load Balancer*: NGINX verteilt Webanfragen intelligent auf mehrere Server, um Verfügbarkeit, Performance und Ausfallsicherheit zu verbessern.
- *Caching*: NGINX kann Inhalte zwischenspeichern, um die Anzahl der Anfragen an Backend-Systeme zu reduzieren und die Antwortzeiten weiter zu verkürzen.
- *SSL/TLS-Termination*: NGINX übernimmt die Entschlüsselung gesicherter Verbindungen (HTTPS) und entlastet so die nachgelagerten Systeme.

NGINX ist heute eine der beliebtesten Lösungen zur Bereitstellung moderner Webinhalte und Application Programming Interfaces (APIs) und bildet in vielen Hochlast-Webarchitekturen das Rückgrat für eine zuverlässige, effiziente und sichere Auslieferung von Anwendungen und Diensten [117]. NGINX dient in dieser Arbeit als HTTP Server für die entwickelte HMI-Anwendung.

3.2.4 Git und GitLab

Git ist ein verteiltes Versionskontrollsystem, das ursprünglich 2005 von Linus Torvalds für die Entwicklung des Linux-Kernels entworfen wurde [76]. Ziel war es, ein leistungsfähiges Tool bereitzustellen, das effiziente Zusammenarbeit und parallele Entwicklung ermöglicht [76]. Im Gegensatz zu traditionellen, zentralisierten Versionskontrollsystemen wie CVS oder Subversion, besitzt jeder Git-Nutzer eine vollständige Kopie des Repositorys inklusive der gesamten Historie der Projektdaten [76, 118]. Das bedeutet, dass Entwickler auch ohne Netzwerkverbindung Änderungen lokal committen und ihre Historie durchsuchen können. Erst beim „Push“ werden Änderungen an ein zentrales (oder gemeinsames) Remote-Repository übertragen [76, 118].

Genau so ein Remote-Repository kann auf z. B. **GitLab** angelegt werden. GitLab ist eine webbasierte DevOps-Plattform für Softwareentwicklung, die auf Git als Versionsverwaltung aufbaut [18]. GitLab bietet zusätzlich zu den reinen Git-Funktionen eine zentrale Umgebung für kollaborative Softwareentwicklung [18].

Ein Alleinstellungsmerkmal von GitLab ist das „All-Remote“-Unternehmensmodell. Bei der Nutzung von GitLab können alle Mitarbeitenden vollständig remote und asynchron arbeiten, d.h. von beliebigen Orten und ohne definierten Arbeitszeitrahmen [18]. Dies wird durch die strikte Nutzung von GitLab-Workflows ermöglicht. Zum Beispiel müssen Änderungen an Code immer nach dem GitLab-typischen Git-Workflow (Fork, Commit, Merge-Request, Code-Review, Merge) erfolgen, sollte dies so in GitLab eingerichtet sein [18].

Andere Kommunikationswege wie E-Mails werden so weit wie möglich minimiert. Es wird auf maximal transparente, nachvollziehbare, asynchrone Kommunikation gesetzt [18]. In Bezug auf diese Arbeit wurden Git und GitLab zur Verwaltung und Versionierung der Software eingesetzt.

3.2.5 DevOps und CI/CD

DevOps beschreibt einen Ansatz in der Softwareentwicklung, bei dem Entwicklung (Development) und IT-Betrieb (Operations) eng zusammenarbeiten. Ziel ist es, durch Automatisierung, kollaborative Prozesse und kontinuierliches Feedback die Softwarequalität zu erhöhen und schnellere, zuverlässigere Auslieferungen zu ermöglichen. DevOps fördert eine Kultur der gemeinsamen Verantwortung für den gesamten Lebenszyklus einer Anwendung. [34]

Continuous Integration (CI) und Continuous Deployment (CD) sind Vorgehensweisen der modernen Softwareentwicklung, die darauf abzielen, Entwicklungs- und Auslieferungsprozesse zu automatisieren, die Softwarequalität zu verbessern und eine schnelle Bereitstellung neuer Funktionen oder Korrekturen zu ermöglichen [114]. Diese Praktiken sind eng miteinander verzahnt und bilden den Kern einer effizienten DevOps- und agilen Entwicklungsumgebung [114].

CI bezeichnet die fortlaufende und automatisierte Integration von Änderungen (z.B. Quellcode, Konfigurationsdateien) in ein zentrales Repository [34, 114]. Entwickler führen mehrmals täglich Integrationen (oft „Commits“ genannt) durch. Jede Integration löst automatisch einen Build- und Testprozess aus [34]. Durch diese Prozesse können, noch bevor diese in Produktion kommen, Fehler und Sicherheitsrisiken im System entdeckt, und anschließend behoben werden [34].

Beim CD wird nach erfolgreichem Durchlaufen der Tests jede Änderung automatisiert bis in die Produktionsumgebung ausgerollt. Damit werden neue Features oder Verbesserungen sofort für Endnutzer verfügbar gemacht, sobald der Code bereitsteht und alle Qualitätskriterien erfüllt sind. [34, 114]

Zusammenfassend stellt DevOps einen ganzheitlichen Ansatz dar, der durch enge Zusammenarbeit zwischen Entwicklung und Betrieb sowie durch Automatisierung und kontinuierliches Feedback eine hohe Softwarequalität und kurze Release-Zyklen ermöglicht. In dieser Arbeit wurden CI und CD Prozesse entworfen, um die Auslieferung der Software zu vereinfachen.

4 Anforderungen und Konzept

Die erfolgreiche Umsetzung eines Human-Machine-Interfaces (HMIs) für das MICRO-System erfordert ein durchdachtes technisches und gestalterisches Gesamtkonzept. Dieses Kapitel legt die konzeptionellen Grundlagen der Anwendung fest und definiert die Anforderungen, die sich aus dem Nutzungskontext, der Zielgruppe sowie den eingesetzten Technologien ergeben. Auf dieser Basis werden zentrale Designentscheidungen sowie die Systemarchitektur entwickelt, die die spätere Implementierung strukturieren und leiten.

4.1 Anforderungsanalyse

Die Entwicklung eines benutzerfreundlichen und funktional zuverlässigen HMIs für ein hybrides Remote-Labor stellt spezifische technische und gestalterische Anforderungen. In diesem Kapitel werden diese Anforderungen systematisch erfasst und analysiert. Auf Basis dieser Analyse werden grundlegende Designentscheidungen und die Systemarchitektur abgeleitet, die als Grundlage für die spätere Implementierung dienen.

Die in dieser Arbeit definierten Anforderungen basieren auf einer Kombination aus theoretischer Analyse und praxisnaher Abstimmung. Ausgangspunkt bildeten die in Abschnitt 1.2 beschriebenen Problemstellungen, aus denen die in Abschnitt 1.3 formulierten Ziele abgeleitet wurden. Diese Ziele wurden anschließend in konkrete, überprüfbare Anforderungen überführt. Ergänzend flossen Erkenntnisse aus Gesprächen mit dem Projektleiter des MICRO Remote-Labors ein, um sicherzustellen, dass die Anforderungen nicht nur den fachlichen Rahmenbedingungen, sondern auch den praktischen Gegebenheiten des Projektkontexts entsprechen. Das Ergebnis ist folgender Anforderungskatalog, der sowohl die technischen als auch die nutzerorientierten Aspekte des geplanten Systems berücksichtigt.

4.1.1 Funktionale Anforderungen

Das HMI für das MICRO-Remote-Labor muss folgende funktionale Anforderungen erfüllen:

F-R1 Das HMI muss den aktuellen Reservierungs- und Verfügbarkeitsstatus und weitere Metadaten verschiedener Laborstationen darstellen können.

F-R2 Das System soll Nutzermeldungen anzeigen und eine Bearbeitung dieser Meldungen ermöglichen.

F-R3 Die RGB-Beleuchtung der Laborstationen muss über die Benutzeroberfläche steuerbar sein.

F-R4 Für sicherheitskritische oder administrative Aktionen (z. B. die Bearbeitung von Reports) sollen Schutzmechanismen wie Bestätigungsdialoge implementiert werden.

4.1.2 Nicht-funktionale Anforderungen

Zusätzlich zu den funktionalen Anforderungen sollen folgende nicht-funktionale Kriterien erfüllt werden:

N-R1 Das HMI soll als Raspberry Pi (siehe Unterabschnitt 3.1.2) mit angeschlossenem kapazitivem Touch-Display entwickelt werden.

N-R2 Die Benutzeroberfläche soll responsiv und nach dem Prinzip des Mobile-First-Designs gestaltet werden, um eine optimale Bedienbarkeit auf Touch-Displays sicherzustellen.

N-R3 Die Software soll die Möglichkeit offenlassen, auf andere Geräte und Systeme (z. B. ein anderes Betriebssystem oder auf mobile Geräte) portiert werden zu können.

N-R4 Farbgebung, Typografie und Layout des HMIs sollen sich an dem Corporate Design der THM orientieren [87] und zu dem Designschema der restlichen Systemkomponenten passen.

N-R5 Das System muss intuitiv bedienbar sein. Auch für technisch versierte Zielgruppen wie Studierende der Elektrotechnik oder des Maschinenbaus, die jedoch nicht notwendigerweise softwareaffin sind.

N-R6 Das HMI muss zentrale Funktionen über große, einfach berührbare Elemente bereitstellen, die speziell für Touch-Displays ausgelegt sind.

N-R7 Die Navigationsstruktur soll flach gehalten sein, sodass gewünschte Funktionen mit maximal zwei bis drei Interaktionen erreichbar sind.

N-R8 Eine hohe Zuverlässigkeit und kurze Reaktionszeiten sind erforderlich, um eine flüssige Bedienung zu gewährleisten.

N-R9 Das System muss leicht wartbar und dokumentiert sein, um spätere Anpassungen oder Fehlerbehebungen zu erleichtern.

N-R10 Die Softwarearchitektur soll so gestaltet sein, dass zukünftige Funktionserweiterungen (z. B. neue Gerätesteuerungen oder Visualisierungsfunktionen) mit geringem Aufwand realisierbar sind.

4.1.3 Zielgruppen und Nutzungskontext

Die Hauptzielgruppe des HMIs sind Studierende technischer Studiengänge, die im Rahmen ihrer Tätigkeit im MICRO-Projekt mit dem MICRO-System arbeiten. Personen diese Zielgruppe werden im Verlauf der Arbeit auch als „MICRO-Administrator“ bezeichnet. Die Nutzung erfolgt vorwiegend im Labor vor Ort, wobei das Interface über ein lokal angebundenes Touch-Display direkt an dem MACRO-Schrank (Unterabschnitt 2.2.3) verfügbar ist.

Der Nutzungskontext ist geprägt von einer gewissen Geräuschkulisse im Raum und gelegentlich wechselnden Nutzern. Das HMI muss daher auf einfache Navigation und robuste Interaktion unter realen Bedingungen ausgelegt sein.

4.2 Vorgehen zur Technologieauswahl

Die systematische Auswahl geeigneter Technologien stellt einen kritischen Erfolgsfaktor für die Entwicklung eines Touch-basierten HMIs dar. Ein strukturiertes Vorgehen gewährleistet dabei nicht nur die technische Realisierbarkeit des Systems, sondern auch dessen langfristige Wartbarkeit und Erweiterbarkeit. Das folgende Kapitel beschreibt die konzeptionelle Herangehensweise, die zu betrachtenden Aspekte des HMIs und die zugrundeliegenden Entscheidungskriterien für die Technologieauswahl, welche in Abschnitt 5.1 durchgeführt wird.

4.2.1 Evaluationsstrategie

Die Auswahl geeigneter Technologien für das HMI-System orientiert sich nicht allein an einzelnen technischen Eigenschaften, sondern an einem ganzheitlichen Abgleich mit den Projektanforderungen. Grundlage bildet dabei eine anforderungsbasierte Betrachtung, bei der die funktionalen und nicht-funktionalen Anforderungen aus Abschnitt 4.1 als zentrale Maßstäbe dienen.

Die Strategie besteht darin, verfügbare Technologien systematisch in Bezug auf diese Anforderungen zu prüfen und ihre Eignung für den konkreten Anwendungskontext herauszuarbeiten.

Durch dieses Vorgehen wird sichergestellt, dass die Auswahl nicht punktuell oder opportunistisch erfolgt, sondern in einem transparenten Prozess, der auf die langfristige Nutzbarkeit und Erweiterbarkeit des Systems ausgerichtet ist.

4.2.2 Komponentenidentifikation

Die Entwicklung eines Touch-basierten HMI-Systems erfordert Entscheidungen über die Wahl geeigneter Technologien in drei Bereichen:

Hardware-Ebene: Die Auswahl der physischen Systemkomponenten umfasst die Bestimmung der Rechenplattform (Einplatinencomputer, Tablet-Computer oder spezialisierte Embedded-Systeme) sowie der Ein- und Ausgabegeräte (Touch-Displays verschiedener Größen und Technologien).

System-Ebene: Die Betriebssystemwahl bestimmt die verfügbaren Entwicklungstools, Treiber-Unterstützung und Systemstabilität. Dabei müssen Aspekte wie Ressourcenverbrauch, Touch-Unterstützung, und die Kompatibilität mit der gewählten Hardware berücksichtigt werden.

Anwendungs-Ebene: Die Auswahl des Software-Frameworks definiert die Entwicklungseffizienz, Wartbarkeit und zukünftige Erweiterbarkeit des Systems. Dabei stehen verschiedene Ansätze zur Verfügung.

4.2.3 Bewertungskriterien

Für die Untersuchung der Technologien sollen verschiedene inhaltliche und organisatorische Aspekte berücksichtigt werden. Dabei fließen sowohl funktionale Gesichtspunkte ein, die sich an den in Abschnitt 4.1 beschriebenen Systemanforderungen orientieren, als auch Qualitätsaspekte, die unter anderem Wartbarkeit, Portierbarkeit und Benutzerfreundlichkeit umfassen. Zusätzlich sollen projektrelevante Faktoren wie Lizenzbedingungen, Kosten, die Verfügbarkeit von Dokumentation sowie Unterstützung durch die Community einbezogen werden. Darüber hinaus können bestimmte Eigenschaften, wie beispielsweise inkompatible Lizenzmodelle oder fehlende Hardwareunterstützung, zum Ausschluss einer Technologie führen.

4.3 Konzeption der Systemarchitektur

Dieses Kapitel beschreibt die konzeptionellen Grundlagen des HMI-Systems, von der verwendeten Hardwareplattform bis hin zur internen Softwarestruktur und dem Design des Web-Interfaces. Ziel ist es, einen ganzheitlichen Überblick über Aufbau, Funktionsweise und Gestaltungsprinzipien des Systems zu geben, um dessen technische Architektur sowie die dahinterliegenden Überlegungen nachvollziehbar darzustellen.

4.3.1 Integration in das bestehende MICRO-System

Das HMI soll als Vermittlungsstelle zwischen MICRO-Administratoren und dem zugrunde liegenden Systemverbund agieren und muss so mit mehreren Bestandteilen des MICRO-Systems interagieren und gut in dieses integriert werden.

Abbildung 4.1 zeigt ein Schaubild, das die erforderliche Integration des HMIs in das MICRO-System veranschaulicht. Das HMI soll sowohl mit dem Produktions-Backend, als auch mit dem Entwicklungs-Backend des MICRO-Systems kommunizieren. Es soll dabei in der Lage sein, Stationsdaten von beiden Backends abzufragen sowie Statusänderungen und Steuerbefehle zuverlässig weiterzuleiten. Die Steuerung einzelner Stationen soll direkt durch das HMI erfolgen. Über die Benutzeroberfläche soll die Aktivierung, Deaktivierung sowie die Änderung von Stationsattributen möglich sein. Alle Interaktionen mit den MICRO-Stationen werden durch die jeweiligen Backend-Systeme vermittelt, sodass eine Trennung und Sicherung der Kommunikationsflüsse gewährleistet ist.

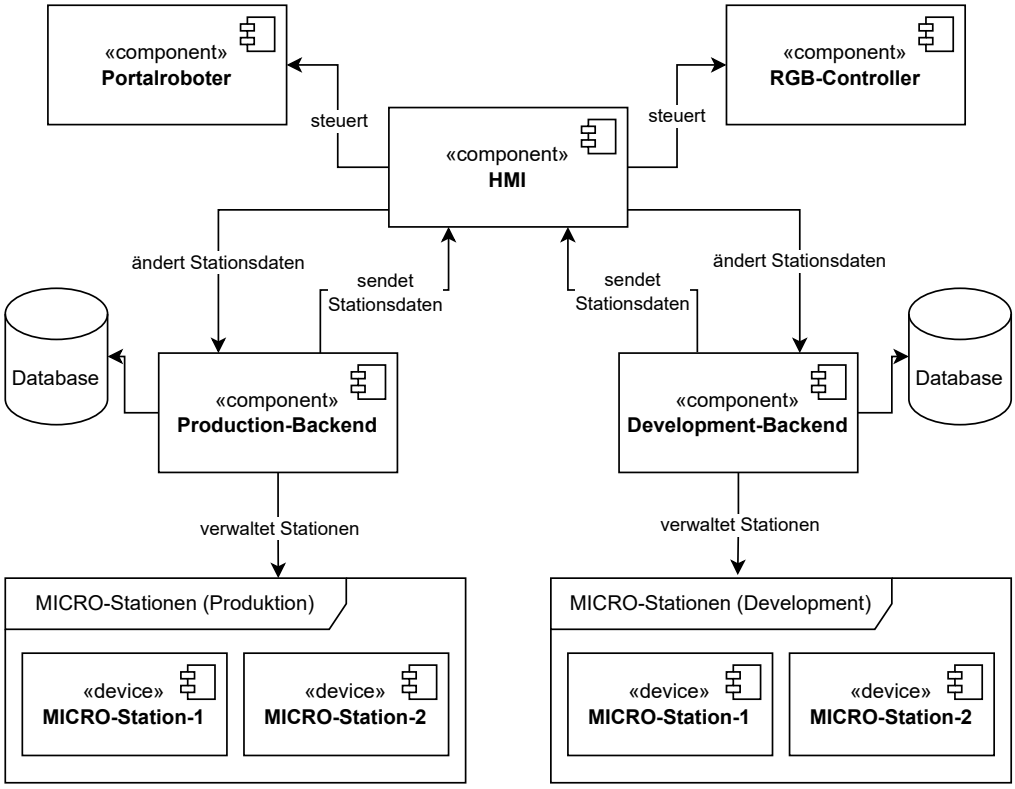


Abbildung 4.1: Integrationsdiagramm des HMI in das bestehende MICRO-System

Für die erweiterte Interaktion mit dem Gesamtsystem soll das HMI eine direkte Verbindung zu Zusatzkomponenten, wie dem RGB-Controller und perspektivisch auch zum Portalroboter, herstellen. Die Steuerung dieser Komponenten erfolgt durch separate Services, welche unabhängig vom Haupt-Backend angesprochen werden. Damit soll das HMI beispielsweise die visuelle Rückmeldung des Laboraufbaus durch RGB-LEDs steuern, sowie künftig Bewegungsbefehle an den Portalroboter senden können.

4.3.2 Konzeption der HMI-Anwendung

Die gesamte Anwendung soll als Web-Service auf einem Raspberry Pi mit angeschlossenem Touch-Display betrieben werden. Der Betrieb erfolgt containerisiert mithilfe von Docker, sodass eine einfache Wartung und Updates durch den Einsatz automatisierter CI/CD-Pipelines ermöglicht werden.

Zusammenfassend soll die Architektur die folgenden Ziele erreichen, um den in Abschnitt 4.1 beschriebenen Anforderungen gerecht zu werden:

- Klare Trennung von Benutzeroberfläche, Backend und Hardwareansteuerung
- Kommunikation ausschließlich über REST -Schnittstellen oder andere relevante Kommunikationsprotokolle
- Unabhängige Anbindung von Produktions- und Entwicklungsumgebungen
- Direkte Steuerung und Rückmeldung zentraler Systemkomponenten wie RGB-Controller und Portalroboter
- Containerisierter Betrieb und automatisierte Bereitstellung auf dedizierter Hardware

Dadurch soll das HMI eine robuste, zukunftsweisende und benutzerfreundliche Plattform zur Verwaltung, Steuerung und Überwachung der Laborinfrastruktur von MICRO bilden und sich flexibel in den bestehenden Systemverbund integrieren lassen.

4.3.3 Designkonzept des HMI Web-Interfaces

Das hier vorgestellte Kapitel basiert auf einem konzeptionellen Entwurf und ersten Skizzen für das HMI Webinterface. Ziel war es, frühzeitig die grundlegende Struktur und Bedienlogik festzulegen, bevor eine konkrete Implementierung erfolgt. Die Skizzen dienen dabei sowohl als visuelle Orientierung für die spätere Entwicklung als auch

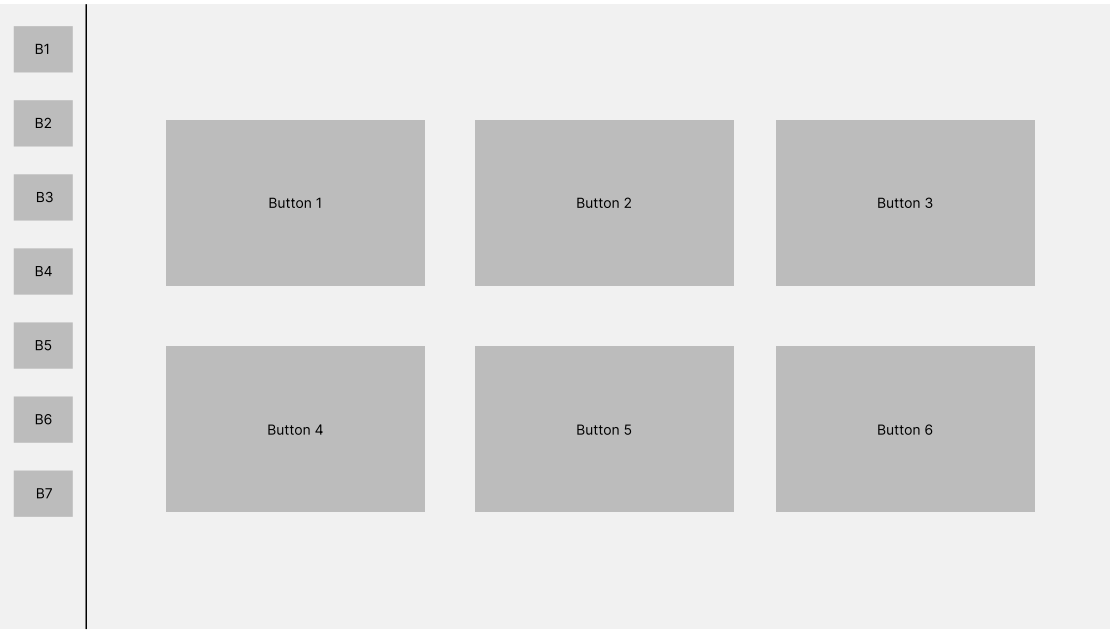


Abbildung 4.2: Wireframe der Startseite der HMI Anwendung

zur Kommunikation mit Stakeholdern über zentrale Ideen und die Umsetzung der funktionalen Anforderungen.

Hauptmenü und Navigation

Abbildung 4.2 zeigt eine Skizzierung der Startseite der Anwendung. Sie soll als ein zentraler Zugangspunkt dienen, von dem aus Nutzer direkt die wichtigsten Funktionsbereiche (z. B. Stationsübersicht und Robotersteuerung) erreichen können. Die zentral dargestellten, rechteckigen Knöpfe dienen als Navigationspunkte, wodurch der Nutzer auf alle Untermenüpunkte zugreifen kann.

Außerdem befindet sich auf der linken Seite der Anwendung eine statische Navigationsleiste, welche es dem Nutzer erlaubt, auch ohne die Rückkehr zur Startseite, alle Menüpunkte zu erreichen. Sowohl die Knöpfe auf der Startseite, als auch die auf der Navigationsleiste leiten immer auf die jeweils selben Untermenüs.

Stationsübersicht

Abbildung 4.3 zeigt die Darstellungsoberfläche aller verfügbaren Versuchseinheiten (MICRO-Cubes, vgl. Unterunterabschnitt 2.2.2). Innerhalb der Cube-Buttons sollen Sta-

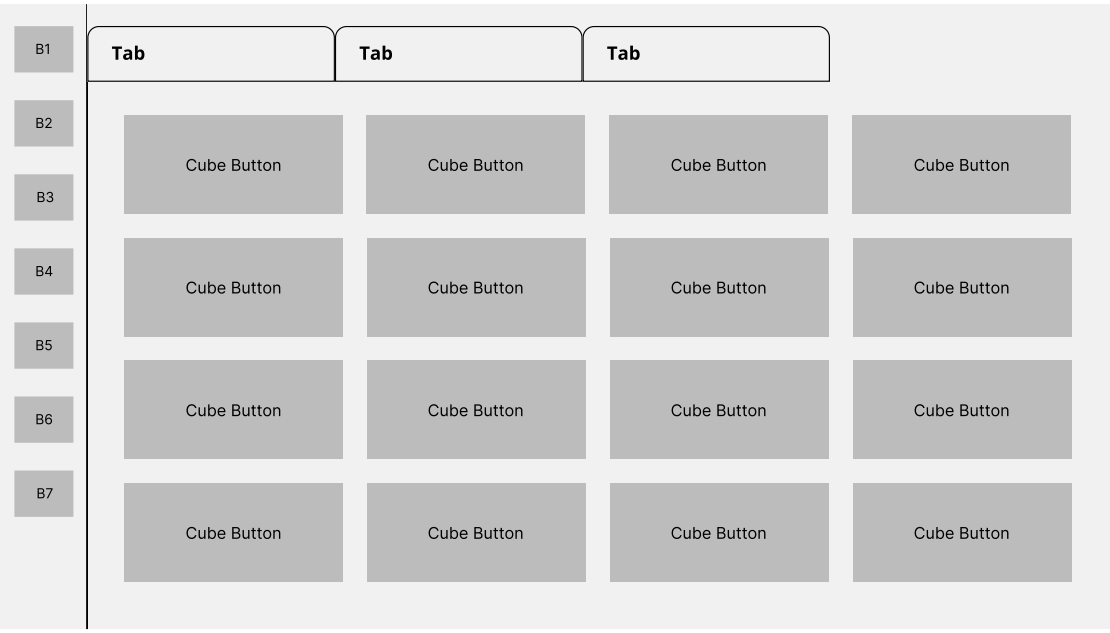


Abbildung 4.3: Wireframe der Stationsübersicht der HMI Anwendung

tusinformationen, wie die aktuelle Nutzung durch einen MICRO-Nutzer oder die generelle Sichtbarkeit für MICRO-Nutzer, angezeigt werden.

Zudem soll eine Tableiste oberhalb der Stationsübersicht die Möglichkeit offen lassen, mehrere MACRO-Aufbauten getrennt verwalten zu können. Dadurch soll sich auch die Möglichkeit ergeben, Entwicklungsstationen klar von Produktionsstationen zu trennen und separat betrachten zu können.

Detailansicht einer Station

Abbildung 4.4 zeigt, wie MICRO-Administratoren weiterführende Informationen (z. B. Sichtbarkeit, letzte Aktivitäten, Reservierungsstatus) zu einer bestimmten Station erhalten können und Verwaltungsfunktionen, wie z. B. das Verbergen einer Station vor MICRO-Nutzern, nutzen können.

Zudem soll diese Ansicht die Möglichkeit bieten, Nutzermeldungen einer Station einsehen zu können (F-R2). Diese Funktionalität soll über die dargestellte Tableiste implementiert werden, welche den Inhalt der Seite durch eine Liste der Nutzermeldungen austauschen soll.

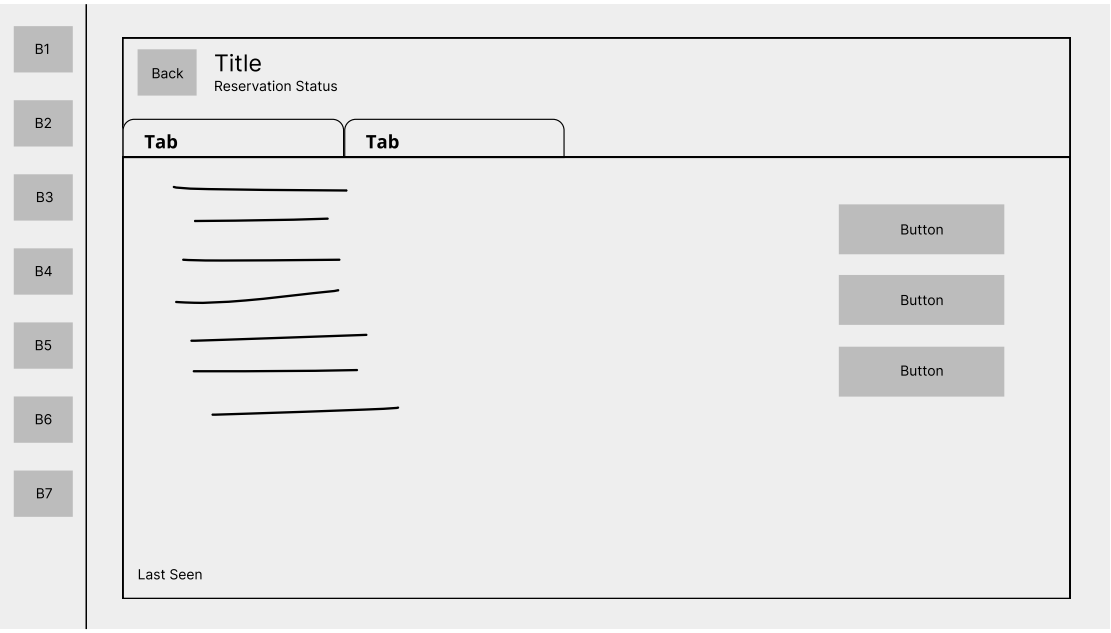


Abbildung 4.4: Wireframe der Detailansicht einer Station der HMI Anwendung

Portalroboter-Steuerung

Abbildung 4.5 zeigt die Robotersteuerung der HMI-Anwendung. Diese fokussiert sich auf eine minimalistische Oberfläche zur präzisen Eingabe der Steuerkoordinaten (z. B. x-/y-Position). Ziel ist es, die Bedienung möglichst einfach zu gestalten. Das Design orientiert sich an einem Controller-ähnlichen Layout, und versucht so eine möglichst intuitive Bedienung zu ermöglichen.

Da der Portalroboter zum Stand der Entwicklung der HMI-Anwendung noch nicht in Betrieb war, wurde die Robotersteuerung nur konzeptionell angefertigt und implementiert und bedarf einer späteren Überarbeitung, sobald der Roboter funktional in Betrieb genommen werden kann und genauere Vorgaben festgelegt wurden.

RGB-LED Steuerung

Die Benutzeroberfläche des RGB-LED-Controllers ist auf eine möglichst einfache und intuitive Steuerung ausgelegt und in Abbildung 4.6 dargestellt. Kern des Interfaces ist ein Color-Picker, über den Nutzerinnen und Nutzer direkt eine beliebige Farbe auswählen können.

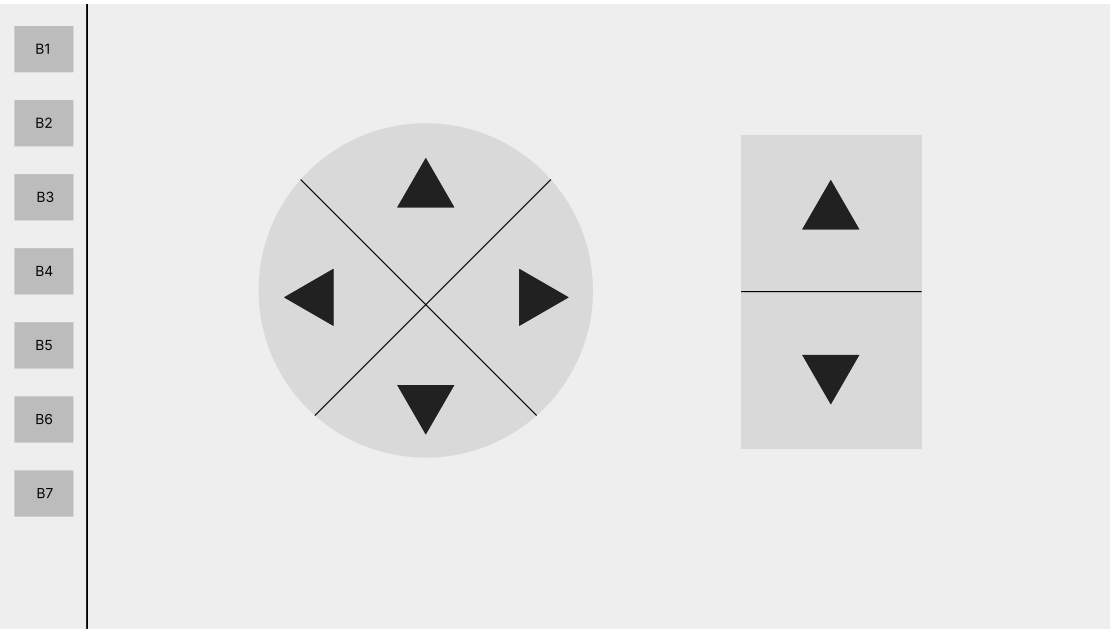


Abbildung 4.5: Wireframe der Robotersteuerung für den Portalroboter

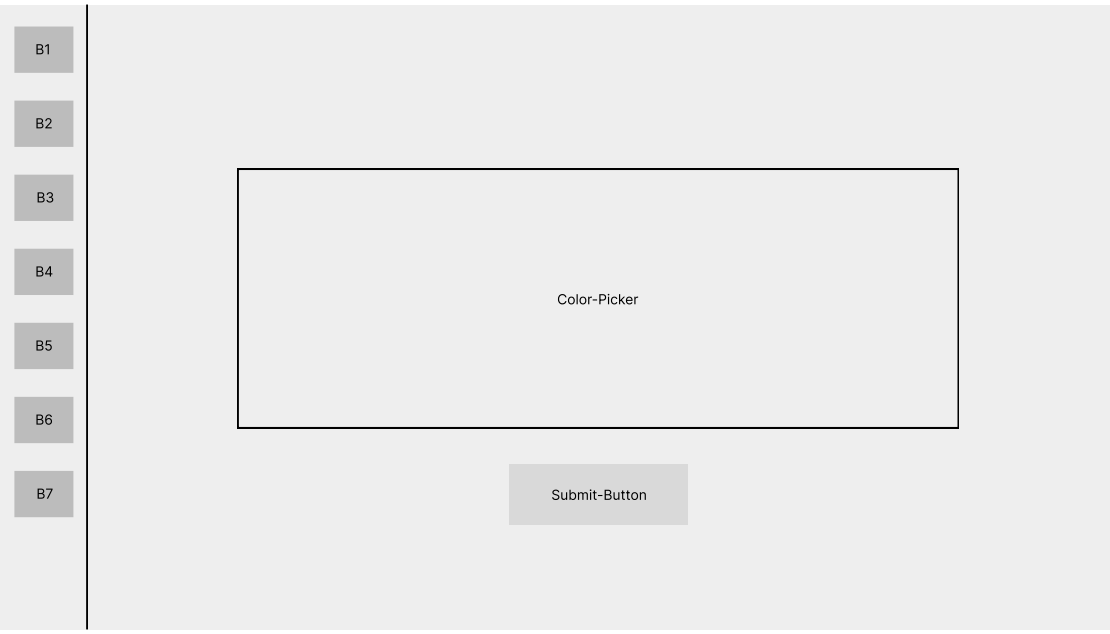


Abbildung 4.6: Wireframe der RGB-LED Steuerung

Um sicherzustellen, dass Einstellungen nicht unbeabsichtigt übernommen werden, soll die Aktivierung der Auswahl erst durch Betätigung des Submit-Buttons erfolgen.

Das Interaktionsprinzip ist damit zweistufig: Zunächst wird eine Farbe im Color-Picker bestimmt, anschließend wird diese bewusst bestätigt. Dieses Vorgehen soll Fehlbedienungen reduziert, Datenverkehr minimieren und gibt den Nutzenden die Möglichkeit, vor der Übernahme verschiedene Optionen auszuprobieren.

Die RGB-LEDs sind ebenfalls in der Lage die Farben iterativ zu verändern. Somit existiert die Möglichkeit Animationen über diese darstellen zu können. Sollte dieser Anwendungsfall gewünscht sein, muss eine Animationsauswahl innerhalb der Steuerung implementiert werden. Möglich wäre ein Dropdown-Menü.

Mit dieser Konzeption soll eine Steuerung realisiert werden, die sowohl Flexibilität bei der Farbauswahl bietet als auch eine kontrollierte und nachvollziehbare Übernahme der Einstellungen sicherstellt.

Ansicht der Nutzermeldungen

Die Nutzermeldungsansicht (oder auch Report-Ansicht) dient der Übersicht und Verwaltung aller eingegangenen Meldungen von Nutzerinnen und Nutzern. Sie soll als tabellarische Darstellung aufgebaut werden und es ermöglichen, Meldungen zu sichten und gezielt zu bearbeiten.

In Abbildung 4.7 findet sich eine konzeptionelle Darstellung der zu entwickelnden Nutzermeldungsansicht. Die Tabelle zeigt in jeder Zeile eine Nutzermeldung und soll in der Spalte ganz rechts Platz lassen für Knöpfe, über welche man mit der Meldung interagieren kann. Das Design orientiert sich hierbei an bestehenden Tabellenlayouts, die im Adminbereich des MICRO-Frontends zu finden sind und soll so vertraut und einheitlich mit dem restlichen System wirken (N-R4).

Die einzelnen Spalten der Tabelle müssen im Verlauf der Realisierung festgelegt werden, da dafür Informationen aus dem bestehenden Report-System von MICRO nötig sind.

Selbige Tabellenstruktur soll ebenfalls für die Report-Ansicht der Detailansicht von Stationen (vgl. Unterunterabschnitt 4.3.3) verwendet werden.

Mit diesem Aufbau wird eine schnelle Orientierung ermöglicht: Alle Meldungen sind zentral einsehbar, und die wichtigsten Verwaltungsaktionen können unmittelbar aus der Übersicht heraus ausgeführt werden.

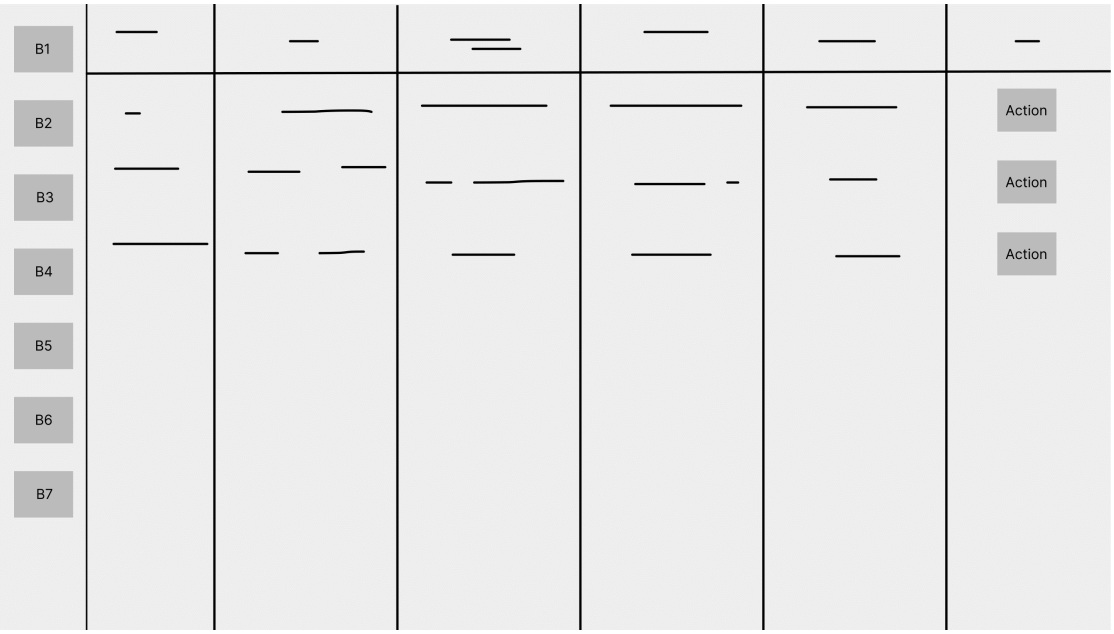


Abbildung 4.7: Wireframe der Ansicht der Nutzermeldungen

Fazit zur Interface-Gestaltung

Das vorgestellte Interface ist als konzeptionell zu verstehen: Die Skizzen konkretisieren die grundlegenden Ansätze, definieren die sichtbaren Funktionsbereiche und unterstützen die iterative Weiterentwicklung des Designs. Durch diese visuelle Konzeptarbeit wird eine solide Basis für die spätere technische Umsetzung (Kapitel 5) geschaffen.

4.3.4 Touch-Design-Herausforderungen

Touchscreens sind aus modernen Benutzerschnittstellen nicht mehr wegzudenken, insbesondere im industriellen Kontext (vgl. Unterabschnitt 2.3.3) oder in eingebetteten Systemen, in denen physische Eingabegeräte wie Tastaturen oder Mäuse aus Platz-, Hygiene- oder Bedienbarkeitsgründen häufig entfallen. Die direkte, haptische Interaktion bringt jedoch spezifische Herausforderungen mit sich, die bei der Gestaltung von HMIs von Beginn an berücksichtigt werden sollten.

In diesem Abschnitt werden zwei typische Probleme bei der Entwicklung von Touch-Interfaces beschrieben: das *Midas-Touch-Problem* sowie das *Fat-Finger-Problem*. Beide betreffen die Präzision und Zuverlässigkeit von Eingaben und können zu Frustration und erhöhter Fehleranfälligkeit führen. Im Folgenden werden Ansätze vorgestellt, wie

diesen Herausforderungen im Rahmen des geplanten Interface-Designs begegnet werden kann.

Midas-Touch-Problem

Das sogenannte Midas-Touch-Problem beschreibt die Schwierigkeit, unbeabsichtigte Eingaben zu vermeiden, die bei der Berührung eines Touchscreens entstehen können [44]. Besonders bei großflächigen Touch-Displays besteht die Gefahr, dass Berührungen, die lediglich der Navigation, dem Abstützen der Hand oder dem Scrollen dienen, fälschlicherweise als aktive Eingabe interpretiert werden.

Zur Reduktion dieses Problems könnten unter anderem folgende Gestaltungsprinzipien eingesetzt werden:

- *Verzögerte Aktivierung kritischer Bedienelemente*: Schaltflächen, die wichtige oder irreversible Funktionen auslösen, könnten mit einer Bestätigungsabfrage versehen werden, um unbeabsichtigte Aktionen zu verhindern.
- *Nutzung geeigneter UI-Komponenten*: Komponentenbibliotheken wie Vuetify [128] bieten bereits Mechanismen, bei denen eine Aktion erst nach dem Loslassen der Berührung ausgelöst wird. Dies könnte genutzt werden, um Eingaben abzubrechen, wenn der Finger während des Berührens vom Bedienelement wegbewegt wird.
- *Visuelles Feedback*: Jede Interaktion könnte unmittelbar ein visuelles Signal (z. B. Farbwechsel oder Animation) geben, um dem Nutzer die Erfassung der Eingabe zu bestätigen.

Fat-Finger-Problem

Das Fat-Finger-Problem beschreibt die Schwierigkeit, Eingaben auf Touch-Geräten zu tätigen, wenn die Größe der Finger im Verhältnis zu den Bedienelementen zu Fehleingaben führt [2]. Besonders bei kleinen Buttons oder dicht platzierten Bedienelementen steigt die Wahrscheinlichkeit, versehentlich ein falsches Element zu aktivieren.

Folgende Gestaltungsentscheidungen können helfen, dieses Problem zu minimieren:

- *Verwendung optimierter UI-Komponenten*: Standardisierte Komponenten wie v-btn, v-card und v-list aus Vuetify bieten vordefinierte Größen und Abstände, die für Touch-Bedienung optimiert sind [127]. Diese könnten konsistent eingesetzt und bei Bedarf durch angepasste Abstände erweitert werden.
- *Reduktion der Informationsdichte*: Statt vieler kleiner Schaltflächen könnte die Oberfläche auf wenige, klar erkennbare Interaktionspunkte reduziert werden, um gezielte Eingaben zu erleichtern.
- *Frühes Prototyping und Tests*: Regelmäßige Überprüfung der Entwürfe auf dem Zielgerät würde sicherstellen, dass die Bedienelemente in der realen Nutzungssituation ausreichend groß und gut erreichbar sind.

Die Berücksichtigung dieser Aspekte bereits in der Konzeptionsphase kann maßgeblich dazu beitragen, die spätere Bedienbarkeit zu verbessern und Fehlerquoten im produktiven Einsatz zu reduzieren.

5 Realisierung

In diesem Kapitel wird die konkrete Umsetzung des im Rahmen dieser Arbeit entwickelten HMIs beschrieben. Der Fokus liegt dabei auf der technischen Realisierung der Hard- und Softwarekomponenten, der Gestaltung der Benutzeroberfläche sowie der Integration in die bestehende MICRO-Infrastruktur. Ziel war es, ein funktionales, erweiterbares und wartbares System zu schaffen.

Die Umsetzung gliedert sich in mehrere aufeinander abgestimmte Teilbereiche: Zunächst wird die Auswahl der benötigten Technologien zur Entwicklung des HMIs getroffen, gefolgt von der Einrichtung der Raspberry-Pi-Umgebung, welche die Hardwarebasis für die HMI-Clients bildet. Daran anschließend folgt die detaillierte Darstellung der grafischen Benutzeroberfläche, die als zentrale Schnittstelle zur Interaktion mit dem MICRO-System dient. Im nächsten Abschnitt werden die notwendigen Schnittstellen zu externen Systemen, sowie der Umgang mit persistenter Datenspeicherung und Authentifizierung erläutert. Abschließend wird der Aufbau der automatisierten Build- und Deployment-Pipeline dargestellt, die eine kontinuierliche Auslieferung und Aktualisierung des Systems gewährleistet.

Durch die modulare Struktur der Umsetzung konnte ein hohes Maß an Flexibilität erreicht werden, das zukünftige Erweiterungen oder Anpassungen mit minimalem Aufwand ermöglicht.

5.1 Auswahl von Technologien

Die Auswahl geeigneter Technologien bildet das Fundament für die Entwicklung eines Touch-basierten Human-Machine-Interfaces (HMIs). In diesem Kapitel werden die Entscheidungsprozesse und Bewertungskriterien für alle wesentlichen Komponenten dargelegt. Neben technischen Erwägungen wie Leistung, Stabilität und Kompatibilität werden dabei auch projektbezogene Anforderungen wie Wartbarkeit (N-R9), Portierbarkeit (N-R3) und Nutzerfreundlichkeit (N-R6) berücksichtigt. Durch eine Evaluation

alternativer Lösungen und die Nutzung wissenschaftlich fundierter Quellen wird eine Begründung für die getroffenen Technologieentscheidungen geliefert.

5.1.1 Hardwarekomponenten

Für die Realisierung des HMIs wurden spezifische Hardwarekomponenten ausgewählt, die sowohl die Rechenleistung als auch die Benutzerinteraktion sicherstellen. Dabei fiel die Wahl auf einen Raspberry Pi als zentrale Recheneinheit sowie verschiedene Touch-Displays, die hinsichtlich ihrer Eignung evaluiert wurden.

Auswahl einer Einplatinenlösung

Bei der Auswahl der eingesetzten Hardware standen Stabilität, Verfügbarkeit und Zukunftsfähigkeit im Fokus. Als Hardwareplattform wurde der Raspberry Pi (siehe Unterabschnitt 3.1.2) gewählt, da dieser kostengünstig, energieeffizient und in der Community gut dokumentiert ist [122]. Die Anbindung von Displays sowie Peripherie über GPIO, I^2C oder USB ist damit unkompliziert möglich [122]. Zudem wurde die Verwendung eines Raspberry Pis als Anforderung vorgeschrieben (N-R1).

Alternativ zu einem Raspberry Pi hätten zum Beispiel (android) Tablets verwendet werden können, da sie Touch und Display bereits mitbringen und eine schlanke, robuste Plattform bieten. Sie eignen sich insbesondere, wenn keine besondere Nutzung von Low-Level-Peripherie (wie GPIO, I^2C, SPI) notwendig ist und das HMI rein web- oder appbasiert arbeitet.

Auch möglich wäre die Nutzung eines Steam Decks [124]. Dieses wird zwar primär als Spieleplattform vermarktet, das verwendete Betriebssystem „SteamOS“ basiert jedoch auf einem Arch Linux [125], welches auch wie gewohnt in einem regulären Desktop-Modus verwendet werden kann [124]. Dadurch eröffnet sich die Möglichkeit, die Entwicklung direkt auf diesem Gerät durchzuführen, wobei die integrierten Analog-Sticks eine vielversprechende Option zur Umsetzung der Portalrobotersteuerung darstellen. Im Vergleich zu einem Raspberry Pi ist das Steam Deck mit einem Preis von mindestens 419€ (Stand: 22.08.2025) jedoch deutlich teurer [123].

Auswahl geeigneter Touch-Displays

Die Wahl des zu verwendeten Displays erfolgt durch eine Evaluation, auf welche in Kapitel 6 weiter eingegangen wird. Tabelle 5.1 zeigt eine Auflistung der getesteten Displays und ihre für diese Auswertung relevanten Spezifikationen. Die Displays wurden so gewählt, dass sie möglichst viele Aspekte abdecken. So wurden beispielsweise Displays mit hochwertigen Panels (QLED, AMOLED) getestet, aber auch Displays verschiedenster Größen und Formaten.

Tabelle 5.1: Auflistung getesteter Touch-Displays für den Raspberry Pi (Stand: 27.07.2025)

ID	Display	Größe	Auflösung	Panel	Preis
D1	Waveshare 15.6 inch QLED Display [72]	15.6 Zoll	1920×1080	QLED	199,99$
D2	Waveshare 5.5 inch AMOLED [73]	5.5 Zoll	1080×1920	AMOLED	121,99$
D3	Waveshare 9.3 inch LCD Display [75]	9.3 Zoll	1600×600	IPS	99,99$
D4	Raspberry Pi Touch Display 2 [77]	7 Zoll	720×1280	TFT	60$
D5	Waveshare 15.6 inch LCD Display [71]	15.6 Zoll	1920×1080	IPS	166,99$
D6	Waveshare 7 inch Pi Zero Display [74]	7 Zoll	1024×600	IPS	52,99$

Bei der Auswertung der Displays zeigten sich neben den technischen Spezifikationen bereits bei erster Verwendung qualitative Unterschiede in der Verarbeitung und Praxistauglichkeit:

Das D2 bietet zwar ein hochauflösendes AMOLED-Panel, jedoch ist die physische Stabilität unzureichend. Bereits leichter Druck auf die Ränder führte während der Einrichtung zu sichtbaren Schäden am Display, welche dieses unbrauchbar machen. Bei der Verwendung dieses Displays ohne Befestigung ist somit Vorsicht geboten.

Beim Modell D3 handelt es sich um ein ungewöhnlich breites Ultrabreitbild-Panel im Format 1600×600, welches eine interessante Alternative zu herkömmlichen Bildformaten bietet.

Das Display D4 überzeugte durch seine robuste Metallbackplate, die das Panel mechanisch gut schützt und es damit für einen längeren Betrieb prädestiniert.

Besonders problematisch war das Modell D6 für den Anwendungszweck als HMI. Dieses Display nutzt statt eines normalen Raspberry Pi, ein Raspberry Pi Zero [77], welcher deutlich niedrigere Leistung bietet als ein Raspberry Pi 5 [77, 78]. Dieses konnte aufgrund der stark limitierten RAM- und Rechenressourcen keine stabil laufenden Docker-Container oder Browser darstellen und fiel damit aus der engeren Auswahl.

5.1.2 Betriebssystem

Für die Entwicklung des Touch-basierten HMIs auf dem Raspberry Pi wurden verschiedene Linux-basierte Betriebssysteme hinsichtlich ihrer Eignung evaluiert. Dabei standen insbesondere Kriterien wie Systemstabilität, Ressourcennutzung und die Kompatibilität (N-R9) mit den verwendeten Touch-Displays, sowie die Unterstützung grundlegender Touch-Funktionalitäten, insbesondere Drag-Scrolling, im Fokus.

Die Auswahl der Betriebssysteme wurde anhand der Dokumentation von Waveshare vorgenommen, da die Displays laut Hersteller nur mit einigen Betriebssystemen kompatibel seien [72]. Somit wurden *Raspberry Pi OS (Desktop und Lite)*, *Kali Linux* und *Ubuntu Desktop* zum Testen ausgewählt. Zusätzlich wurden einige weitere Ubuntu Derivate (*Xubuntu, Lubuntu, Ubuntu Core*) und weitere Betriebssysteme (*Manjaro*) getestet, um einerseits ein Betriebssystem zu finden, welches möglicherweise eine bessere Nutzererfahrung bietet, und andererseits, um die Aussagen des Herstellers zu überprüfen.

Die Gegenüberstellung verschiedener Betriebssysteme aus Tabelle 5.2 für den Einsatz im geplanten HMI-System zeigt deutliche Unterschiede in der Nutzbarkeit der Betriebssysteme für das HMI-System. Während **Raspberry Pi OS** eine stabile und hardwarekompatible Grundlage bot, scheiterte es wie auch **Kali Linux** an der fehlenden Unterstützung für Drag-Scrolling, was eine flüssige Touch-Bedienung verhinderte. **Ubuntu Desktop** überzeugte durch eine große Softwarebasis, erwies sich jedoch als zu ressourcenintensiv, wodurch eine verzögerte und träge Bedienung entstand. Leichtgewichtige Varianten wie **Xubuntu** und **Lubuntu** versprechen zwar eine bessere Performance, waren jedoch aufgrund von Darstellungsproblemen auf Waveshare-Touchdisplays unbrauchbar und konnten gar nicht grafisch dargestellt werden. Gleiches galt für **Ubuntu Core** und **Manjaro Linux**, bei denen die grafische Oberfläche gar nicht angezeigt wurde.

Tabelle 5.2: Vergleich der getesteten Betriebssysteme bezüglich Touch-HMI-Anforderungen am Raspberry Pi

Betriebssystem	**Vorteile**	**Probleme / Nachteile**
Raspberry Pi OS (Version vom 19.11.2024) [79]	Stabil, gute Hardware-Kompatibilität	Kein natives Drag-Scrolling, Touch als Mauszeiger, erfüllt N-R2 nicht
Raspberry Pi OS Lite [79] + Weston [21]	Ressourcenschonend, stabil, vollständige Touch-Gesten-Unterstützung (inkl. Drag-Scrolling), leichtgewichtig, Kiosk-Modus möglich	-
Kali Linux (2025.1) [70]	Unterstützt grundlegend Touch-Eingaben	Kein natives Drag-Scrolling, Touch als Mauszeiger, erfüllt N-R2 nicht
Ubuntu Desktop (24.04.2) [80]	Umfangreiche Softwarebasis	Sehr ressourcenintensiv, verzögerte UI-Reaktion, für Nutzung im Kontext ausgeschlossen (N-R8)
Xubuntu [83]	Leichtgewichtig	Grafische Oberfläche wird auf Waveshare-Displays nicht dargestellt
Lubuntu [81]	Leichtgewichtig	Grafische Oberfläche wird auf Waveshare-Displays nicht dargestellt
Ubuntu Core [82]	-	Grafische Oberfläche wird auf Waveshare-Displays nicht dargestellt
Manjaro Linux [20]	-	Grafische Oberfläche wird auf Waveshare-Displays nicht dargestellt

Als optimale Lösung erwies sich schließlich die Kombination aus **Raspberry Pi OS Lite** und dem **Wayland-Compositor Weston**. Dieses Setup ist nicht nur ressourcenschonend und stabil, sondern bietet auch volle Unterstützung für Touch-Gesten inklusive Drag-Scrolling. Zudem ermöglicht Weston den Betrieb im Kiosk-Modus, was perfekt für ein fest installiertes HMI-System ist. Damit erfüllte diese Konfiguration als einzige alle Anforderungen hinsichtlich Benutzerfreundlichkeit, Performance und Hardwarekompatibilität und wurde folglich als Zielplattform ausgewählt.

5.1.3 HMI-Software

Für die Entwicklung eines HMIs auf einem Raspberry Pi existieren verschiedene Ansätze und Technologien. Ziel war es, eine möglichst wartbare (N-R9) und leicht erweiterbare Lösung zu finden (N-R10), die gleichzeitig eine ansprechende Benutzeroberfläche für ein Touch-Display bietet und die Nutzung von Touch-Gesten zulässt (N-R2). Zudem soll die Software möglichst plattformübergreifend entwickelt werden, um eine Portierbarkeit auf andere Geräte zu ermöglichen (N-R3). Im Folgenden werden einige relevante Technologien vorgestellt und im Hinblick auf ihre Eignung für das Projekt bewertet.

Qt

Qt [22] ist ein leistungsfähiges C++-Framework zur Erstellung plattformübergreifender Benutzeroberflächen. Es bietet umfangreiche GUI-Komponenten und ist für performante, native Anwendungen geeignet [119]. Außerdem ist es auch in der Wirtschaft weit verbreitet und wird beispielsweise von Mercedes-Benz, ABB oder LG für Softwarelösungen verwendet [22].

GTK

Das GIMP-Toolkit (GTK) ist ein weiteres etabliertes Toolkit, das insbesondere im Linux-Umfeld verbreitet ist (z. B. bei GNOME-Anwendungen) [102]. Es ist primär in C++ geschrieben, erlaubt aber auch die Nutzung mit anderen Sprachen wie Python, JavaScript oder Rust [120].

Electron

Electron kombiniert Webtechnologien mit einer Node.js-Umgebung und baut auf den Chromium-Browser auf [23]. Electron läuft nativ auf allen bekannten Betriebssystemen [23] und erlaubt die Entwicklung von Desktop-Apps mit HTML, CSS und JavaScript [23].

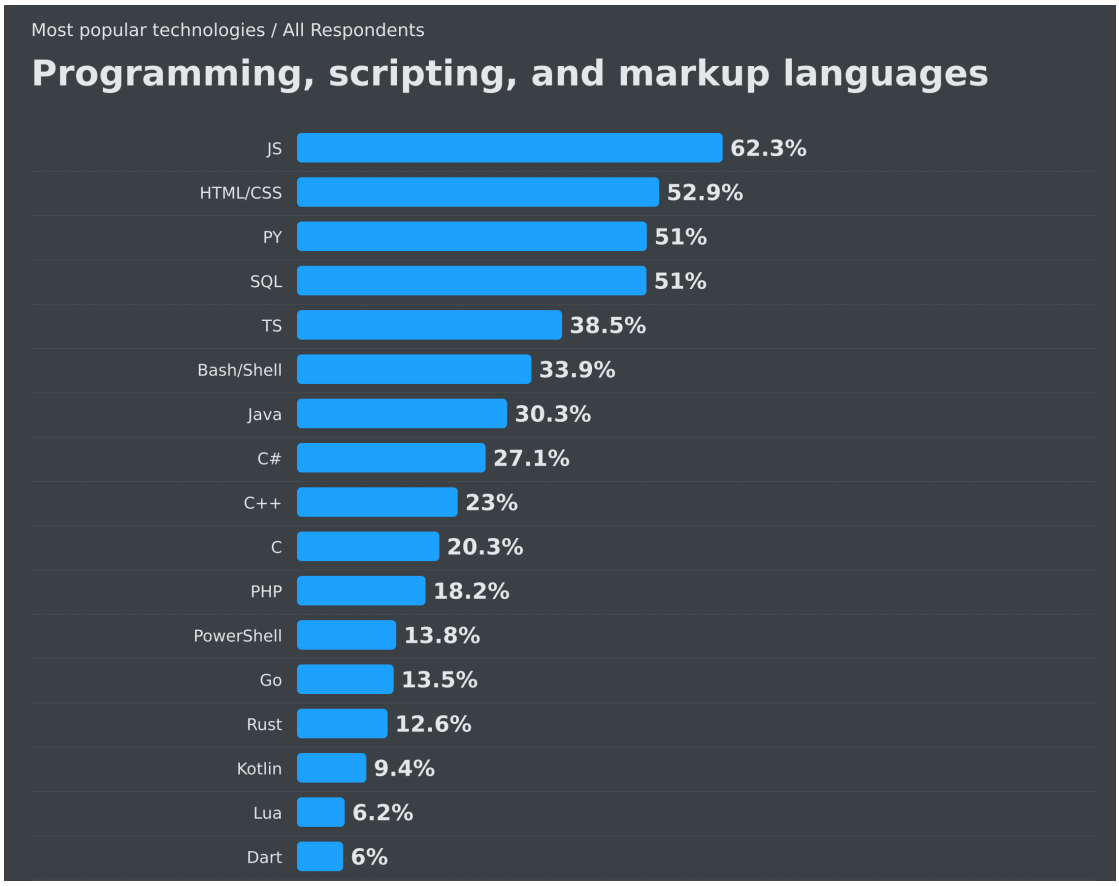


Abbildung 5.1: Vergleich der Verwendung und Beliebtheit von Programmiersprachen [64]

Flutter

Flutter [52] ist ein von Google entwickeltes Open-Source-Framework zur plattformübergreifenden Entwicklung von Benutzeroberflächen. Es verwendet die Programmiersprache Dart und ermöglicht die Erstellung nativer Anwendungen für mobile Geräte, Desktops und Webbrowser mit einer einzigen Codebasis [52, 54].

Webframeworks im Browser

Da Electron grundsätzlich eine attraktive Lösung zur Entwicklung plattformübergreifender Anwendungen bietet, jedoch aufgrund seines Ressourcenverbrauchs für den Raspberry Pi weniger geeignet ist, wurde als ressourcenschonendere Alternative die Nutzung eines reinen Browsers im Kiosk-Modus in Betracht gezogen. Dabei wird die

Webanwendung lokal auf dem Raspberry Pi ausgeführt und direkt im Browser (z. B. Chromium) gestartet.

Auswahl der zu verwendeten Softwarelösung

Tabelle 5.3 zeigt eine Gegenüberstellung der Vor- und Nachteile der genannten möglichen Softwarelösungen der HMI-Software.

Aufgrund der Lizenzbeschränkungen wurde **Qt** als nicht ideal für die Entwicklung der HMI Software empfunden, da der Programmcode des HMIs vorerst nicht publik gemacht werden soll. Qt bleibt aber generell eine interessante Alternative für andere Projekte, vor allem da es viel Anwendung in der Wirtschaft findet [22].

Wegen der Einschränkung, dass **GTK**-Applikationen nicht auf mobile Geräte (z. B. auf Android- oder iOS-Geräte) portiert werden können, erweist sich GTK ebenfalls nicht als optimal, da es der Anforderung N-R3 widerspricht.

Insgesamt eignet sich **Electron** besonders gut für die plattformübergreifende Entwicklung moderner Desktop-Anwendungen, da es auf etablierten Webtechnologien basiert. Für ressourcenbeschränkte Umgebungen wie den Raspberry Pi ist der verursachte Overhead durch die Erweiterung von Chromium jedoch problematisch, sodass alternative Lösungen mit geringerem Ressourcenverbrauch vorzuziehen sind.

Flutter bietet insgesamt eine moderne und leistungsfähige Plattform zur Entwicklung interaktiver Benutzeroberflächen, die sich besonders durch ihre plattformübergreifende Nutzbarkeit und native Touch-Unterstützung auszeichnet. Für den Einsatz im vorliegenden Projekt ist es jedoch aufgrund der geringeren Verbreitung von Dart (nur 6% befragter Entwickler haben schon einmal mit Dart gearbeitet, vgl. Abbildung 5.1) und der fehlenden Integration in die bestehende technologische Basis weniger vorteilhaft, insbesondere im Hinblick auf die Wartbarkeit gemäß Anforderung N-R9. Auch mögliche Performance-Einschränkungen auf dem Raspberry Pi könnten die Eignung zusätzlich einschränken.

Die Ausführung der Anwendung im **Browser mit einem Webframework** stellt eine pragmatische und ressourcenschonende Lösung dar, die insbesondere für den Einsatz auf leistungsschwächerer Hardware wie dem Raspberry Pi geeignet ist. Trotz gewisser Einschränkungen beim Hardwarezugriff überwiegen die Vorteile in Bezug auf Portierbarkeit, Entwicklungsaufwand und technologische Anschlussfähigkeit innerhalb des

Tabelle 5.3: Vergleich der untersuchten Frameworks

Framework	Vorteile	Nachteile
Qt	• Hohe Performance • Native Hardwareanbindung • Große Community, gute Dokumentation	• Höhere Einarbeitung in C++ • Lizenzprobleme bei nicht-öffentlichen Projekten (GPL, LGPL) [22]
GTK	• Gute Linux-Integration • Open-Source, lizenzfreundlich • Portierbar auf viele Betriebssysteme [120]	• Keine Portierung auf mobile Geräte [10]
Electron	• Große Auswahl an Webtools • Verbreitete Webentwicklung (vgl. Abbildung 5.2) • Schnelle UI-Entwicklung • Gute OS-Integration	• Hoher Overhead durch Aufbau auf Chromium [23] • Schlecht geeignet für leistungsschwache Hardware wie Raspberry Pi (widerspricht N-R8)
Flutter	• Einheitlicher Code für mehrere Plattformen [52] • Große Community, gute Dokumentation • Native Touch-Unterstützung	• Dart wenig verbreitet (vgl. Abbildung 5.1), erhöht Einarbeitung; widerspricht N-R9 • Eingeschränkte Performance auf Raspberry Pi
Webframeworks im Browser	• Plattformunabhängig und leicht portierbar (N-R3) • Große Auswahl moderner Frameworks • Geringe Einarbeitung durch vorhandene Nutzung im Projekt (N-R9) • Browser bieten Touch-Gesten [49]	• Eingeschränkter Low-Level-Hardwarezugriff [51]

Projekts. Zudem bleibt die Möglichkeit offen, das System im späteren Verlauf auf Electron zu portieren, da beides auf Web-Technologien basiert. Somit erfüllt diese Lösung

alle Anforderungen.

Nach der Bewertung der genannten Technologien fiel die Entscheidung zugunsten einer Anwendung implementiert durch ein **Webframework im Kiosk-Modus in einem Browser**. Diese Lösung bietet die größte Flexibilität bei gleichzeitig einfacher Handhabung. Besonders vorteilhaft ist die weite Verbreitung von Webtechnologien. Etwa 40% der Befragten Entwickler aus allen Bereichen gaben bei einer Umfrage von Stack Overflow an, bereits einmal mit React [85] gearbeitet zu haben (Abbildung 5.2). Viele Entwickler wissen mit Webtechnologien umzugehen, vor allem wird in MICRO bereits Vue.js benutzt und bietet somit eine niedrigere Einstiegshürde. Zudem setzen viele der alternativen HMI-Lösungen aus Abschnitt 2.3 ebenfalls auf webbasierten Softwarelösungen, was die Praktikabilität dieser Lösung weiter bestärkt.

Auch in Hinblick auf zukünftige Erweiterungen, beispielsweise für größere Displays, zusätzliche Stationen oder die Integration neuer Funktionen, bietet ein webbasierter Ansatz die beste Skalierbarkeit. Da viele Systeme, auch außerhalb des Raspberry-Pi-Umfelds, einen Browser zur Verfügung stellen, ist diese Architektur besonders portabel.

Des Weiteren lässt diese Form der Entwicklung die Möglichkeit offen, die Software im späteren Verlauf auf Electron umzubauen, da beide Ansätze auf Web-Technologien setzen.

Somit werden alle Anforderungen aus Abschnitt 4.1 durch die gewählte Technologie erfüllt.

5.1.4 Vergleich von Webframeworks

Die Entwicklung moderner Webapplikationen, insbesondere für HMIs in industriellen Umgebungen, erfordert die sorgfältige Auswahl geeigneter Frontend-Frameworks. JavaScript-Frameworks haben sich als unverzichtbare Werkzeuge für die Entwicklung komplexer, interaktiver Benutzeroberflächen etabliert, da sie strukturiertere, wartbarere und effizientere Entwicklungsprozesse ermöglichen [19]. Drei Frameworks haben sich dabei als führende Technologien herauskristallisiert: Angular, React und Vue.js (vgl. Abbildung 5.2) [19]. Diese drei Frameworks dominieren den Markt der Frontend-Entwicklung und werden in unterschiedlichen Anwendungsbereichen eingesetzt, von einfachen Webseiten bis hin zu komplexen Single-Page-Applications (SPA) für industrielle Steuerungssysteme [65].

Grundlegende Architekturansätze und Entwicklungsphilosophien

Angular [50], ursprünglich von Google entwickelt und erstmals 2010 als AngularJS veröffentlicht, wurde 2016 vollständig überarbeitet und in Angular 2+ (Angular 2 und alle Nachfolgeversionen) umbenannt [108]. Im Folgenden bezieht sich der Begriff „Angular“ ausschließlich auf Versionen in Angular 2+. Angular setzt als Programmiersprache auch TypeScript und implementiert eine component-basierte Architektur [19, 108]. Das bedeutet, dass Applikationen aus mehreren Komponenten aufgebaut werden, welche wiederverwendbar sind. Angular nutzt DOM-Manipulation mit einem eigenen Change Detection-System, das bei jeder Änderung den gesamten DOM durchläuft, um Echtzeitänderungen zu verwirklichen [108].

React JS [85] wurde 2013 von Meta (damals Facebook) veröffentlicht und gilt heutzutage als eines der beliebtesten Webentwicklungs-Framework (vgl. Abbildung 5.2, Abbildung 5.2). React verfolgt einen library-basierten Ansatz mit einer minimalistischen Kernphilosophie [108]. Dabei konzentriert sich React primär auf die Kernfunktionalitäten seiner komponentenbasierten Struktur und überlässt andere Aspekte wie Routing oder State Management externen Bibliotheken [66, 108]. Das Framework nutzt JSX (JavaScript XML), eine syntaktische Erweiterung von JavaScript, die HTML-ähnliche Syntax innerhalb von JavaScript ermöglicht [19, 108]. Reacts Hauptinnovation ist das Virtual-DOM-Konzept, bei dem Änderungen zunächst in einer virtuellen Repräsentation des DOM durchgeführt und dann effizient mit dem realen DOM synchronisiert werden [108]. Dieser Ansatz ermöglicht optimierte Performance durch Minimierung direkter DOM-Manipulationen [108].

Vue.js [129], kreiert von Evan You und 2014 veröffentlicht, wurde als progressive Framework konzipiert, das auf den Grundideen von Angular und React aufbaut [108]. Vue implementiert ein reaktives Datenbindungssystem, das automatische Synchronisation zwischen Datenmodell und Browseransicht ermöglicht [19]. Das Framework nutzt in seiner Rendering-Pipeline ebenfalls ein Virtual-DOM-System, ähnlich wie React [66, 108].

Vue.js zählt zu den am schnellsten an Popularität gewinnenden JavaScript-Frameworks [66, 108]. Abbildung 5.2 zeigt das Ergebnis einer Umfrage auf Stack Overflow [63] über die Verwendung und Beliebtheit von Web-Frameworks. Hierbei wurden Entwickler aller Bereiche befragt, ob sie bereits mit den Technologien gearbeitet haben und ob sie diese weiterhin verwenden würden. Die Grafik verdeutlicht, dass Vue.js inzwischen eine ähnlich hohe Beliebtheit erreicht hat wie etablierte Alternativen wie React [85] oder Angular [50].

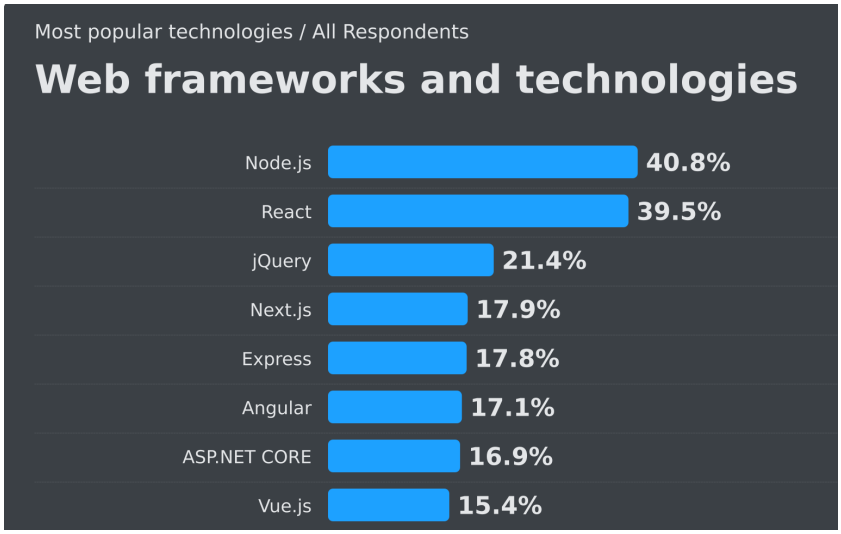


Abbildung 5.2: Vergleich der Beliebtheit und Verwendung von Web-Technologien [64]

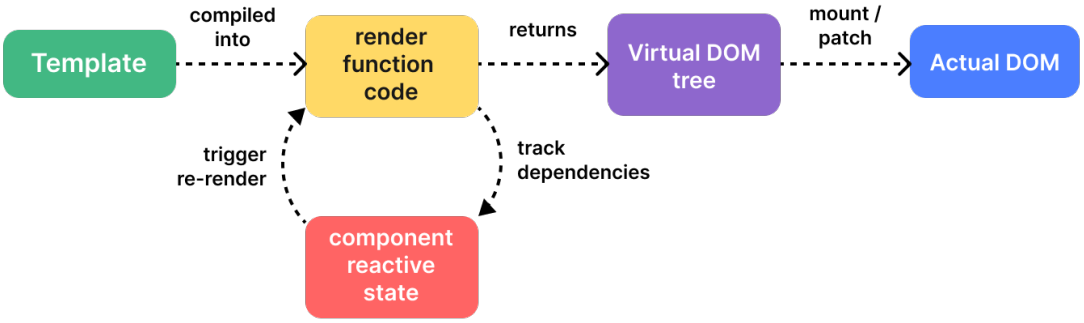


Abbildung 5.3: Visualisierung der Rendering-Pipeline in Vue.js [126]

Abbildung 5.3 stellt die Rendering-Pipeline von Vue.js grafisch dar. Es werden zuerst Template-Beschreibungen in Renderfunktionen übersetzt. Diese Renderfunktionen erzeugen einen sogenannten Virtual-DOM-Baum. Der Virtual-DOM dient als Zwischenschritt und ermöglicht es, Änderungen effizient zu berechnen. Erst nach einem Vergleich des virtuellen DOM mit dem tatsächlichen DOM (sog. „Diffing“) werden die minimal notwendigen Änderungen an die tatsächliche Seite übertragen [126]. Ändert sich der reaktive Zustand einer Komponente, werden die abhängigen Renderfunktionen erneut ausgelöst und der Prozess beginnt von vorne. Dieser Mechanismus erhöht die Performanz und Reaktivität der Anwendung deutlich gegenüber Alternativen, wie direkte DOM-Manipulation [66, 108].

Vue ist zudem besonders leichtgewichtig [108] und zeichnet sich durch eine geringe Lernkurve aus, da es auf Standard-Webtechnologien (HTML, CSS, JavaScript) aufbaut, ohne zusätzliche Sprachen zu erfordern [19].

Die drei vorgestellten Frameworks, Angular, React und Vue, verfolgen unterschiedliche Architekturansätze und Philosophien, basieren jedoch alle auf komponentenbasierter Entwicklung und modernen Webtechnologien. Während Angular einen umfassenden, strikt strukturierten Ansatz mit vollständigem Tooling bietet, verfolgt React eine minimalistische Philosophie mit großer Flexibilität durch externe Bibliotheken. Vue kombiniert Stärken beider Ansätze in einem leichtgewichtigen, einsteigerfreundlichen Framework mit klarer Trennung von Struktur, Logik und Darstellung. Je nach Projektanforderung (z. B. Skalierbarkeit, Performance oder Entwicklerfreundlichkeit) eignet sich eines dieser Frameworks besonders gut. Im Allgemeinen trifft man jedoch mit keinem dieser drei Frameworks eine falsche Wahl für den Großteil der Webentwicklungsprojekte.

Performance-Analyse und Evaluation der Frameworks

Für die Entwicklung der HMI-Software auf einem Raspberry Pi ist vor allem ein performantes Framework vonnöten, da ein Raspberry Pi nur bedingt viele Ressourcen mitbringt. Folglich dazu werden die Frameworks auf ihre Leistung evaluiert und es kommt schließlich zu einem Fazit zum idealen Framework für die Entwicklung der HMI-Software.

Rathinam [108], sowie Cincović und Punt [19], untersuchten in ihren jeweiligen Arbeiten aus den Jahren 2023 und 2020 die Performanz der drei führenden Frontend-Frameworks: Angular, React und Vue. Beide Studien kamen übereinstimmend zu dem

Name Duration for...	vue-v3.5.13	angular-cf-v20.0.1	react-classes-v19.0.0
Implementation notes			
Implementation link	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	27.1 ±0.2 (1.19)	32.9 ±0.2 (1.45)	28.3 ±0.2 (1.25)
replace all rows updating all 1,000 rows. (5 warmup runs).	30.8 ±0.1 (1.21)	39.0 ±0.2 (1.53)	34.4 ±0.2 (1.35)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	13.2 ±0.2 (1.32)	12.4 ±0.4 (1.24)	15.1 ±0.3 (1.51)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	3.3 ±0.1 (1.57)	4.0 ±0.2 (1.90)	4.3 ±0.2 (2.05)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	14.9 ±0.4 (1.18)	15.5 ±0.3 (1.23)	105.5 ±0.8 (8.37)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	13.3 ±0.2 (1.36)	11.9 ±0.1 (1.21)	12.2 ±0.1 (1.24)
create many rows creating 10,000 rows. (5 warmup runs).	284.8 ±0.8 (1.20)	353.9 ±0.9 (1.49)	441.5 ±8.0 (1.86)
append rows to large table appending 1,000 to a table of 1,000 rows. (5 warmup runs).	32.4 ±0.3 (1.21)	38.1 ±0.3 (1.43)	34.1 ±0.3 (1.28)
clear rows clearing a table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	11.8 ±0.3 (1.31)	20.6 ±0.7 (2.29)	18.0 ±0.4 (2.00)
weighted geometric mean of all factors in the table	1.26	1.48	1.56
compare: Green means significantly faster, red significantly slower	compare	compare	compare

Abbildung 5.4: Vergleich der Berechnungsgeschwindigkeiten von Angular, React und Vue in ms [68]

Name	vue-v3.5.13	angular-cf-v20.0.1	react-classes-v19.0.0
ready memory Memory usage after page load.	0.9 (1.80)	1.5 (3.00)	1.2 (2.40)
run memory Memory usage after adding 1,000 rows.	3.9 (2.29)	4.7 (2.76)	4.6 (2.71)
update every 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	3.9 (2.29)	4.8 (2.82)	5.1 (3.00)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	1.2 (2.00)	2.2 (3.67)	1.9 (3.17)
run memory 10k Memory usage after adding 10,000 rows.	28.6 (2.36)	29.1 (2.40)	32.4 (2.68)
geometric mean of all factors in the table	2.14	2.90	2.78

Abbildung 5.5: Vergleich der Speichernutzung von Angular, React und Vue in MB [68]

Ergebnis, dass Vue in nahezu allen rechenintensiven Tests schneller und ressourcenschonender abschnitt als die beiden anderen Frameworks [19, 108]. Besonders deutlich zeigte sich der Unterschied im sogenannten „Swap Rows“-Test, bei dem in einer Datentabelle mit 1000 Einträgen gezielt die Elemente an Position 1 und 998 vertauscht werden, um die Reaktionsgeschwindigkeit des Frameworks auf Datenmanipulationen zu messen [1]. Hier waren Angular und React laut Rathinam nahezu 7-mal langsamer als Vue [108].

Die zugrunde liegenden Messwerte bezieht Rathinam aus einer kontinuierlich gepflegten Vergleichsplattform [68], die verschiedene Frameworks unter identischen Bedingungen gegenüberstellt. Die in Abbildung 5.4 und Abbildung 5.5 dargestellten aktuellen Daten (Stand: 28.07.2025, Chrome Version 138.0.7204.50) zeigen, dass sich Angular seit der Veröffentlichung von Rathinams Studie leistungstechnisch deutlich verbessert hat. Inzwischen liefern Angular und Vue vergleichbare Werte, wobei Vue weiterhin leicht vorne liegt. React hingegen ist in diesem spezifischen Test weiter zurückgefallen und mittlerweile rund 8-mal langsamer als Vue. Abgesehen von diesem Extremwert sind die Unterschiede bei der Ausführungsgeschwindigkeit insgesamt gering, wobei Vue tendenziell die beste Performance bietet.

Ein ähnliches Bild zeigt sich beim Speicherverbrauch. Sowohl die Literatur [19, 108] als auch die aktuellen Vergleichsdaten in Abbildung 5.5 belegen, dass Vue den geringsten Speicherbedarf aufweist.

Neben der technischen Leistungsfähigkeit bietet Vue noch weitere Vorteile. Es gilt als besonders einsteigerfreundlich und weist im Vergleich die geringste Lernkurve auf [19]. Darüber hinaus ist Vue durch seinen geringen Overhead, effiziente Rendering-Methoden und seine flexible Architektur ideal geeignet für die Entwicklung sowohl von Single-Page-Applications (SPAs) als auch Multi-Page-Applications (MPAs) [66]. Ein zusätzlicher Vorteil im vorliegenden Projektkontext ist, dass Vue bereits an anderer Stelle im Projekt verwendet wird. Damit leistet Vue einen direkten Beitrag zur Einhaltung der Anforderung N-R9 hinsichtlich Wartbarkeit und Konsistenz innerhalb des Projektes.

Auf Grundlage dieser technischen und organisatorischen Überlegungen fiel die Wahl für die Entwicklung der HMI-Software auf Vue.js.

5.2 Entwicklung der Raspberry Pi Umgebung

Um das HMI auf dem Raspberry Pi umzusetzen wurde ein Chromium Kiosk erstellt. Dieses wird durch ein Bashskript gestartet, welches automatisch durch Weston beim

Start des Systems aktiviert und auf einem Touch Display dargestellt (vgl. Abbildung 5.6) wird. Das folgende Kapitel erläutert die Implementierung der HMI-Anwendung und alle Voraussetzungen.

5.2.1 Interne Struktur des HMIs auf Betriebssystemebene

Aufbauend zur Konzeption des Human-Machine-Interface (HMI) für das Remote-Labor MICRO wurde eine Architektur entwickelt, die hohe Wartungsfreundlichkeit und Aktualität gewährleistet. Die zugrundeliegende Architektur, dargestellt in Abbildung 5.6, basiert auf einer klaren Trennung von Bedienoberfläche und Systemebene, sowie auf Automatisierung.

Nach dem Einschalten des Geräts initialisiert das Betriebssystem alle benötigten Dienste und startet den grafischen Display-Server Weston, der die Darstellung der Benutzeroberfläche übernimmt. Der Weston-Service startet beim Systemboot das Skript *kiosk.sh*, wodurch der Webbrowser Chromium im Kiosk-Modus geöffnet wird. Dieser Modus stellt sicher, dass ausschließlich das Webinterface angezeigt wird und Bedienende keine Möglichkeit haben, auf andere Systemfunktionen zuzugreifen (siehe Unterabschnitt 3.1.3). Dies trägt maßgeblich zur Verbesserung der User Experience (UX) (vgl. Unterabschnitt 3.1.3) und zur Absicherung des Systems bei, indem es den Zugriff gezielt einschränkt.

Zur Automatisierung wiederkehrender Aufgaben wird ein Cronjob eingerichtet, welcher minütlich das deploy.sh Skript ausführt. Dieses Skript prüft nach Aktualisierungen im GitLab Repository des Web-Interfaces. Im Falle einer neuen Version wird auf Basis dieser das HMI-Webinterface als Docker-Container neu gebaut und bereitgestellt. Dadurch wird eine manuelle Auslieferung neuer Softwareversionen automatisiert. Dieses Skript ist ein essenzieller Bestandteil der CI/CD Pipeline, auf welche in Abschnitt 5.5 näher eingegangen wird.

Diese Grundlage bildet die Basis für den zuverlässigen, skalierbaren und sicheren Betrieb des HMIs in modernen remote-basierten Forschungsszenarien und adressiert explizit die Anforderungen an Usability (N-R5), Wartbarkeit (N-R9) und Sicherheit (F-R4) in einer verteilten Laborumgebung.

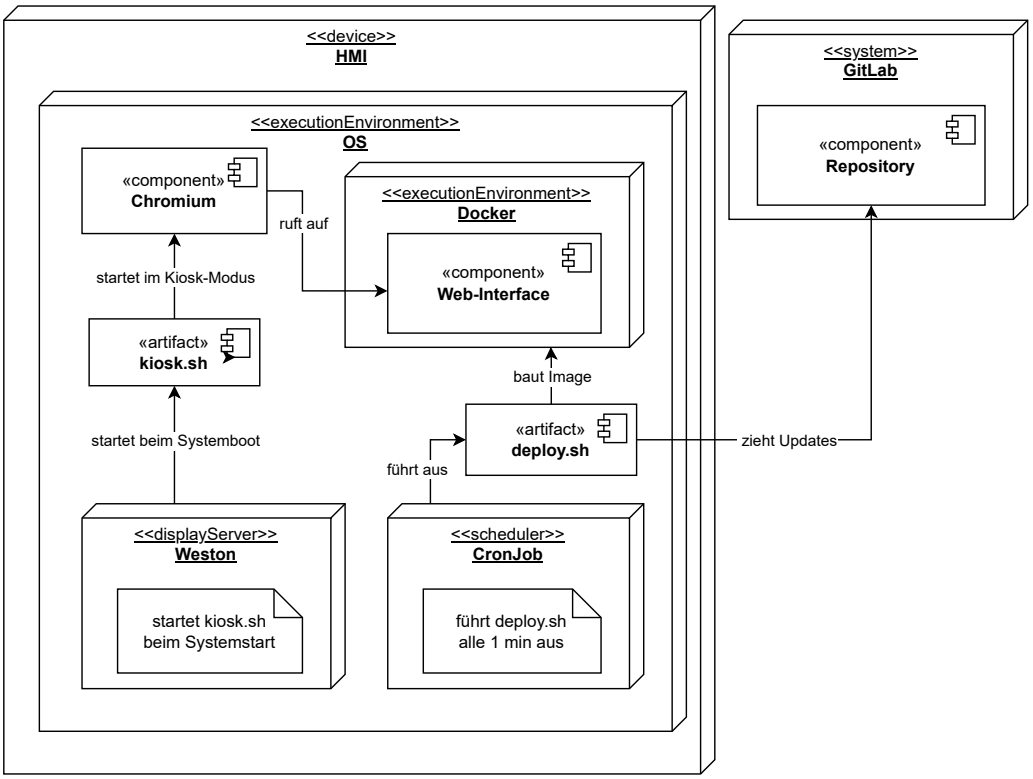


Abbildung 5.6: Innere Struktur des HMIs auf Betriebssystemebene

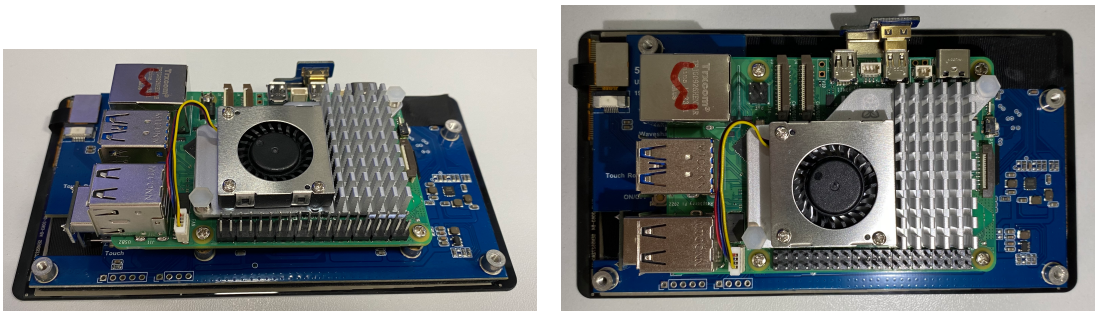


Abbildung 5.7: Rückseite des HMI-Aufbaus des 5.5 Zoll Waveshare Displays

5.2.2 Hardwareaufbau

Die Hardwareplattform besteht aus einem Raspberry Pi, der per HDMI- und USB Schnittstellen mit einem kapazitiven Touch-Display als Hauptinteraktionsfläche verbunden ist (siehe Abbildung 5.7). Die blaue Platine ist die Rückseite des Displays. Darauf ist ein Raspberry Pi 5 aufgeschraubt und mit rechtwinkligen Verbindungsstücken an das Display per HDMI (für Bild) und USB (für Touch) verbunden. Die RGB-LEDs des MACRO-Schrankes werden über eine einfache Mikrocontroller-Einheit angesteuert, die über ein REST-Backend Kommandos empfängt. Die gesamte Einheit soll in den MACRO-Schrank integriert werden und dort zentral für den Laborbetrieb genutzt werden.

5.2.3 Einrichtung der Touch-Displays

Für die Nutzung der Waveshare Displays mit Raspberry Pi Geräten ist es notwendig, spezifische Einstellungen in der Datei *config.txt* auf der SD-Karte des Betriebssystems vorzunehmen. Diese Datei befindet sich im Hauptverzeichnis der SD-Karte und muss nach der Installation entsprechend angepasst werden, um die jeweiligen Displays korrekt anzusteuern.

Die folgenden Konfigurationen sind für die verwendeten Waveshare Displays erforderlich und müssen an das Ende der Datei angefügt werden:

- **15.6 Zoll Waveshare QLED Display (D1)**

```
hdmi_group=2
hdmi_mode=82
hdmi_cvt 1920 1080 60 6 0 0 0
```

- **5.5 Zoll Waveshare Display (D2)**

```
max_framebuffer_height=1920
config_hdmi_boost=10
hdmi_group=2
hdmi_force_hotplug=1
hdmi_mode=87
hdmi_timings=1080 1 26 4 50 1920 1 8
  2 6 0 0 0 60 0 135580000 3
```

- **9.3 Zoll Waveshare Display (D3)**

```
hdmi_group=2
hdmi_mode=87
hdmi_cvt 1600 600 60 6 0 0 0
```

- **15.6 Zoll Waveshare LCD Display (D5)**

```
hdmi_group=2
hdmi_mode=82
hdmi_cvt 1920 1080 60 6 0 0 0
```

- **7 Zoll Waveshare Display (Pi Zero Version) (D6)**

```
hdmi_force_hotplug=1
config_hdmi_boost=10
hdmi_group=2
hdmi_mode=87
hdmi_cvt 1024 600 60 6 0 0 0
```

Diese Einstellungen sorgen dafür, dass die Displays korrekt erkannt und mit der vorgesehenen Auflösung sowie Bildwiederholfrequenz betrieben werden. Für das Raspberry Pi Display (D4) ist keine Konfiguration notwendig.

5.2.4 Aufsetzen des Betriebssystems

Zur Nutzung der HMI-Software auf den Raspberry Pi Geräten wird ein leichtgewichtiges Linux-Betriebssystem (Raspberry Pi OS Lite) eingesetzt. Die Installation von Raspberry Pi OS Lite erfolgt über die Raspberry Pi Imager Software, welche von Raspberry zur Verfügung gestellt wird [79]. Nach der Installation ist noch, wie in Unterabschnitt 5.2.3 näher erläutert, die Konfiguration für die Displays vonnöten, sollte der verwendete Bildschirm eine extra Konfiguration benötigen. Im Anschluss an die Installation sind mehrere Konfigurationsschritte erforderlich, um die grafische Oberfläche für das Betriebssystem bereitzustellen.

Zur Darstellung der Benutzeroberfläche kommt der Display-Server *Weston* (vgl. Unterabschnitt 5.1.2) zum Einsatz. Die benötigten Pakete werden folgendermaßen installiert:

```
$ apt install weston xwayland qtwayland5
```

Anschließend wird ein Systemd-Dienst (siehe Listing 1) eingerichtet, um Weston beim Systemstart automatisch zu starten:

```
$ cp weston.service /etc/systemd/system/weston.service
$ sudo systemctl enable weston.service
```

Dieser Service wird nach dem Start des Systems aktiviert und startet Weston für eine grafische Desktopoberfläche. Zudem werden alle Ausgaben von Weston in einer Logdatei eingetragen, um Debugging zu vereinfachen:

Um den Raspberry Pi ohne manuelle Anmeldung zu starten, wird die Autologin-Funktion über `raspi-config` aktiviert:

```
$ sudo raspi-config
```

Da sich das Gerät im Betrieb im Kiosk-Modus (vgl. Unterabschnitt 3.1.3) befindet und so eine Interaktion mit dem Betriebssystem oder anderen Anwendungen nicht möglich ist, muss ein Zugriff per Fernsteuerung erfolgen. Dies kann man mithilfe der Secure Shell (SSH) erreichen:

```
$ ssh micro@macro-hmi
```

Damit ist nun eine leichtgewichtige Betriebssystem-Basis mit einer voll funktionsfähigen grafischen Oberfläche geschaffen, die als Fundament für die weitere Entwicklung und Integration der HMI-Software dient.

```
# weston.service
[Unit]
Description=Weston Wayland Compositor
After=multi-user.target

[Service]
User=micro
Environment="XDG_RUNTIME_DIR=/run/user/1000"
ExecStart=/usr/bin/weston --tty=1 \
  --log=/home/micro/weston.log
Restart=always
RestartSec=2

[Install]
WantedBy=multi-user.target
```

Listing 1: Inhalt der weston.service Datei

5.2.5 Aufsetzen von Weston

Weston ist bislang nur funktional installiert worden, kann und muss jedoch noch für den Anwendungszweck als grafische Oberfläche des HMIs angepasst werden. Die Datei *weston.ini* dient zur Konfiguration des Weston-Compositors. Sie definiert grundlegende Einstellungen zur Anzeige, Eingabe sowie zum Start von Anwendungen. Die Konfiguration in Listing 2 ist für das Raspberry Pi Touch Display 2 optimiert.

Im *[core]*-Abschnitt wird mit *xwayland=true* die Unterstützung für X11-Anwendungen aktiviert.

Der *[keyboard]*-Abschnitt legt das Tastaturlayout auf Deutsch fest, für den Fall, dass man per angeschlossener Tastatur auf das HMI zugreifen muss.

Der *[output]*-Abschnitt definiert die Ausgabeeinstellungen. Für HDMI-Displays wird der Name *HDMI-A-1* verwendet. Für das Touch Display 2 ist *DSI-2* das korrekte Ausgabegerät, da das Display über ein Flachbandkabel an den DSI Anschluss des Raspberry Pis angeschlossen ist. Die Anzeige wird auf die Auflösung des Displays angepasst. In dem Beispiel (Listing 2) wird dies auf *720x1280* gesetzt. Zudem wird die Anzeige um 270° gedreht, um sie im Horizontalmodus anzeigen zu lassen (*rotate-270*). Der Skalierungsfaktor (*scale*) kann für eine bessere Lesbarkeit der Benutzeroberfläche sorgen. Besonders bei kleineren Displays mit einer hohen Auflösung ist eine höhere Skalierung empfoh-

```
[core]
xwayland=true

[keyboard]
keymap_layout=de

[output]
name=DSI-2
mode=720x1280
transform=rotate-270
scale=1

[input-method]
path=/usr/lib/aarch64-linux-gnu/weston-keyboard

[autolaunch]
path=/home/micro/kiosk/kiosk.sh
```

Listing 2: Beispielhafter Inhalt einer weston.ini Konfigurationsdatei

len. Das 5.5 Zoll Waveshare Display (D2) nutzt den Skalierungsfaktor 2 um eine lesbare Darstellung zu gewährleisten.

In Abschnitt *[input-method]* wird der Pfad zur Onscreen-Tastatur angegeben. Weston stellt eine integrierte Onscreen-Tastatur zur Verfügung. Der Pfad zu dieser kann jedoch je nach verwendetem System unterschiedlich sein und muss manuell ersucht werden.

Der *[autolaunch]*-Abschnitt sorgt dafür, dass das Kiosk-Skript automatisch mit dem Start von Weston ausgeführt wird. Die Benutzeroberfläche des HMIs wird dadurch automatisch gestartet, ohne dass ein manuelles Eingreifen erforderlich ist.

5.2.6 Installation von Docker

Für die Webanwendung wird Docker eingesetzt. Die Installation erfolgt gemäß der offiziellen Anleitung für Debian-basierte Systeme und dem Docker eigenen Installationsskript [30].

Für die Installation von Docker ist lediglich das Ausführen folgender Befehle nötig:

```
$ curl -fsSL https://get.docker.com -o get-docker.sh
$ sudo sh ./get-docker.sh --dry-run
```

Anschließend sollte Docker installiert sein. Unter Linux-Systemen ist es folglich noch üblich, dass jeder Docker Befehl als Root-Nutzer ausgeführt werden muss. Da dies lästig sein kann, gibt Docker einem auch die Möglichkeit, dies zu umgehen, weist aber ausschließlich darauf hin, dass dadurch Sicherheitsrisiken entstehen können [32]. Für das Umgehen müssen folgende Befehle ausgeführt werden [31]:

```
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
```

Für die Entwicklung des HMIs wurde diese Änderung bewusst vorgenommen. Da das System ausschließlich im internen Labornetzwerk betrieben wird und ausschließlich offizielle Docker-Images zum Einsatz kommen, wird das verbleibende Sicherheitsrisiko als ausreichend gering eingeschätzt, um diese Anpassung verantwortungsvoll vornehmen zu können. Dieses Verhalten wird für das automatische Deployment der Anwendung noch relevant und in Abschnitt 5.5 näher erläutert.

5.2.7 Aufsetzen des Kiosks

Um die HMI-Anwendung beim Systemstart automatisch in der Kiosk-Ansicht (vgl. Unterabschnitt 3.1.3) anzuzeigen, wurde ein Shell-Skript *kiosk.sh* implementiert. Dieses Skript startet einen Browser in einem optimierten Modus und lädt die Benutzeroberfläche lokal auf dem Raspberry Pi.

Beim Start des Skripts werden alle Konsolenausgaben in eine Logdatei (*/home/micro/kiosk/kiosk.log*) umgeleitet. Dies dient der Nachvollziehbarkeit im Fehlerfall und fördert die Wartbarkeit des Systems (N-R9). Zusätzlich wird jeder Aufruf des Skripts mit einem Zeitstempel protokolliert. Das gesamte Skript findet sich in Listing 3.

Die ersten drei Zeilen des Skripts aktivieren das besagte Logging für dieses Skript, was alle Konsolenausgaben in die angegebene Logdatei umleitet. Die Ziel-URL der Webanwendung wird in der *URL*-Variable gespeichert und ist auf *http://localhost:8080* gesetzt, welches die Adresse des Docker-Containers der HMI-Anwendung ist. Der zu verwendende Browser kann über die *BROWSER* Variable gewählt werden. Zur Auswahl stehen hier Chromium, Firefox und Midori, wobei die Verwendung von Chromium empfohlen ist und die beiden anderen Browser hauptsächlich nur zu Testzwecken dienen. Da das

```
#!/bin/bash

# activate logging
LOGFILE="/home/micro/kiosk/kiosk.log"
exec >> "$LOGFILE" 2>&1
echo "==== $(date): kiosk.sh started ===="

# URL of the web application to be displayed
URL="http://localhost:8080"

# Choose Browser: chromium or midori
BROWSER="chromium"

# Wait until weston has started
sleep 2

# Start browser in kiosk mode
if [ "$BROWSER" = "chromium" ]; then
  chromium-browser --kiosk --noerrdialogs \
  --disable-infobars --app=$URL \
  --disable-translate --disable-features=TranslateUI \
  --disable-features=Translate \
  --disable-features=TranslateRanker \
  --disable-pinch \
  --enable-wayland-ime \
  --overscroll-history-navigation=0 \
  --incognito --disable-application-cache \
  --disk-cache-size=1 --media-cache-size=1 \
  --aggressive-cache-discard --disable-cache \

elif [ "$BROWSER" = "firefox" ]; then
  MOZ_ENABLE_WAYLAND=1 firefox --kiosk -- "$URL"

elif [ "$BROWSER" = "midori" ]; then
  flatpak run org.midori_browser.Midori -e Fullscreen -a "$URL"
fi

# infinite loop to keep the script running
while true; do
  sleep 1
done
```

Listing 3: Inhalt des kiosk.sh Skripts

System den Wayland-Compositor *Weston* nutzt, wird mit *sleep 2* eine kurze Wartezeit eingefügt, um sicherzugehen, dass dieser vollständig gestartet ist. Abhängig von dem gewählten Browser wird anschließend die Benutzeroberfläche im Kiosk-Modus gestartet. Für Chromium wurden hierzu zahlreiche Parameter gesetzt, um unerwünschte Funktionen wie Übersetzung, Caching, Browser-Benachrichtigungen oder Zoom zu deaktivieren und so die Immersion durch weitere Abgrenzung zu den darunter liegenden Systemen in der Software zu erhöhen. Zum Schluss sorgt eine Endlosschleife dafür, dass das Skript aktiv bleibt und bei einem unerwarteten Browserabbruch nicht beendet wird.

5.2.8 Installationskript

Um den manuellen Installationsprozess zu vereinfachen und eine einheitliche Einrichtung der HMI-Umgebung auf verschiedenen Geräten sicherzustellen, wurde ein automatisiertes Bash-Skript entwickelt. Dieses Installationsskript richtet alle erforderlichen Komponenten ein und ermöglicht es, ein lauffähiges Kiosk-System mit minimalem Aufwand bereitzustellen.

Das Skript liegt mitsamt allen Konfigurationsdateien in einem GitLab-Repository. Zur Nutzung muss das Repository heruntergeladen und das Installationsskript ausgeführt werden. Das Skript erledigt anschließend folgende Aufgaben automatisch:

- *Browserauswahl*: Zu Beginn wird abgefragt, welcher Browser für den Kiosk-Modus verwendet werden soll. Unterstützt werden sowohl Chromium als auch Midori.
- *Installation benötigter Pakete*: Es werden die Pakete *weston, xwayland* und *qtwayland5* installiert. Je nach gewähltem Browser wird zusätzlich entweder *chromium-browser* oder *midori* installiert.
- *Konfiguration von Weston*: Die Datei *weston.service* wird in das systemd-Verzeichnis kopiert und aktiviert. Anschließend wird die Konfigurationsdatei *weston.ini* im Benutzerverzeichnis abgelegt. Falls vorhanden, wird der Pfad zum *weston-keyboard* automatisch erkannt und eingetragen.
- *Einrichtung des Kiosk-Skripts*: Das Skript *kiosk.sh* wird in das Benutzerverzeichnis kopiert, ausführbar gemacht und im Autostart-Bereich der *weston.ini* referenziert. Der gewählte Browser wird im Skript automatisch hinterlegt.

- *Docker-Installation*: Docker wird über das offizielle Installationsskript von Docker [30] installiert. Der Benutzer wird zur Docker-Gruppe hinzugefügt, um die Notwendigkeit der Verwendung von Root-Rechten für die Verwendung von Docker-Befehlen zu umgehen (vgl. Unterabschnitt 5.2.6).

- *Deploy-Skript und Umgebungsvariablen*: Das *deploy.sh*-Skript wird kopiert, ausführbar gemacht und ein Beispiel für die *.env*-Datei bereitgestellt. Dieses muss vom Benutzer anschließend noch manuell angepasst werden.

- *Cronjob und Logrotation*: Ein Cronjob wird eingerichtet, der das *deploy.sh*-Skript (vgl. Abschnitt 5.5) jede Minute ausführt. Zusätzlich werden Konfigurationen für *logrotate* erstellt, um die Protokolldateien regelmäßig zu rotieren und zu komprimieren.

- *Abschließende Hinweise*: Das Skript gibt am Ende Hinweise auf ausstehende manuelle Schritte, z. B. das Einfügen des GitLab Tokens, das Setzen der URL im Kiosk-Skript oder das Anpassen der Bildschirmauflösung in der Konfigurationsdatei von Weston.

Es empfiehlt sich, das Skript auf einem frisch installierten System auszuführen. Eine abschließende Benutzerabfrage ermöglicht es, das System nach Abschluss der Einrichtung direkt neu zu starten, um alle Änderungen wirksam werden zu lassen.

Durch den Einsatz des Installationsskripts werden folgende Vorteile erzielt:

- *Zeitersparnis*: Die manuelle Ausführung zahlreicher Einzelschritte entfällt.

- *Reproduzierbarkeit*: Alle Systeme können identisch eingerichtet werden.

- *Fehlerminimierung*: Durch automatische Pfadfindung und Konfiguration werden typische Fehlerquellen (z. B. falsche Tastaturlayouts oder fehlende Abhängigkeiten) vermieden.

Somit stellt das Skript einen zentralen Bestandteil des Installationsprozesses dar und unterstützt die skalierbare Ausrollung der HMI-Anwendung auf mehreren Geräten.

5.3 Umsetzung der HMI-Benutzeroberfläche

Ein zentrales Ziel dieser Arbeit war die Entwicklung einer intuitiven und robusten grafischen Benutzeroberfläche zur Steuerung, Überwachung und Verwaltung der MICRO-Stationen. Die grafische HMI-Anwendung stellt dabei die primäre Interaktionsschnittstelle zwischen Nutzenden und dem System dar. Entsprechend hoch sind die Anforderungen an Bedienbarkeit, Zuverlässigkeit, Erweiterbarkeit und Touch-Tauglichkeit (siehe Abschnitt 4.1).

Dieser Abschnitt beschreibt die Umsetzung der Benutzeroberfläche sowohl aus funktionaler als auch aus gestalterischer Sicht. Zunächst wird ein Überblick über die Architektur, die Realisierung des Navigationskonzepts und zentrale Interaktionsmodule gegeben. Anschließend werden spezifische Designprinzipien und Herausforderungen behandelt, die sich insbesondere durch die Touch-Bedienung ergeben haben, sowie die entwickelten Lösungsansätze zur Verbesserung der UX.

5.3.1 Überblick der entwickelten Ansichten

Der folgende Abschnitt beschreibt die technische und gestalterische Umsetzung der HMI-Anwendung. Ziel war es, eine benutzerfreundliche, wartbare und erweiterbare Oberfläche zur Steuerung und Überwachung der MICRO-Infrastruktur zu entwickeln. Im Fokus stehen dabei sowohl die Interaktionskonzepte und Benutzerführung als auch konkrete Implementierungsdetails wie Routing, Komponentenstruktur und Kommunikationsmechanismen mit dem Backend. Die nachfolgenden Abschnitte führen durch zentrale Bestandteile der Anwendung und erläutern deren Funktion, Aufbau und technische Besonderheiten am Beispiel einzelner UI-Module.

Startseite und Navigationsleiste der HMI-Anwendung

Die Startseite (dargestellt in Abbildung 5.8) des HMIs ist als zentrale Navigationsoberfläche, wie in der Konzeption geplant, implementiert worden. Das Layout orientiert sich an einem kachelbasierten Design, das eine schnelle und einfache Bedienung, insbesondere auf Touch-Displays, ermöglicht.

Zentral dargestellt sind in Kacheln platzierte Schaltflächen: „Station Overview“, „Robot Controller“, „LED Controller“ und „Reports“. Diese Schaltflächen verteilen sich auf dem Bildschirm und bieten dem Nutzer einen direkten Zugang zu den Untermenüs des

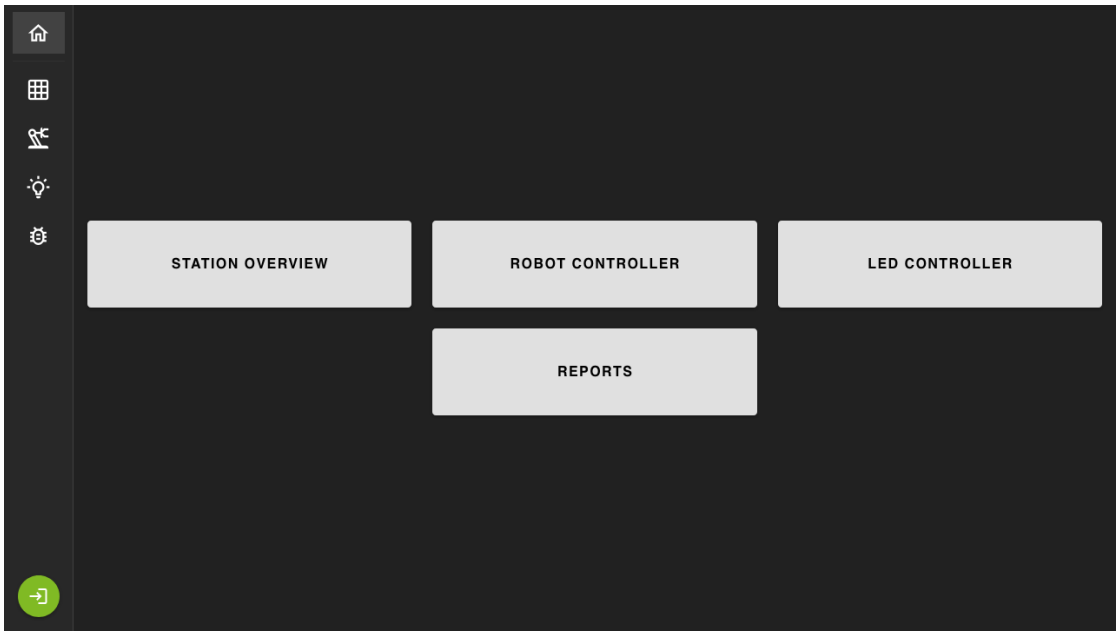


Abbildung 5.8: Startseite der HMI-Anwendung

Systems. Durch die Beschriftung und die großzügigen Abmessungen sind sie speziell auf eine fehlerarme Bedienung mit Touch-Eingaben ausgelegt und fördern so die UX (N-R6).

Auf der linken Seite befindet sich eine vertikale Navigationsleiste mit Symbolen. Hier werden die schnelle Orientierung und der Wechsel zwischen verschiedenen Modulen des HMIs gefördert. Die Navigation umfasst unter anderem ein Home-Symbol zur Rückkehr auf die Startseite sowie weitere Icons für spezifische Untermenüs.

Am unteren Rand der Leiste befindet sich ein grüner Button mit Pfeilsymbol, welcher als manueller Login-Button genutzt werden kann, sollte es bei der automatischen Authentifizierung mit dem Produktions- oder Entwicklungsbackend zu Komplikationen kommen.

Die Menüknöpfe auf der Seitenleiste spiegeln dieselben auf der Startseite des HMIs wider. Sowohl die Buttons der Seitenleiste als auch die auf der Startseite generieren sich automatisch aus den verlinkten Seiten im Vue-Router [130]. In Listing 4 findet sich ein Auszug aus der Vue-Router-Datei. Es ist zu erkennen, dass alle Unterseiten, die sowohl in der Navigationsleiste als auch auf der Startseite angezeigt werden sollen, als Unterpfade von */navigation-views* abgelegt sind. Jede dieser Seiten erhält zusätzlich einen Titel und ein Icon, welche anschließend zur Darstellung des jeweiligen Menüpunktes in der Seitennavigation und auf der Startseite verwendet werden. Durch diese Implemen-

tierung vermeidet man redundanten Code für die Startseite und die Navigationsleiste und fördert so die Wartbarkeit und Anpassbarkeit des Systems (N-R9, N-R10).

```
const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    ...
    {
      path: '/navigation-views',
      children: [
        {
          path: 'test',
          name: 'test-view',
          meta: {
            title: 'Test',
            icon: 'mdi-flask-outline',
            visible: true
          },
          component: () => import('@/views/TestView.vue'),
        },
      ]
    },
  ],
})
```

Listing 4: Auszug aus dem Vue-Router der HMI-Anwendung

Insgesamt trägt die zentrale Startseite in Kombination mit der dynamisch generierten Navigationsleiste maßgeblich zu einer intuitiven und wartungsfreundlichen Benutzerführung bei, indem sie sowohl die Touch-Bedienbarkeit verbessert als auch redundanten Code vermeidet.

Übersicht und Ansicht der Station

Die Stationenübersicht, dargestellt in Abbildung 5.9, bildet ein zentrales Element des HMIs und präsentiert, wie konzeptionell geplant, sämtliche verfügbaren Versuchseinheiten in einer strukturierten, kachelorientierten Grid-Ansicht (F-R1). Jede Station wird durch ein eigenes Kachel-Widget repräsentiert, das sowohl grundlegende Informationen (wie Name und ID) als auch den aktuellen Reservierungsstatus anzeigt. Mit einem kleinen Augensymbol wird dargestellt, ob die Station momentan für MICRO-Nutzer

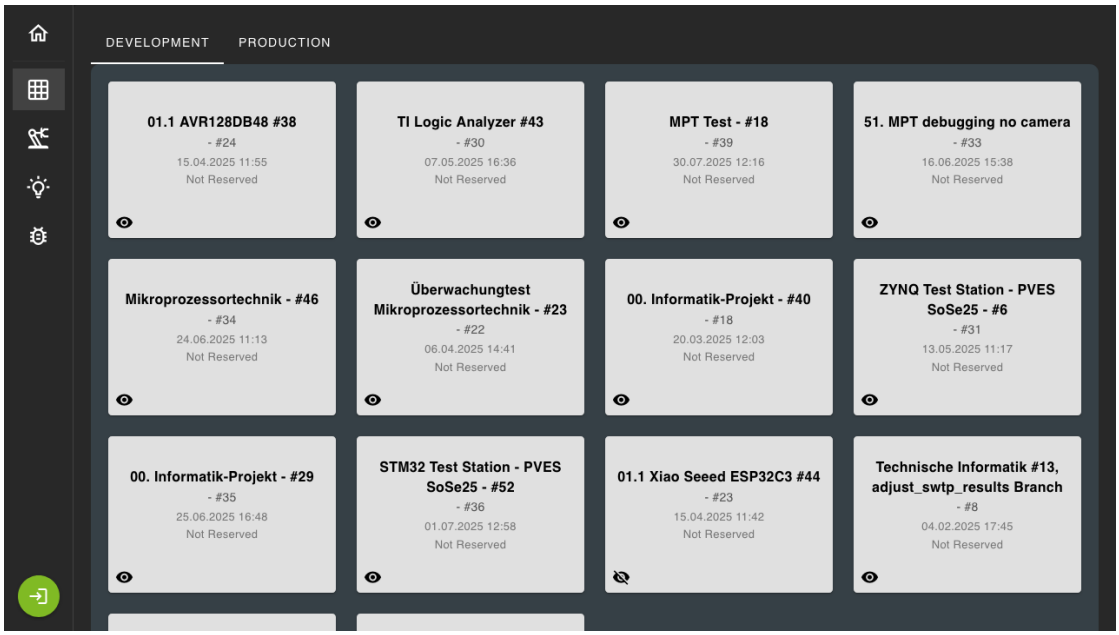


Abbildung 5.9: Stationsübersicht der HMI-Anwendung

sichtbar oder unsichtbar ist. Dies unterstützt MICRO-Administratoren bei der Erfassung und Auswahl der gewünschten Experimentierstation.

Die Stationsübersicht setzt auf ein, wie in der Konzeption geplantes, Tab-System: Über die Navigation am oberen Rand lassen sich unterschiedliche Quellen (z. B. „Development“ und „Production“) durch eigene Tabs separieren. Jedem Tab ist ein spezifischer Backend-Endpunkt zugeordnet, über den die zugehörigen Stationen abgefragt werden. Dies fördert eine übersichtliche Trennung von unterschiedlichen Laborbereichen (z. B. zwischen Entwicklungs- und Produktionsstationen) und erleichtert die Erweiterbarkeit durch weitere Kategorien. Sollte es beispielsweise zukünftig einen zweiten MACRO-Schrank geben, könnte dieser durch einen eigenen Tab separiert dargestellt werden.

Die Implementierung macht es besonders einfach, neue Tabs für zusätzliche Stationsquellen hinzuzufügen: Es genügt, ein weiteres Objekt mit Name und API-Route in das „tabs“-Array zu ergänzen (siehe Listing 5). Das dynamische Rendering sorgt dafür, dass für jedes Tab automatisch die passende Grid-Ansicht in eigenem Tab mit den zugehörigen Stationen erstellt wird. Die Integration weiterer Stationstypen oder Backend-Quellen kann somit flexibel und ohne Anpassung der Kernlogik erfolgen und fördert somit eine einfache Erweiterbarkeit (N-R10) und Wartbarkeit (N-R9).

Wird eine Station ausgewählt, gelangt man in eine Detailansicht, wie in Abbildung 5.10

```
const tabs = ref([
  { name: 'Development', items: [], route: 'dev-api'},
  { name: 'Production', items: [], route: 'prod-api'},
]);
```

Listing 5: Auszug aus dem Vue-Router der HMI-Anwendung

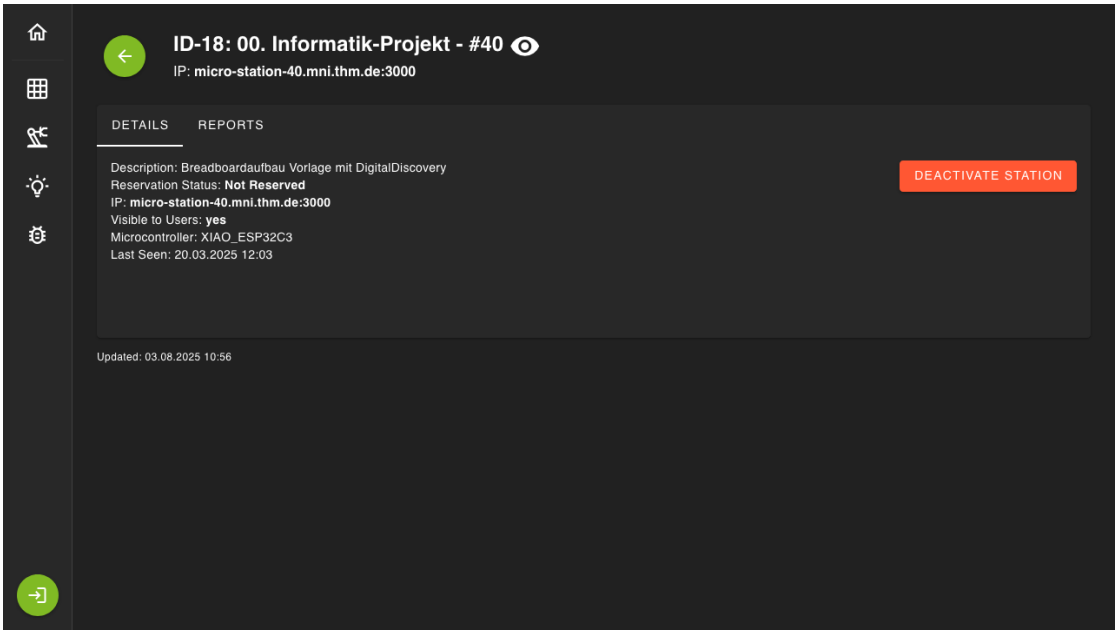


Abbildung 5.10: Detailansicht einer Station der HMI-Anwendung

dargestellt. In dieser Ansicht erhält ein MICRO-Administrator weiterführende Informationen zur jeweiligen Station und kann mit dieser interagieren. Aktuell beschränkt sich die Interaktion auf das Aktivieren bzw. Deaktivieren der Station, wodurch sie für MICRO-Nutzer sichtbar oder unsichtbar geschaltet wird. Perspektivisch lässt sich diese Funktionalität jedoch um zusätzliche Aktionen erweitern, beispielsweise zur Bearbeitung von Stationsdetails.

Wie konzeptionell geplant beinhaltet die Detailansicht einen „Report“-Tab, dargestellt in Abbildung 5.11. Dort können vom System gesammelte Nutzermeldungen zur jeweiligen Station eingesehen und bei Bedarf gelöscht werden (F-R2).

Bislang erfolgt die Aktualisierung der Stationsdaten im HMI alle fünf Sekunden über eine REST-Anfrage an das jeweilige Backend. Dieser Ansatz verursacht jedoch einen erheblichen Netzwerk-Overhead, da viele Anfragen keine neuen Informationen liefern. Im Rahmen dieser Arbeit wurde daher eine Testimplementierung im „Spring-

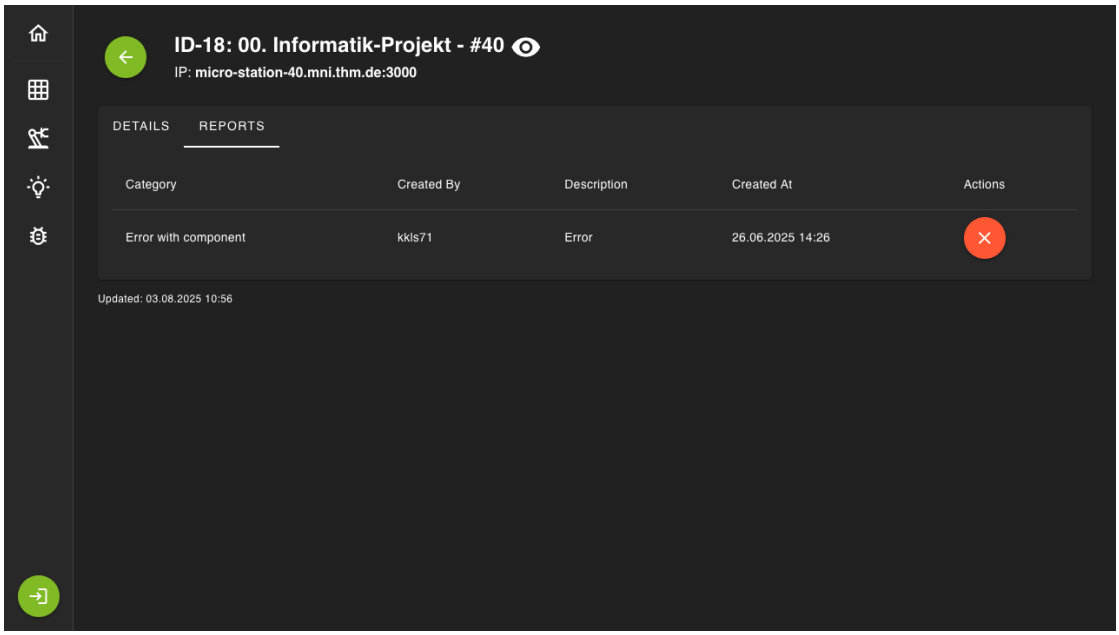


Abbildung 5.11: Ansicht für Nutzermeldungen einer Station der HMI-Anwendung

backend“ realisiert, die stattdessen auf Server-Sent-Events (SSE) basiert. SSE ist eine Alternative zu WebSockets, bei der der Server eigenständig Nachrichten an den Client senden kann, ohne dass dieser zuvor eine Anfrage stellen muss, wie es bei klassischen Hypertext Transfer Protocol (HTTP)-Verfahren der Fall ist. Auf diese Weise kann das Backend gezielt nur dann Aktualisierungen an das HMI senden, wenn tatsächlich Änderungen vorliegen, wodurch unnötiger Datenverkehr vermieden und die Effizienz des Systems erhöht wird. Die Testimplementierung ist aber noch nicht final entwickelt worden und bedarf weiterer Arbeit, bevor sie produktiv verwendet werden kann.

Insgesamt bietet die Stationsübersicht eine skalierbare (N-R10), wartungsfreundliche (N-R9) und benutzerzentrierte Lösung zur Verwaltung und Darstellung verteilter Experimentierstationen. Durch das Tab-basierte Layout, die dynamische Integration neuer Quellen und die strukturierte Detailansicht mit Interaktions- und Meldefunktion wird eine effiziente Bedienung für Administratoren ermöglicht. Die prototypische Einführung von SSE zur effizienteren Datenaktualisierung zeigt zudem das Potenzial zur weiteren Optimierung der Systemperformance.

Steuerung des Portalroboters

Die Steuerungsansicht für den Portalroboter wurde im Verlauf der Entwicklungsphase als provisorische Lösung umgesetzt und ist in Abbildung 5.12 und Abbildung 5.13 dar-

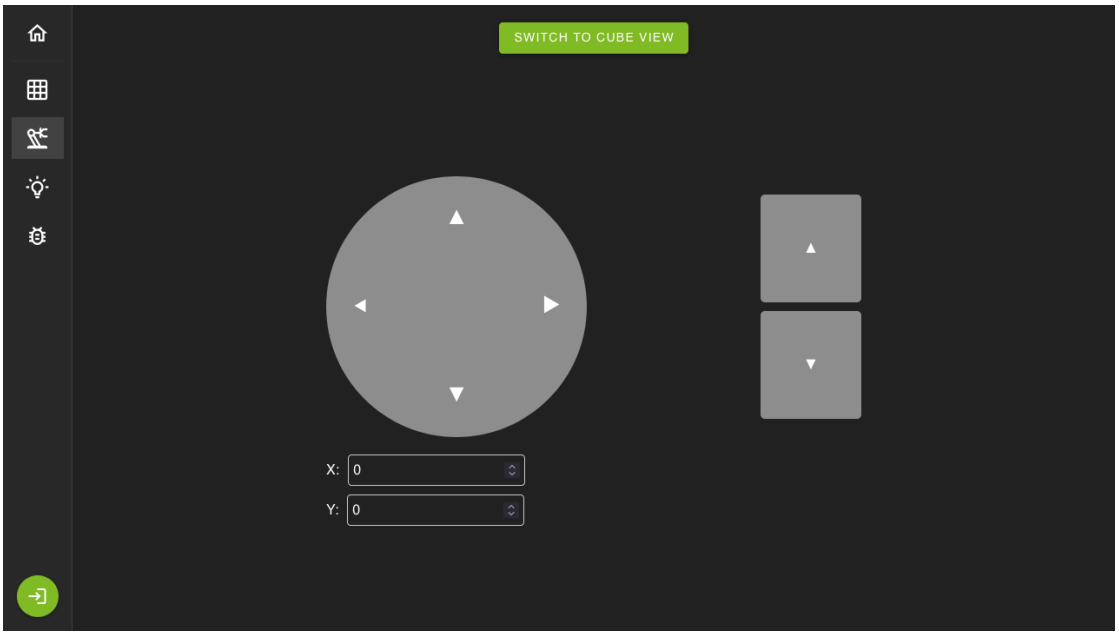


Abbildung 5.12: Manuelle Robotersteuerung der HMI-Anwendung

gestellt. Da der Roboter zum Zeitpunkt der Implementierung noch nicht vollständig funktionsfähig war und konkrete Anforderungen sowie technische Details zur Steuerung ausstanden, beschränkte sich die Umsetzung auf eine vorläufige Benutzeroberfläche.

Die Oberfläche bietet grundlegende, richtungsbasierte Steuerungselemente (Pfeiltasten für Bewegungen auf der x- und y-Achse sowie für vertikale Bewegungen) auf der manuellen Steuerungsansicht und wurde ausgehend von der Konzeption aus Unterunterabschnitt 4.3.3 implementiert. Zusätzlich existieren Eingabefelder zur Positionswahl, sodass verschiedene Steuerungsparadigmen erprobt werden können. Das finale Design und der Umfang der erforderlichen Schaltflächen werden abhängig von der tatsächlichen technischen Umsetzung des Roboters sowie dem Feedback aus den ersten praktischen Erprobungen zu einem späteren Zeitpunkt definiert und spezifiziert. Die derzeitige Lösung dient somit lediglich als Platzhalter und experimentelle Testumgebung.

Zusätzlich zu dem geplanten Design aus Abbildung 4.5 wurde eine zweite Steuerungsmöglichkeit entworfen, dargestellt in Abbildung 5.13. Hierbei soll es möglich sein, durch Drücken auf eines der Felder, den Cube an der entsprechenden Position im MACRO-Schrank anzusteuern. So wird eine einfache Bedienung ermöglicht, ohne den Roboter manuell in die richtige Position bewegen zu müssen. Diese Steuerung muss jedoch

Abbildung 5.13: Cube-basierte Robotersteuerung der HMI-Anwendung

ebenfalls erst noch roboterseitig implementiert werden und ist bislang nicht funktional nutzbar.

Steuerung der RGB-LEDs

Die Steuerung der RGB-LEDs ist als interaktive Oberfläche gestaltet und ermöglicht eine flexible, benutzerfreundliche Auswahl und Anpassung der Lichtanimationen im Remote-Labor. Der Menüpunkt ist als Screenshot in Abbildung 5.14 und Abbildung 5.15 zu finden. Zusätzlich zu den geplanten Schaltflächen steht, wie bereits in Unterunterabschnitt 4.3.3 als Erweiterung vorgesehen, oberhalb der Farbwahl ein Dropdown-Menü, über das verschiedene Animationsmodi ausgewählt werden können. Diese Animationsoptionen werden dynamisch vom angebundenen RGB-Controller geladen, sodass jederzeit aktuelle und, falls vorhanden, neu hinzugefügte Animationen unmittelbar zur Verfügung stehen.

Auf Basis der gewählten Animation kann der Nutzer die Farbe der Animation auswählen, sollte diese das unterstützen (siehe Abbildung 5.14). Im dargestellten Prototypen ist die Farbauswahl teilweise deaktiviert, falls die aktuell ausgewählte Animation keine individuelle Farbeingabe vorsieht („Color selection unavailable“, siehe Abbildung 5.15). Die gewählten Einstellungen werden per Klick auf die Schaltfläche „SET ANIMATION“ an den Controller übertragen, der daraufhin die entsprechende Animation an den LEDs

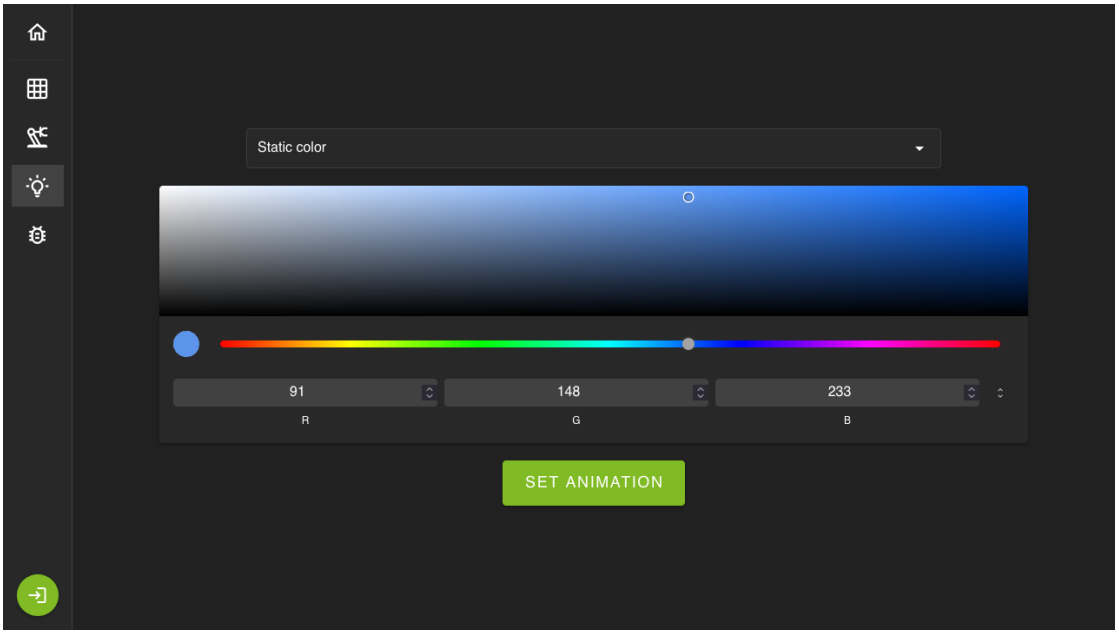


Abbildung 5.14: RGB-LED Steuerung der HMI-Anwendung mit Farbwahl

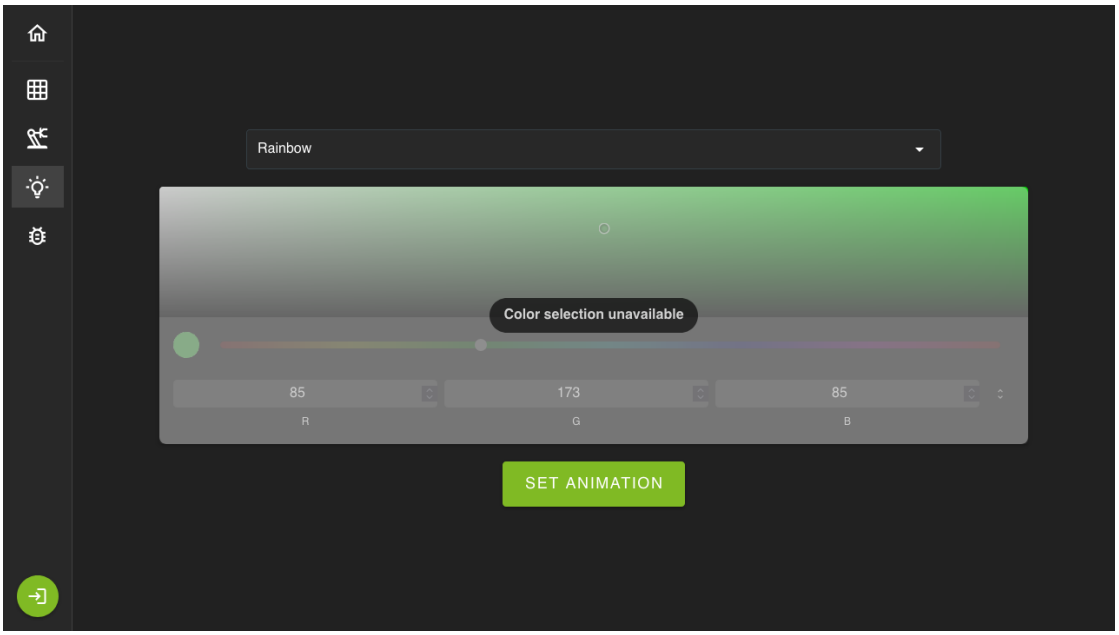


Abbildung 5.15: RGB-LED Steuerung der HMI-Anwendung ohne Farbwahl

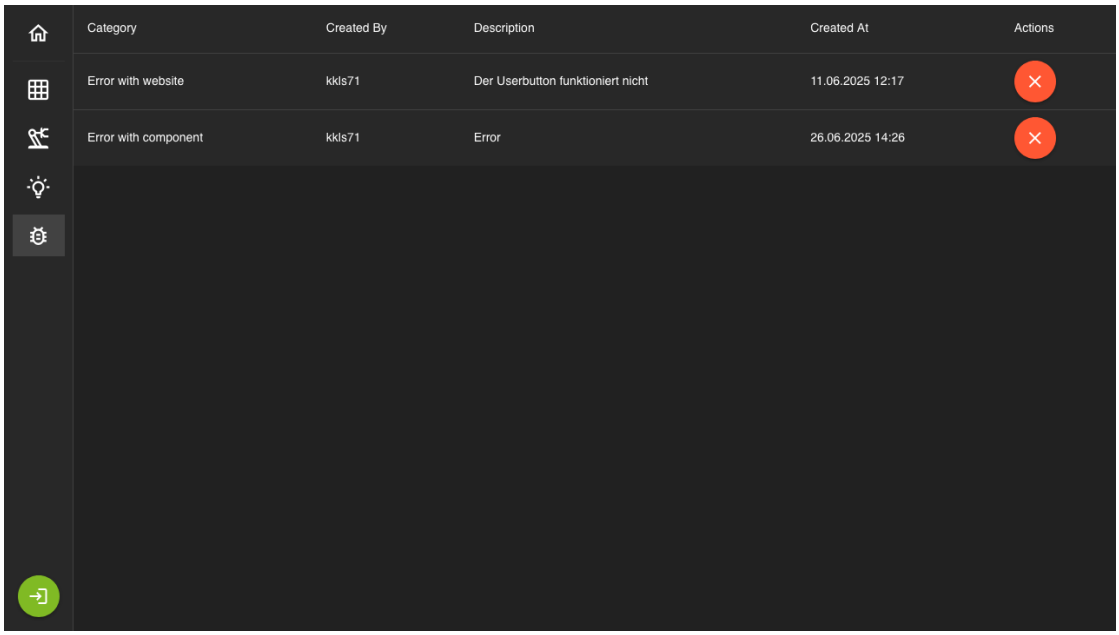


Abbildung 5.16: Ansicht der systemweiten Nutzermeldungen der HMI-Anwendung

darstellt. Eine bildliche Darstellung der LEDs findet sich in Abbildung 2.6 und Abbildung 2.7.

Ansicht der Nutzermeldungen

Das HMI dient außerdem als ein Darstellungswerkzeug für das zentralisiertes Meldungssystem von MICRO (F-R2), das für alle systemweiten Status- und Fehlermeldungen verantwortlich ist. So finden sich auf der Reports-Menüseite (Abbildung 5.16) alle Reports für das System wieder. Hier werden sowohl Fehler des MICRO-Frontends, als auch Probleme der einzelnen Stationen gebündelt präsentiert.

Im Interface wird bei kritischen Aktionen, wie beispielsweise dem Löschen einer Fehlermeldung, zunächst ein modaler Dialog angezeigt (siehe Abbildung 5.17). Dieser fordert den Nutzer auf, den Vorgang zu bestätigen oder abzubrechen, und weist klar auf die Konsequenz hin, dass eine Löschung nicht rückgängig gemacht werden kann. Erst nach expliziter Bestätigung wird die Aktion durchgeführt. Dadurch wird die Gefahr versehentlicher, nicht rückgängig zu machender Änderungen minimiert und die Bedienungssicherheit erhöht (F-R4).

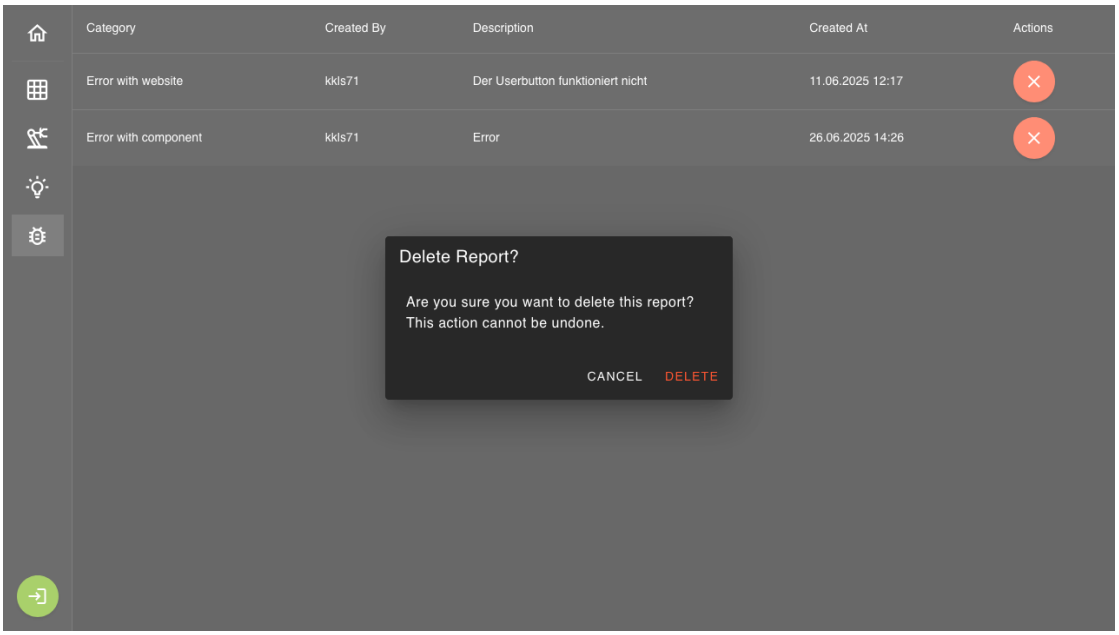


Abbildung 5.17: Bestätigungsabfrage zum Löschen von Nutzermeldungen in der HMI-Anwendung

Statusmeldung innerhalb des HMIs

Ergänzend zu den Menüpunkten informiert eine Snackbar-Komponente am unteren Bildschirmrand über kontextabhängige Systemereignisse, wie etwa das erfolgreiche Laden von Daten, das Scheitern einer Kommunikation mit einer Stationsquelle oder allgemeine Systemfehler über alle Menüpunkte hinweg. Diese Meldungen erscheinen automatisch und verschwinden nach kurzer Zeit wieder, sodass sie den Arbeitsfluss nicht unterbrechen, aber dennoch zuverlässig wahrgenommen werden können.

5.3.2 Designprinzipien

Das Interface wurde nach dem Prinzip „mobile first“ gestaltet (N-R2), mit dem Versuch, möglichst klare, abgegrenzte Komponenten und große Interaktionsflächen zu schaffen (N-R6). Der Fokus liegt auf einfacher visueller Hierarchie und intuitiver Bedienbarkeit (N-R5). Farbgebung und Symbolik folgen etablierten Konventionen (z. B. Grün für „bereit“, Rot für „Fehler“), um Missverständnisse zu vermeiden. Außerdem ist die Farbgebung dieselbe des MICRO-Frontends und folgt dem THM-Coorporate-Design (N-R4) [87].

Für ein einfaches und konsistentes Design der Oberfläche wurde Vuetify [128] verwendet. Vuetify ist ein Erweiterungspaket für Vue.js, das eine Vielzahl vorgefertigter Komponenten auf Grundlage von Googles Material Design [53] bereitstellt. Durch die Nutzung dieser Komponenten, etwa für responsive Buttons, Dialoge oder Navigationsleisten, wird die Entwicklung moderner und konsistenter Benutzeroberflächen deutlich vereinfacht. Viele Interaktionsmuster und Layouts sind dabei bereits integriert, was sowohl Entwicklungsaufwand reduziert (N-R9) als auch die Benutzerfreundlichkeit verbessert (N-R6). Außerdem kommen auf der Benutzeroberfläche Material-Design-Icons zum Einsatz, die durch ihre weite Verbreitung bereits vielen Nutzerinnen und Nutzern vertraut sind und somit zur Verbesserung der UX beitragen (N-R6).

5.3.3 Navigationsrealisierung

Die Benutzerführung ist bewusst flach gehalten (N-R7): Die Startansicht (siehe Abbildung 5.8) zeigt einen Überblick in Form von Buttons über alle existierenden Navigationsseiten. Von dort aus gelangen die Nutzerinnen und Nutzer zu allen relevanten Seiten, wie etwa der Stationsübersicht oder dem RGB-Controller. Die Rücknavigation ist immer über eine fixierte Symbolleiste an der linken Seite oder bei tief verschachtelten Untermenüs über einen Zurückknopf möglich (siehe Abbildung 5.10).

5.3.4 Umsetzung der Touch-Design-Herausforderungen

Die in Unterabschnitt 4.3.4 beschriebenen Ansätze zum Umgang mit dem Midas-Touch- und Fat-Finger-Problem wurden in der finalen HMI-Implementierung vollständig berücksichtigt.

Midas-Touch-Problem

Zur Vermeidung unbeabsichtigter Eingaben wurden kritische Bedienelemente, wie die Schaltfläche zum Löschen von Reports, mit einer zusätzlichen Bestätigungsabfrage versehen. Erst nach einer expliziten Bestätigung wird die zugehörige Funktion ausgeführt.

Für alle Schaltflächen kamen Vuetify-Standardkomponenten (*v-btn*) zum Einsatz, die Aktionen erst nach dem Loslassen der Berührung ausführen. Dadurch können Eingaben während der Berührung abgebrochen werden, wenn der Finger vom Element wegbewegt wird. Ergänzend wurde bei jeder Interaktion ein visuelles Feedback (Notifikati-

on oder eine Animation auf dem Eingabebereich) integriert, um Nutzern unmittelbar Rückmeldung über die erfasste Eingabe zu geben.

Fat-Finger-Problem

Zur Verbesserung der Treffgenauigkeit wurden sämtliche interaktive Elemente gemäß den Material-Design-Vorgaben und Empfehlungen [53] gestaltet. Zur Umsetzung dieser Vorgaben wurden konsistent Vuetify-Komponenten verwendet, ergänzt durch angepasste Abstände (Padding, Margin), um versehentliche Berührungen benachbarter Elemente zu vermeiden.

Die Benutzeroberfläche wurde gezielt auf wesentliche Interaktionspunkte reduziert. Dadurch wurden kleine, dicht platzierte Schaltflächen vermieden. Um die Praxistauglichkeit sicherzustellen, erfolgte die Überprüfung der Layouts regelmäßig auf dem Zielgerät (Raspberry Pi Touch-Display). Während dieser Tests wurden Bedienelemente, die als zu klein oder schwer erreichbar wahrgenommen wurden, nachjustiert.

Ergebnisse

Durch diese Maßnahmen traten in der abschließenden Evaluation (vgl. Kapitel 6) keine unbeabsichtigten Eingaben auf. Die Teilnehmenden berichteten dabei, die Bedienelemente zuverlässig und ohne Probleme bedienen zu können. Eine detaillierte Analyse der Ergebnisse findet sich in Kapitel 6.

5.4 Systemintegration und Schnittstellen

Die Integration des HMIs in das bestehende MICRO-System ist ein zentraler Aspekt dieser Arbeit. Um eine zuverlässige (N-R8) und erweiterbare (N-R10) Kommunikation zwischen Benutzerschnittstelle, Backendsystemen und externer Hardware zu gewährleisten, wurden bestehende Schnittstellen analysiert, erweitert und neue Verbindungen geschaffen. Ziel ist es, eine modulare Systemarchitektur zu realisieren, die eine klare Trennung zwischen Benutzerinteraktion, Datenverarbeitung und physischer Steuerung ermöglicht. In den folgenden Abschnitten werden die technischen Grundlagen dieser Integration, der Umgang mit persistenten Daten, die Authentifizierung sowie die Einbindung von Zusatzkomponenten wie der RGB-Steuerung detailliert beschrieben.

5.4.1 Schnittstellen zu bestehenden Systemen

Das HMI ist zentraler Bestandteil der Bedienoberfläche und agiert als Vermittlungsstelle zwischen MICRO-Administratoren und dem zugrunde liegenden Systemverbund. Es kommuniziert über standardisierte REST-Schnittstellen (vgl. Unterabschnitt 3.2.1) mit den bestehenden Backends des MICRO-Systems, die sowohl in einer Produktions- als auch in einer Entwicklungsumgebung betrieben werden.

Abbildung 4.1 stellt grafisch dar, wie das HMI in das bestehende MICRO-System integriert ist und mit welchen Komponenten es kommuniziert. Ziel der Architektur ist es, eine klare Trennung zwischen Benutzerinteraktion, Datenverarbeitung und Hardwareansteuerung zu gewährleisten. Das HMI überträgt Eingaben und Steuerkommandos an das jeweilige Backend, welches wiederum die Verarbeitung übernimmt und den Zugriff auf die zugehörige Datenbank sowie die angebundenen Stationen realisiert. Darüber hinaus interagiert das HMI direkt mit Zusatzkomponenten, wie etwa dem RGB-Controller oder in Zukunft auch dem Portalroboter.

Zu Beginn der Entwicklung stellte sich die gleichzeitige Nutzung von Entwicklungs- und Produktionsbackend als Herausforderung dar. Beide Umgebungen bieten identische REST-Schnittstellen unter derselben Pfadstruktur (z. B. */lab/api*), jedoch unter unterschiedlichen Domains.

Zur Lösung dieses Problems wurde ein HTTP-Proxy auf Basis von Vite implementiert (siehe Listing 6). Die Proxy-Konfiguration leitet alle Anfragen abhängig vom Pfadpräfix entweder an das Entwicklungs- (*/dev-api*) oder an das Produktionsbackend (*/prod-api*) weiter. Intern werden diese Präfixe entfernt und die Anfrage wird auf den gemeinsamen Pfad */lab/api* gemappt.

Ein besonderes Detail ist das Umschreiben der *Set-Cookie*-Header. Da beide Umgebungen den gleichen Pfad (*/lab/api*) verwenden, würde der Browser die Cookies nicht korrekt trennen. Eine eigene Funktion ersetzt daher den *Path*-Wert im Cookie-Header dynamisch durch den jeweiligen Proxy-Pfad (*/dev-api* bzw. */prod-api*). Damit wird sichergestellt, dass Cookies, wie ein Token, der die Authentifizierung speichert, korrekt zugeordnet und keine Session-Konflikte erzeugt werden.

Diese Proxy-Lösung ermöglicht eine saubere Trennung der Umgebungen bei gleichzeitig einheitlicher Codebasis. Der Proxy versteckt die Komplexität hinter einer klaren Schnittstelle und erlaubt:

```
proxy: {
    '/dev-api': withCookieRewrite({
        target: 'https://micro-dev.mni.thm.de',
        changeOrigin: true,
        secure: false,
        rewrite: (path) => path.replace(/^\/dev-api/,
        backend_base_route),
    }, '/dev-api'),

    '/prod-api': withCookieRewrite({
        target: 'https://micro.mni.thm.de',
        changeOrigin: true,
        secure: false,
        rewrite: (path) => path.replace(/^\/prod-api/,
        backend_base_route),
    }, '/prod-api'),
},
```

Listing 6: Auszug der Proxy-Konfiguration aus vite.config.ts

- eine einfache Umstellung der Umgebung durch Konfiguration statt Codeänderungen
- konsistente API-Aufrufe in der Applikation (relativ zu */dev-api* oder */prod-api*)
- eine sichere Trennung der Session-Cookies durch dynamisches Rewriting

Die Architektur bleibt somit wartbar, sicher und flexibel gegenüber zukünftigen Erweiterungen. Durch diese Entkopplung der Komponenten (Abbildung 4.1) wird auch die Wiederverwendbarkeit und Austauschbarkeit einzelner Systemteile erleichtert (N-R9).

5.4.2 Umgang mit persistenten Daten

In der entwickelten HMI-Anwendung werden für den Umgang mit persistenten Daten zentrale State-Management-Lösungen mittels *Pinia-Stores* [88] eingesetzt. Das Problem bei Frameworks wie Vue.js ist, dass gesetzte Variablen in Komponenten die Daten nicht über den ganzen Verlauf der Nutzung der Anwendung behalten. Sobald eine Komponente aus dem DOM entladen wird, verlieren dessen Variablen ihren Wert. Um nun Daten konsistent über den Verlauf der Anwendung speichern zu können, kann man diese

entweder im Browser eigenen Local-Storage [90] speichern, oder man verwendet vorgefertigte Packages, wie etwa den Pinia-Store, welcher intern zwar den Local-Storage nutzt, jedoch noch weitere Funktionalitäten bietet.

Diese Dekomposition in spezialisierte Stores ermöglicht eine strukturierte Trennung der Datenhaltung nach Komponenten und vereinfacht dabei sowohl die Verwaltung als auch die Nutzung innerhalb von Vue.js.

In den in der HMI-Anwendung verwendeten Pinia-Stores werden sämtliche entitätsbasierten Informationen, wie etwa zu Stationen, Reports oder Benutzern, verwaltet. Dies stellt sicher, dass der Anwendungszustand konsistent bleibt, auch wenn verschiedene UI-Komponenten gleichzeitig darauf zugreifen oder diesen verändern.

Zudem werden die Stores verwendet, um Funktionen, welche von mehreren Komponenten verwendet werden, an einem zentralen Ort zu sammeln. Somit wird Redundanz vermieden und die Wartung der Software vereinfacht (N-R9). Listing 7 zeigt eine beispielhafte Implementierung eines Pinia-Stores. In der *state* Funktion werden Variablen gesetzt, welche man persistent speichern möchte. In der *getters* Funktion legt man Getter-Funktionen an, um auf die States zuzugreifen. Unter *actions* legt man alle weiteren Funktionen an. Diese Funktionen können global im gesamten Vue.js Projekt verwendet werden.

Insgesamt bietet der Einsatz von Pinia-Stores eine simple Möglichkeit, gemeinsam genutzte Daten sowie zentrale Logik innerhalb der Anwendung konsistent und übersichtlich zu verwalten. Die klare Trennung von Zustand, Zugriff und Logik erhöht dabei nicht nur die Wartbarkeit, sondern verbessert auch die Skalierbarkeit der Anwendung im weiteren Entwicklungsverlauf.

5.4.3 Authentifizierung

Da das MICRO-Backend ursprünglich ausschließlich für die Nutzung in Verbindung mit dem MICRO-Frontend konzipiert wurde, existiert bislang keine tokenbasierte Authentifizierung, wie sie bei vielen modernen APIs üblich ist [56]. Die Authentifizierung der MICRO-Nutzer erfolgt derzeit über das LDAP-System der THM.

Für das HMI wurde übergangsweise ein dedizierter Nutzer in der Datenbank angelegt, der sich mithilfe eines Benutzernamens und Passworts am System authentifiziert und dabei das LDAP-System umgeht. Dieser Ansatz stellt lediglich eine temporäre Lösung dar.

```
export const useStationStore = defineStore('station-store', {
  state: () => ({
    stationsMap: {} as Record<string, Station[]>,
  }),
  getters: {
    getStationsMap: (state) => state.stationsMap,
  },
  actions: {
    async fetchStations(baseRoute: string) {
      const result: RequestResult<Station[]> =
        await getRequest<Station[]>(`/${baseRoute}/stations`);
      if (result.hasFailed) {
        console.error(result.error);
        return;
      }
      this.stationsMap[baseRoute] = result.data;
    },
  },
});
```

Listing 7: Exemplarische Implementation eines Pinia-Stores

Langfristig sollte eine neue, sicherere Authentifizierungsmethode implementiert werden. Gupta et al. [56] untersuchen in diesem Zusammenhang verschiedene Verfahren zur Absicherung von Schnittstellen, die als Grundlage für eine künftige Erweiterung des Systems dienen können, hier aber nicht weiter aufgearbeitet werden.

5.4.4 Anbindung der RGB-Steuerung

Bereits vor Beginn dieser Arbeit existierte in MICRO eine einfache Softwarelösung auf dem Raspberry Pi, über welche sich grundlegende RGB-Effekte am MACRO-Schrank steuern ließen. Diese Umsetzung erforderte jedoch einen manuellen Zugriff auf das System über eine SSH-Verbindung, um ein entsprechendes Skript auszuführen, das dann die Beleuchtungseffekte startete oder änderte.

Im Rahmen dieser Arbeit wurde diese Vorversion grundlegend überarbeitet und in einen modularen *Flask-basierten REST-Service* [100] überführt. Durch diese Umstrukturierung ist es nun möglich, die RGB-LEDs des Schranks über standardisierte HTTP-Anfragen anzusteuern. Zusätzlich wurde das System um mehrere neue Beleuchtungsmodi erweitert. Durch diese Änderungen ist die RGB-Steuerung benutzerfreundlicher

und deutlich flexibler nutzbar. Diese Anbindung ermöglicht sowohl manuelle Eingaben über das HMI als auch perspektivisch automatisierte Abläufe, wie beispielsweise ein System zur automatischen Warnung bei Bewegungen des Portalroboters.

5.5 Aufbau der Build- und Deployment-Pipeline

Zur Sicherstellung einer kontinuierlichen Integration und eines automatisierten Deployments wurde für die Weboberfläche des HMIs eine CI/CD-Pipeline auf Basis von *GitLab CI/CD* [46] implementiert. Ziel war es, jede Änderung am Code automatisch zu überprüfen, zu bauen und, im Falle eines erfolgreichen Merges in den Hauptbranch, bereitzustellen.

5.5.1 GitLab CI/CD Pipeline

Die Pipeline besteht aus drei Hauptphasen:

1. *Linting-Phase:* In dieser Phase wird der Quellcode mithilfe von *ESLint* auf stilistische und syntaktische Fehler überprüft.
2. *Build-Phase:* Für alle Branches außer *main* wird ein Test-Build mit *npm run build* durchgeführt, um sicherzustellen, dass Änderungen kompilierbar sind und keine Buildfehler enthalten.
3. *Deploy-Phase:* Änderungen, die in den *main*-Branch gemergt werden, lösen den Deployment-Prozess aus. Dabei wird ein Docker-Image auf Grundlage der ARM64-Architektur des Raspberry Pis gebaut und in die GitLab-Container-Registry hochgeladen. Zusätzlich zu dem automatischen Deployment-Prozess existiert eine manuelle Variante ohne Cache, um im Fehlerfall Buildprobleme durch veraltete Zwischenspeicher zu vermeiden.

In dieser Pipeline werden sensible Daten wie Zugangsdaten für das Docker-Registry-Login, sowie Build-Parameter wie API-Zugangsdaten über CI/CD-Umgebungsvariablen [47] bereitgestellt. Diese Variablen sind nicht im Repository selbst gespeichert, sondern werden über die GitLab-Oberfläche im Projekt unter *Settings* > *CI/CD* > *Variables* definiert und verwaltet. Sie stehen der Pipeline zur Laufzeit als Umgebungsvariablen zur Verfügung und können sicher im Skript verwendet werden. Auf diese Weise wird verhindert, dass sensible Informationen im Code auftauchen, und gleichzeitig eine flexible Konfiguration der Build- und Deploymentprozesse ermöglicht.

5.5.2 Automatisiertes Deployment via Cronjob

Der tatsächliche Rollout auf das Zielsystem erfolgt über ein Shell-Skript, dargestellt in Listing 8, das lokal auf dem Raspberry Pi durch einen Cronjob alle 60 Sekunden ausgeführt wird. Das Skript prüft, ob eine neue Version des Docker-Images in der Registry verfügbar ist, und aktualisiert anschließend bei einer neuen Version die laufende Anwendung.

```
IMAGE="<REGISTRY-URL>"
ACCESS_TOKEN="<ACCESS_TOKEN>"
CONTAINER_NAME="webapp"

# Login bei GitLab
echo "$ACCESS_TOKEN" | docker login registry.gitlab.com \
    -u "gitlab-ci-token" --password-stdin

# Aktuelles und entferntes Image vergleichen
REMOTE_DIGEST=$(docker pull $IMAGE | grep "Digest:" \
    | awk '{print $2}')
LOCAL_DIGEST=$(docker inspect ...)

if [ "$REMOTE_DIGEST" == "$LOCAL_DIGEST" ]; then
    echo "No updates available."
    exit 0
fi

# Vorherigen Container stoppen und entfernen
docker stop $CONTAINER_NAME && docker rm $CONTAINER_NAME

# Neuen Container mit Umgebungsvariablen starten
source .env
docker run -d --restart always --name $CONTAINER_NAME \
    -e VITE_API_USERNAME="$USERNAME" \
    -e VITE_API_PASSWORD="$PASSWORD" \
    -p 8080:80 $IMAGE
```

Listing 8: Verkürzter Inhalt des deploy.sh Skriptes

Das Vorgehen umfasst:

1. Authentifizierung mit dem GitLab-Container-Registry via Access-Token, welches bei der Einrichtung des HMIs vom Nutzer in GitLab erstellt [48] und im Skript manuell gesetzt werden muss.

2. Vergleich des Image-Digests zwischen Registry und lokal laufendem Container, um festzustellen, ob eine neue Version der Anwendung vorliegt.
3. Falls ein neues Image vorhanden ist:
    1. Stoppen und Entfernen des alten Docker-Containers
    2. Start eines neuen Docker-Containers mit dem aktualisierten Image
    3. Übergabe der Zugangsdaten für die Backend-API via Umgebungsvariablen aus einer *.env*-Datei

Durch diese automatisierte Prüfung wird sichergestellt, dass das System kontinuierlich aktuell bleibt, ohne dass manuelle Eingriffe erforderlich sind. Die Ausführung erfolgt per Cronjob, der wie folgt definiert ist:

```
* * * * * /home/micro/kiosk/deploy.sh >> \
/home/micro/kiosk/deploy.log 2>&1
```

So wird das Skript einmal pro Minute gestartet, wobei die Standard- und Fehlerausgaben in eine Logdatei geschrieben werden.

5.5.3 Vor- und Nachteile der gewählten CI/CD-Strategie

Die eingesetzte CI/CD-Strategie kombiniert zentrale Build- und Testprozesse in GitLab mit einem dezentralen, lokal angestoßenen Deployment-Prozess auf dem Zielsystem. Dieses hybride Vorgehen bringt mehrere *Vorteile* mit sich:

- *Automatisierung*: Alle Schritte von der Codeprüfung über das Erstellen des Docker-Images bis hin zur Bereitstellung auf dem Zielgerät laufen vollständig automatisiert ab.
- *Robustheit durch Digest-Vergleich*: Vor jedem Deployment wird geprüft, ob das aktuelle Image aus der Registry vom lokal Eingesetzten abweicht. Dadurch werden unnötige Neustarts vermieden, was wiederum Ausfallzeiten (N-R8) minimiert.
- *Sicherheitsbewusstes Deployment*: Ein direkter Zugriff von GitLab auf das HMI über das lokale Netzwerk der THM wurde bewusst vermieden, um das HMI-System nicht nach außen zu exponieren. Stattdessen übernimmt ein lokales Skript das Pull-basierte Deployment, wodurch die Sicherheit erhöht und eine externe Erreichbarkeit des Systems nicht notwendig wird.

- *Wartbarkeit und Nachvollziehbarkeit (N-R9)*: Die Trennung in klar definierte Phasen (Build, Test, Deployment) sowie das Logging auf dem Raspberry Pi erleichtern die Fehlersuche bei auftretenden Problemen erheblich.
- *Plattformkompatibilität (N-R3)*: Das CI-System baut das Docker-Image explizit für die Architektur *linux/arm64*, was den reibungslosen Einsatz auf ARM-basierten Geräten wie dem Raspberry Pi gewährleistet und eine einfache Umstrukturierung auf andere Gerätearchitekturen ermöglicht.

Der einzige *Nachteil* dieses Modells ist ein unnötiger Datenverkehr zwischen dem HMI und GitLab. Jede Minute wird eine Anfrage für eine neue Version gesendet, die meist erfolglos ausfällt. Die Alternative dazu wäre ein Deployment von GitLab ausgehend auf das HMI. Dafür müsste jedoch das HMI von außerhalb des Netzwerkes erreichbar sein, oder ein komplexerer Ansatz implementiert werden, welcher weiterhin die Sicherheit des Systems gewährleistet.

Insgesamt bietet diese Struktur eine zuverlässige, sichere und wartungsfreundliche Lösung für das Deployment des HMI-Frontends im MICRO-System, ohne Kompromisse bei Netzwerksicherheit oder Entwicklungsflexibilität eingehen zu müssen. Einzig ein erhöhter Netzverkehr ist der Nachteil dieses Ansatzes.

6 Evaluation

Ziel der durchgeführten Evaluation war es, die Eignung verschiedener Touch-Displays für den Einsatz als Human-Machine-Interface (HMI) im MICRO-System zu bewerten. Dabei standen nicht nur der Vergleich der physischen Eigenschaften und Benutzererfahrungen mit den unterschiedlichen Displays im Vordergrund, sondern auch die Frage, ob die entwickelte HMI-Anwendung im aktuellen Stand bereits den Anforderungen an Bedienbarkeit, Funktionalität und Nutzerfreundlichkeit genügt (vgl. Abschnitt 4.1).

Um beide Aspekte umfassend zu analysieren, wurde ein Fragebogen entwickelt, der verschiedene Perspektiven der Nutzererfahrung erfasst: die Einschätzung durch standardisierte Fragebogenteile wie den UEQ, sowie gezielte Aussagen, Multiple-Choice-Fragen und Freitextfelder für qualitative Rückmeldungen. Die Ergebnisse liefern wertvolle Erkenntnisse über Stärken und Schwächen sowohl der eingesetzten Hardware als auch der HMI-Software und bilden die Grundlage für konkrete Optimierungsmaßnahmen und die zukünftige Entwicklung des HMIs. Die vollständige Auswertung der Daten der Evaluation ist im Anhang (Abschnitt 2) zu finden.

6.1 Teilnehmende

An der Evaluation nahmen insgesamt 14 Studierende aus dem Fachbereich der Informatik der THM teil. Diese Teilnehmergruppe wurde bewusst ausgewählt, da sie über fortgeschrittenes technisches Vorwissen verfügt und die Teilnehmenden im Rahmen des MICRO-Projekts die Rolle zukünftiger MICRO-Administratoren übernehmen könnten.

Die Studierenden repräsentieren somit exakt die primäre Zielgruppe des entwickelten HMI-Systems (vgl. Unterabschnitt 4.1.3) und bilden einen optimalen Rahmen für die Evaluation.

6.2 Methodik

Die Datenerhebung erfolgte durch eine strukturierte Nutzerstudie mit Studierenden der THM als primäre Zielgruppe des HMI-Systems. Die Teilnehmer, welche die Rolle von MICRO-Administratoren repräsentieren, durchliefen einen standardisierten Evaluationsprozess, bestehend aus praktischen Testaufgaben und anschließender Bewertung mittels validierter Fragebögen. Der vollständige Fragebogen ist im Anhang (Abschnitt 1) dokumentiert.

Der Fragebogen konnte in Papierform und als interaktives PDF ausgefüllt werden. Jeder Teilnehmer musste dabei für jeden der 5 evaluierten Displays jeweils einen Fragebogen ausfüllen.

Der Fragebogen beginnt mit einem einleitenden Text, der den Kontext der Evaluation erklärt. Anschließend müssen Teilnehmende einige Aufgaben lösen, um sich mit dem HMI vertraut zu machen und alle Kernfunktionalitäten zu erfahren. Die Aufgaben beinhalten:

1. *RGB-Lichtsteuerung*: Navigation zur Steuerung der RGB-Lichter und Veränderung der Lichtfarbe
2. *Stationsübersicht*: Aufrufen der Stationsübersicht und Ablesen des Reservierungsstatus
3. *Stationsspezifische Reports*: Überprüfung auf vorliegende Reports einer einzelnen Station
4. *Systemweite Reports*: Navigation zur systemweiten Reports-Ansicht

Der darauffolgende Teil des Fragebogens basiert auf bewährten wissenschaftlichen Methoden der UX-Forschung. Als primäres Bewertungsinstrument wurde der User Experience Questionnaire (UEQ) eingesetzt (vgl. Abschnitt 2.4), welcher von Laugwitz et al. entwickelt wurde [69] und eine umfassende Bewertung der Nutzererfahrung durch 26 bipolare Adjektiv-Paare auf einer 7-stufigen Skala ermöglicht. Bei dem in der Evaluation verwendeten UEQ-Bogen handelt es sich um die Kurzversion des UEQ nach Laugwitz et al., die um einige eigene Adjektivpaare ergänzt wurde, um gezielter auf das evaluierte HMI einzugehen.

Ergänzend zum UEQ werden im folgenden Teil Aussagen auf einer 5-Punkte-Likert-Skala (1=„stimme überhaupt nicht zu“ bis 5=„stimme voll zu“) bewertet. Die Gestaltung dieses Fragebogenteils lehnt sich an das bewährte Vorgehen der Evaluationsverfahren

SUS und meCUE an. Diese Aussagen fokussierten sich auf technische Aspekte wie Bildschirmgröße, Textlesbarkeit, Touch-Präzision und Farbdarstellung, um eine detaillierte Bewertung der Hardware-Komponenten zu ermöglichen, stellten aber auch generelle Fragen zur Nutzbarkeit und UX der HMI-Software.

Am Ende des Evaluationsbogens werden zudem Fragen zum allgemeinen Eindruck der Displays sowie zur bevorzugten Touch-Eingabemethode (Finger oder Stift) gestellt. Ergänzend dazu bieten Freitextfelder Raum für Verbesserungsvorschläge, Wünsche und positiv wahrgenommene Aspekte.

Tabelle 6.1: Auflistung evaluierter Touch-Displays für das HMI (Stand: 27.07.2025)

ID	Display	Größe	Auflösung	Panel	Preis
D1	Waveshare 15.6 inch QLED Display [72]	15.6 Zoll	1920×1080	QLED	199,99$
D2	Waveshare 5.5 inch AMOLED [73]	5.5 Zoll	1080×1920	AMOLED	121,99$
D3	Waveshare 9.3 inch LCD Display [75]	9.3 Zoll	1600×600	IPS	99,99$
D4	Raspberry Pi Touch Display 2 [77]	7 Zoll	720×1280	TFT	60$
D5	Waveshare 15.6 inch LCD Display [71]	15.6 Zoll	1920×1080	IPS	166,99$

Zur Vermeidung eines systematischen Bias wurde die Reihenfolge der Displays zwischen den Teilnehmenden variiert. Das jeweils zuerst getestete Display war in sieben Fällen Display 1 und in sieben weiteren Fällen Display 5. Daraufhin bearbeiteten die Teilnehmenden die Displays entweder in aufsteigender oder in absteigender Reihenfolge. Tabelle 6.1 gibt nochmal einen Überblick über die evaluierten Displays. Gegenüber der ursprünglichen Tabelle (Tabelle 5.1) aus Kapitel 4 wurde das Waveshare-Display D6 (welches den Raspberry Pi Zero nutzt) nicht mehr berücksichtigt, da es bereits in Unterunterabschnitt 5.1.1 aus der näheren Auswahl fiel.

Die Displays wurden auf einem selbstgebauten Aufbau, bestehend aus einem Aluminiumprofil und zwei 3D-gedruckten Standfüßen, aufgehängt. Ein Rendering des Aufbaus und ein Bild der tatsächlichen Aufhängung finden sich jeweils in Abbildung 6.1 und Abbildung 6.2.

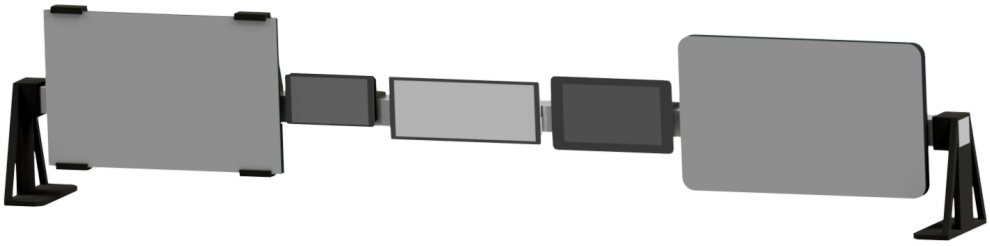


Abbildung 6.1: 3D-Rendering des Evaluationsaufbaus

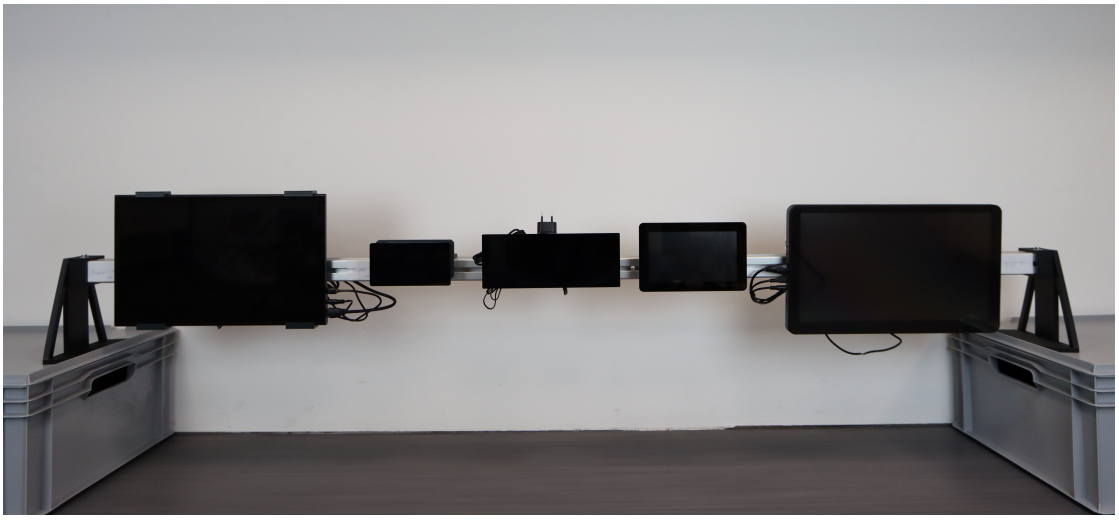


Abbildung 6.2: Bild des tatsächlichen Evaluationsaufbaus

6.3 Ergebnisse

In diesem Kapitel werden die gesammelten Rückmeldungen zusammengefasst dargestellt. Die Ergebnisse sind entsprechend den Fragebogenteilen gegliedert und erlauben eine Betrachtung sowohl der Hardware-Komponenten (Displays) als auch der HMI-Software im Zusammenspiel mit den jeweiligen Endgeräten. Besonderes Augenmerk liegt dabei auf der Benutzerfreundlichkeit, der visuellen Darstellung, der Eingabemethodik sowie auf typischen Problemen und Verbesserungspotenzialen, die von den Teilnehmenden identifiziert wurden.

6.3.1 Teil 1: Aufgaben

Teil 1 handelte von den Aufgaben, welche die Teilnehmenden vor der Bearbeitung des Fragebogens lösen mussten. Hierzu wurden keine Daten erhoben, jedoch wurden, bis auf vereinzelte Ausnahmen, alle Aufgaben von allen Teilnehmenden erfolgreich abgeschlossen. Die einzige Aufgabe, die Probleme hervorgerufen hat, wie auch in einer Frage aus Teil 5 deutlich wird, war die Auswahl der RGB-LEDs (Aufgabe 1). Hier war es oftmals nicht intuitiv ersichtlich, dass man zuerst eine Animation wählen muss, bevor man eine Farbauswahl treffen kann. Viele Teilnehmende gingen davon aus, dass man sofort eine Farbwahl treffen kann, und waren schließlich verwundert, dass die Farbwahl deaktiviert war.

6.3.2 Teil 2: UEQ-Fragebogen

Tabelle 6.2 zeigt die Gesamtwertungen des UEQ-Fragebogenteils. Eine genauere Auflistung der Scores der einzelnen Fragen findet sich in Abschnitt 2 des Anhangs. Die Antworten sind im Zahlenbereich -3 bis +3 angegeben, wobei +3 die bestmögliche und -3 die schlechteste Wertung darstellt. Das Waveshare 15.6″ QLED Display (D1) erreichte mit einem Gesamtscore von 1,54 die höchste Bewertung, gefolgt vom Waveshare 9.3″ IPS Display (D3) mit 1,32 Punkten. Das Waveshare 15.6″ LCD Display (D5) erhielt 1,23 Punkte, während das Raspberry Pi Touch Display 2 (D4) mit 0,94 Punkten und das Waveshare 5.5″ AMOLED Display (D2) mit 0,79 Punkten die niedrigsten Bewertungen erzielten.

Einzelne Adjektivpaare erhielten vergleichsweise niedrige Bewertungen: Das 5.5 Zoll AMOLED Display wurde bei der Dimension „überladen–aufgeräumt“ mit nur 0,57 Punkten bewertet. Das Raspberry Pi Display erhielt bei „konventionell–originell“ einen ne-

Tabelle 6.2: Gesamtwertung (Mittelwert) des UEQ-Fragebogens (Teil 2) absteigend sortiert

Rang	Display	UEQ-Score
0	Waveshare 15.6 inch QLED Display (D1)	1,54
1	Waveshare 9.3 inch Display (D3)	1,31
2	Waveshare 15.6 inch LCD Display (D5)	1,23
3	Raspberry Pi Touch Display 2 (D4)	0,94
4	Waveshare 5.5 inch AMOLED (D2)	0,79

Tabelle 6.3: Gesamtwertung (Mittelwert) der Aussagen-Scores (Teil 3) pro Display absteigend sortiert

Rang	Display	Score
0	Waveshare 15.6 inch QLED Display (D1)	4,45
1	Waveshare 15.6 inch LCD Display (D5)	4,28
2	Waveshare 5.5 inch AMOLED (D2)	4,05
3	Raspberry Pi Touch Display 2 (D4)	4,02
4	Waveshare 9.3 inch Display (D3)	3,99

gativen Wert von -0,07. Die Daten zeigen, dass kleinere Displays von den Teilnehmenden niedriger bewertet wurden als größere Displays.

6.3.3 Teil 3: Aussagen

Tabelle 6.3 zeigt die Gesamtwertungen der Aussagen-Scores. Die Aussagen wurden auf einer 5-Punkte-Likert-Skala (1=„stimme überhaupt nicht zu“ bis 5=„stimme voll zu“) bewertet. Hierbei ist erneut das Waveshare 15.6 Zoll QLED (D1) an oberster Stelle mit einem durchschnittlichen Score von 4,45. Gefolgt von dem anderen 15.6 Zoll Display (D5) zeigt sich auch hier, dass die größeren Displays wieder am besten bewertet wurden. Die kleineren Displays verlagern sich auf die letzten drei Plätze und haben insgesamt eine sehr enge Wertung mit einer Diskrepanz von nur 0,06 Punkten. Die genauen Angaben zu jeder Frage finden sich in Abschnitt 2 des Anhangs.

Das QLED Display (D1) erreichte in den Kategorien Bilddarstellung, Größe und Bedienbarkeit die höchsten Werte. Die Fehlermeldungen wurden mit 3,86 Punkten am schlechtesten bewertet und von den Teilnehmenden als kaum wahrnehmbar beurteilt.

Besonders das 5.5 Zoll Waveshare Display (D2) hat bei der Frage „Ich habe nicht das Gefühl, dass Inhalte zu klein dargestellt sind.“ den geringsten Score aller Displays mit 2,71 Punkten bekommen. In Bezug auf die Bildschärfe erhielt es 4,5 Punkte.

Das 9.3 Zoll Waveshare Display (D3) erreichte hinsichtlich der Displaygröße 3,14 Punkte. Gleichzeitig gaben die Teilnehmenden an, sich gut auf ihre Aufgaben konzentrieren und durch das System navigieren zu können.

Das Raspberry Pi Touch Display 2 (D4) erfuhr eine ähnliche Einschätzung wie (D2) und wurde mit 3,07 Punkten im Vergleich zu den anderen Displays als „zu klein“ wahrgenommen. Für die Bedienbarkeit der Oberfläche wurden hingegen vergleichsweise hohe Werte vergeben.

Das letzte 15.6 Zoll Display (D5) erhielt in den Kategorien Größe und Bedienbarkeit hohe Bewertungen, während die Farbgebung mit 3,71 Punkten geringer bewertet wurde.

6.3.4 Teil 4: Multiple Choice

Die Multiple-Choice-Fragen lieferten wichtige Erkenntnisse zu Nutzerpräferenzen und den Bilddarstellungen der Displays.

Teil 4.1: Präferierte Eingabemethode

Abbildung 6.3 und Abbildung 6.4 zeigen die Auswertungen der Präferenz der Eingabemethode bei Touch-Displays der Teilnehmenden. Die Fingereingabe dominiert deutlich bei allen getesteten Displays. Das Raspberry Pi Touch Display 2 (D4), beide Waveshare 15.6 Zoll Displays (D1 und D5) sowie das Waveshare 9.3 Zoll Display (D3) erhielten jeweils 12 Nennungen für die Finger-Bedienung. Das Waveshare 5.5 Zoll AMOLED Display (D2) erreichte 10 Nennungen für die Fingereingabe.

Die Stift-Eingabe wurde deutlich seltener bevorzugt, mit jeweils nur einer Nennung bei den meisten Displays. Eine Ausnahme bildet das AMOLED Display (D2) mit drei Nennungen für die Stift-Eingabe. Alternative Eingabemethoden wurden durchgängig mit jeweils einer Nennung pro Display angegeben. Hierbei nannte der Teilnehmende eine Maussteuerung als präferierte Eingabemethode.

Die Abbildungen in Abbildung 6.4 visualisieren, wie häufig die verschiedenen Eigenschaften („schneller“, „präziser“, „andere“) den jeweiligen Eingabemethoden und Displays zugeordnet wurden. Da in dieser Frage Mehrfachnennungen zugelassen waren, wurden kombinierte Antworten wie „schneller und präziser“, im Diagramm getrennt gezählt, sodass jede Nennung ihren jeweiligen Eigenschaftsbalken erhöht.

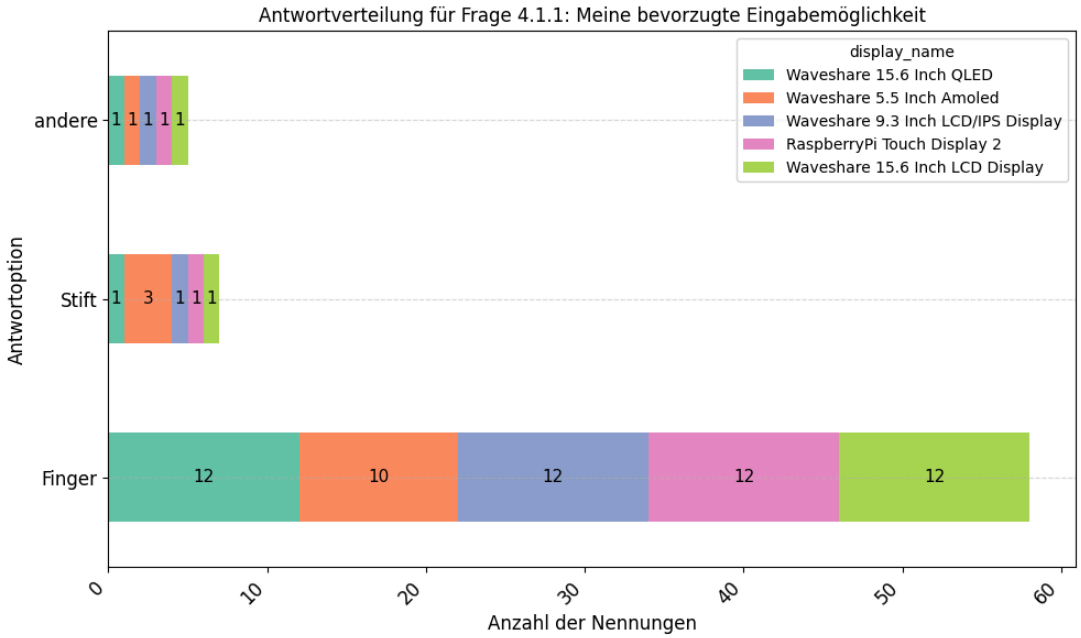


Abbildung 6.3: Ergebnisse von Teil 4.1.1 des Fragebogens

Für die bevorzugte Eingabemethode „Finger“ war die Eigenschaft „schneller“ insgesamt am häufigsten vertreten. Besonders das Waveshare 9.3 Zoll Display (D3) sticht mit 10 Nennungen hervor, dicht gefolgt vom Raspberry Pi Touch Display 2 (D4) sowie von beiden 15.6 Zoll Displays (D1 und D5) mit jeweils 9 und 10. Das AMOLED Display (D2) erreichte sieben Nennungen für „schneller“. Die Eigenschaft „präziser“ wurde für die Fingereingabe meist seltener vergeben, kam aber auf allen Displays mehrfach vor.

Die Stifteingabe wurde ausschließlich mit „präziser“ assoziiert. Hier zeigen sich insgesamt nur wenige Nennungen über alle Displays hinweg, wobei das AMOLED Display (D2) mit drei Stimmen besonders hervorsticht.

Für die Kategorie „andere“ zeigt sich bei allen Displays eine sehr geringe Zahl an Nennungen (jeweils maximal eine). Beispielsweise wurde beim Raspberry Pi Touch Display 2 (D4) unter anderem der Vermerk gemacht, dass bei der Bedienung mit dem Finger ein unwohles Gefühl entstanden sei.

Teil 4.2: Bildschirmhelligkeit

Abbildung 6.5 zeigt die Ergebnisse der Frage über die Bildschirmhelligkeit der einzelnen Displays. Die Bewertung „passend“ dominiert bei der Helligkeitsbewertung. Das

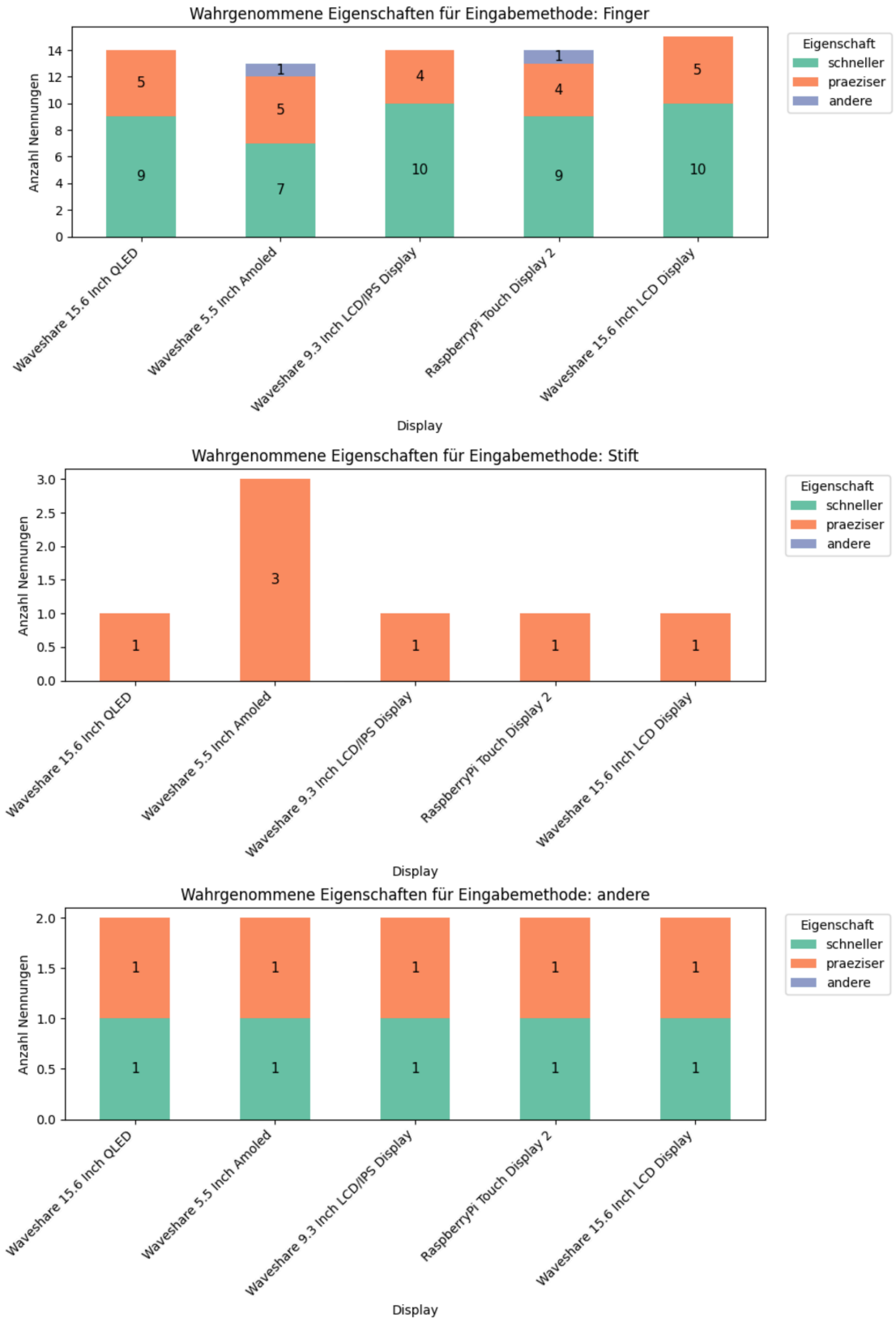


Abbildung 6.4: Ergebnisse von Teil 4.1.2 des Fragebogens

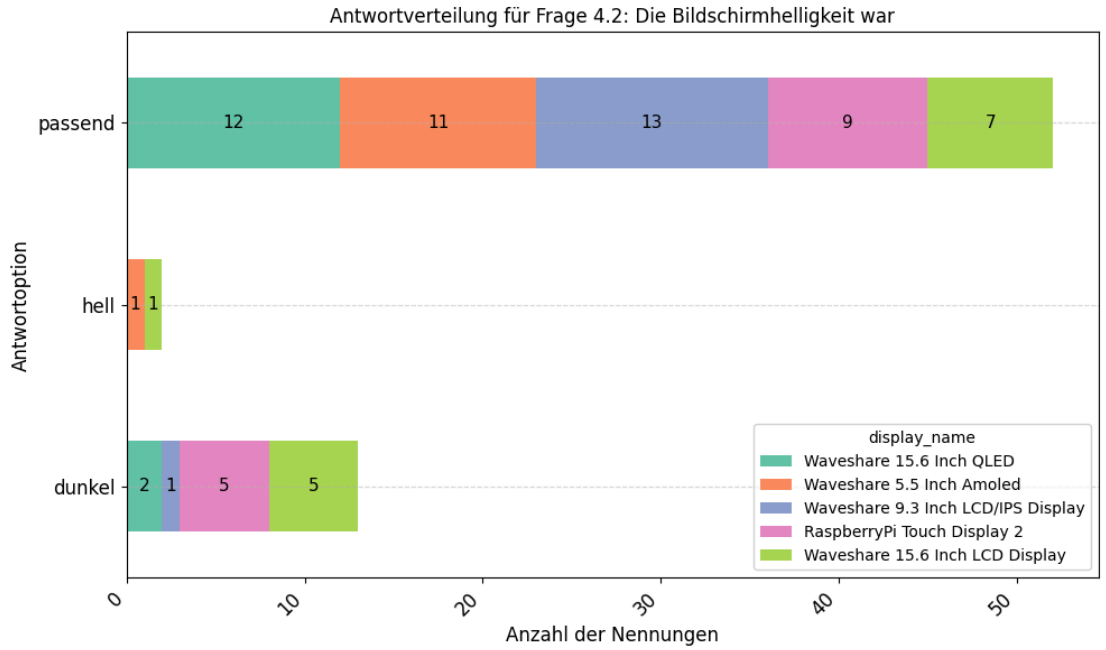


Abbildung 6.5: Ergebnisse von Teil 4.2 des Fragebogens

Waveshare 9.3 Zoll Display (D3) erhielt die höchste Zustimmung mit 13 Nennungen, gefolgt vom QLED Display (D1) mit 12 Nennungen und dem AMOLED Display (D2) mit 11 Nennungen. Das Raspberry Pi Touch Display 2 (D4) erreichte neun Nennungen für „passend“, während das 15.6 Zoll LCD Display (D5) sieben Nennungen erhielt.

Antworten bezüglich zu geringer Helligkeit konzentrierten sich hauptsächlich auf das Raspberry Pi Touch Display 2 (D4) und das Waveshare 15.6 Zoll LCD Display (D5) mit jeweils fünf „zu dunkel“ Nennungen. Das QLED Display erhielt nur zwei entsprechende Kritikpunkte, während die anderen Displays minimal kritisiert wurden.

Die Freitextantworten zeigen, dass das 15.6 Zoll LCD Display (D5) eine blassere Farbdarstellung und einen schlechten Kontrast besitzt. Außerdem gilt dieses, und das Raspberry Pi Touch Display 2 (D4) als deutlich zu verspiegelt und reflektiere zu viel Licht.

Teil 4.3: Bildschirmgröße

Die Größenbewertung in Abbildung 6.6 zeigt deutliche displayspezifische Unterschiede. Das AMOLED 5.5 Zoll Display (D2) erhielt mit 12 Nennungen die meisten Antworten bezüglich zu geringer Größe, gefolgt vom Raspberry Pi Touch Display 2 (D4) mit neun „zu klein“ Nennungen und dem 9.3 Zoll Display (D3) mit sieben entsprechenden Bewertungen. Die größeren 15.6 Zoll Displays erhielten keine „zu klein“ Bewertungen.

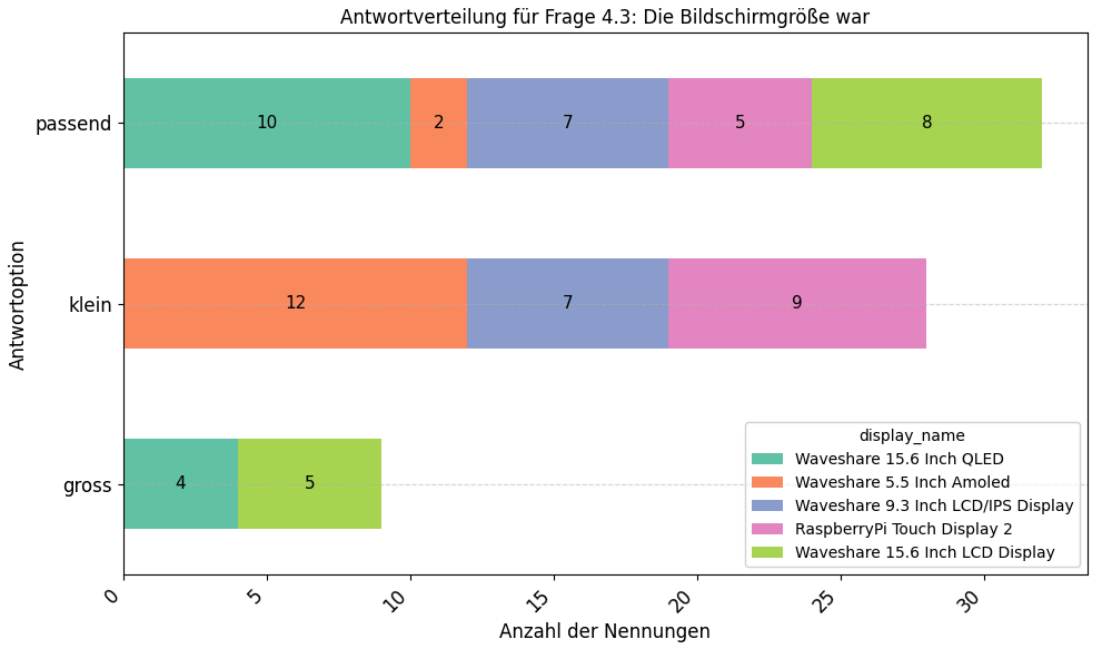


Abbildung 6.6: Ergebnisse von Teil 4.3 des Fragebogens

Bei den „passend“ Bewertungen führt das QLED 15.6 Zoll Display (D1) mit zehn Nennungen, gefolgt vom LCD 15.6 Zoll Display (D5) mit acht Nennungen. Das 9.3 Zoll Display (D3) und das Raspberry Pi Display (D4) erhielten sieben bzw. fünf „passend“ Bewertungen, während das AMOLED Display (D2) nur zwei solche Nennungen erhielt.

„Zu groß“ gelten in 5 bzw. 4 Fällen das 15.6 Zoll LCD Display (D5) und das 15.6 Zoll QLED Display (D1).

Die qualitativen Freitextantworten (vgl. Anhang Abschnitt 2) zur Bildschirmgröße liefern detaillierte Einblicke in die nutzerspezifischen Wahrnehmungen und ergänzen die quantitativen Multiple-Choice-Daten um wichtige kontextuelle Informationen.

Das *Waveshare 15.6 Zoll QLED Display* (D1) wurde hinsichtlich der Integration in das Gesamtsystem beschrieben. Teilnehmende bemerkten, dass das Display zum restlichen MACRO-Schrank passt, die Größe der Bedienelemente als angenehm empfunden wurde und die Textdarstellung als scharf und groß wahrgenommen wurde. Einzelne Kommentare bezogen sich auf Layout und Darstellung: Das Display selbst wirke teilweise etwas groß, die UI-Elemente wurden als relativ groß beschrieben und es wurde angemerkt, dass ein kleineres Display ausreichend gewesen wäre.

Bei dem *Waveshare 5.5 Zoll AMOLED Display* (D2) beschrieben Teilnehmende, dass das Display für montierte Anwendungen klein ist und beim Arbeiten nahe herangetreten

werden muss. Außerdem wurden Interaktionen mit den Bedienelementen als herausfordernd für größere Finger wahrgenommen. Kommentare zur Informationsdichte bezogen sich darauf, dass das Interface auf der Displaygröße überladen wirke und die Textdarstellung, vor allem in der Stationsübersicht, zu klein sei.

Das *Raspberry Pi Touch Display 2* (D4) wurde hauptsächlich für seine ästhetischen und ergonomischen Schwächen kommentiert. Die Ränder wurden als sehr breit wahrgenommen, wodurch der Bildschirm kleiner wirke. Die Textlesbarkeit wurde unterschiedlich beschrieben. Teilnehmende bemerkten, dass Text teilweise klein sei und beim Lesen näher herangegangen werden müsse. Das Display wurde zudem als vergleichbar mit einem kleinen Tablet wahrgenommen.

Das *Waveshare 9.3 Zoll Display* (D3) wurde oftmals in Bezug auf Seitenverhältnis und Informationsdarstellung von den Teilnehmenden bewertet. Teilnehmende äußerten den Wunsch nach mehr vertikal nutzbarer Fläche, um Scrollen zu reduzieren. Die erweiterte Breite wurde im Vergleich zur Höhe als weniger relevant für die Menge an dargestellter Information angegeben. Andere Kommentare bezogen sich positiv auf das Seitenverhältnis und die Größe im Vergleich zu den kleineren Displays (D2).

Das *Waveshare 15.6 Zoll LCD Display* (D5) wurde seltener kommentiert. Anmerkungen bezogen sich auf die Gesamtwirkung mit dem MACRO-Schrank und auf die Nutzung der verfügbaren Bildschirmfläche. Der Bildschirm wurde als schwerfällig wahrgenommen und soll wohl die Größe des Displays nicht optimal nutzen.

Die Freitextantworten verdeutlichen, dass die Bildschirmgröße nicht isoliert, sondern im Kontext der Anwendung, der Montage und der ergonomischen Anforderungen bewertet wird. Besonders hervorzuheben ist die wiederkehrende Kritik an der mangelnden Optimierung des UIs für verschiedene Displaygrößen und Seitenverhältnisse.

Teil 4.4: Informationsmenge

Abbildung 6.7 zeigt die Auswertung der Datenerhebung über eine Informationsdichte, und somit eine kognitive Belastung, für jedes Display. Die Mehrheit der Teilnehmer fühlte sich bei keinem Display mit der Informationsmenge überfordert. Die „Nein“ Antworten dominieren bei allen Displays.

Überforderungserfahrungen wurden am häufigsten mit fünf „Ja“ Nennungen beim AMOLED 5.5 Zoll Display (D2) berichtet. Das Raspberry Pi Touch Display 2 (D4) erhielt drei entsprechende Nennungen, während die anderen Displays jeweils nur zwei Überforderungsberichte erhielten.

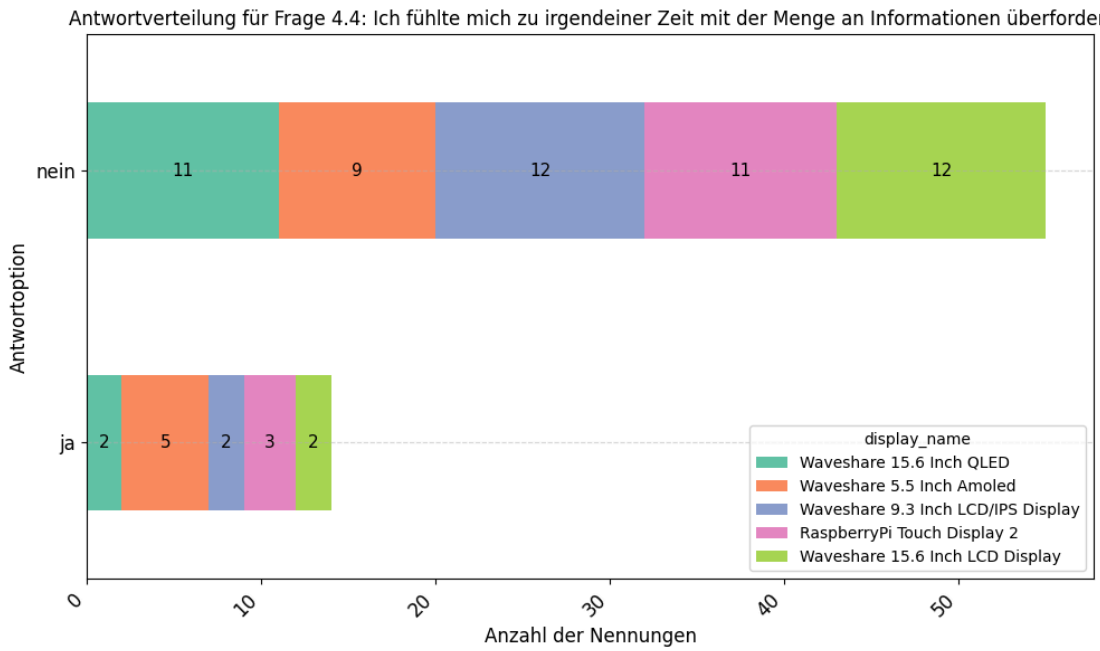


Abbildung 6.7: Ergebnisse von Teil 4.4 des Fragebogens

Die qualitativen Rückmeldungen zur wahrgenommenen Informationsüberforderung konzentrieren sich auf spezifische Interface-Bereiche und displayabhängige Darstellungsprobleme, wobei die Stationsübersicht als primärer Problembereich identifiziert wurde und dort das Hauptproblem die zu hohe Informationsdichte ist.

6.3.5 Teil 5: Freitextantworten

Basierend auf der Auswertung der Evaluationsdaten können die Freitextantworten zu den verschiedenen Displays systematisch nach den wichtigsten Erkenntnissen strukturiert werden. Die Teilnehmer bewerteten fünf verschiedene Displays anhand von sechs offenen Fragen, wobei insgesamt 145 Freitextantworten gesammelt wurden.

Waveshare 15.6 Zoll QLED Display (D1)

Das Waveshare 15.6 Zoll QLED (D1) wurde von Teilnehmenden als *intuitiv bedienbar* beschrieben, mit einer *klaren Struktur* der Benutzeroberfläche. Das Panel wurde hinsichtlich *Farbdarstellung* und *Lesbarkeit* positiv wahrgenommen und als modern wirkend

beschrieben. Die *Displaygröße* wurde als passend für die dargestellten Inhalte eingeschätzt, und die *Touch-Responsivität* wurde als präzise bewertet, ohne dass Artefakte auftraten.

Gleichzeitig wurde das Display von einzelnen Teilnehmenden als *für bestimmte Aufgaben zu groß* eingeschätzt. Die *Montagemöglichkeiten* wurden als eingeschränkt wahrgenommen, wobei insbesondere eine Möglichkeit zur *Neigung des Displays* als wünschenswert beschrieben wurde.

Auf Basis der Rückmeldungen wurde vorgeschlagen, die *Anpassung des Neigungswinkels* für eine ergonomischere Nutzung zu ermöglichen.

Waveshare 5.5 Zoll AMOLED Display (D2)

Das Waveshare 5.5 Zoll AMOLED Display (D2) war das kleinste getestete Display. Teilnehmende beschreiben die *Farbqualität* als auffällig und *deutlich differenzierter im Vergleich zu den anderen Displays*. Die *kompakte Bauform* fördere die schnelle Erfassung von Informationen. Zudem wurden *scharfe Texte* und eine *klare Bilddarstellung* festgestellt.

Gleichzeitig wurde die Displaygröße als *für montierte Anwendungen zu klein* beschrieben. Teilnehmende gaben an, dass die Nutzung für Alltagsanwendungen eingeschränkt sei. Ergonomische Aspekte wurden hervorgehoben: Das *Display erwärmte sich während der Nutzung* und die *Textdarstellung wurde als klein oder teilweise abgeschnitten* wahrgenommen, was die Lesbarkeit und Bedienung beeinflusste.

Waveshare 9.3 Zoll LCD Display (D3)

Das Waveshare 9,3 Zoll Display (D3) wurde als Display mit einem Ultrawide-Format getestet. Teilnehmende beschrieben das breite Seitenverhältnis als *andersartig* und *potenziell vorteilhaft* für die Darstellung von Texten, die als *angenehm lesbar* wahrgenommen wurden. Das Display wurde als Kompromiss zwischen Größe und Funktionalität eingeordnet.

Gleichzeitig wurde angemerkt, dass das Interface das breite Format nicht vollständig nutzte, wodurch Fläche ungenutzt blieb. Weitere Hinweise betrafen die *Farbdarstellung*, insbesondere einen Blaustich und eingeschränkte Blickwinkelstabilität. Die *vertikale Fläche* wurde als begrenzt beschrieben, wodurch mehr vertikaler Platz für die Nutzung wünschenswert gewesen wäre.

Raspberry Pi Touch Display 2 (D4)

Das Raspberry Pi Touch Display 2 (D4) wurde als Standard-Display eingesetzt. Teilnehmende gaben an, dass die *Größe* an ein kleines Tablet erinnere und ein vertrautes Nutzungserlebnis vermittle. Die *Farbdarstellung* wurde als natürlich beschrieben, und die Glasoberfläche erzeuge ein *gutes haptisches Gefühl* und wirke *qualitativ hochwertig*.

Gleichzeitig wurden die *breiten Displayränder* und die Sichtbarkeit der *Pixelstruktur* als Einschränkungen bei der Nutzung genannt. Spiegelungen bei unterschiedlichen Blickwinkeln wurden ebenfalls beobachtet. Für produktive Anwendungen wurde die *Displaygröße* als begrenzt eingeschätzt, während das Display alternativ auch für ein reines Statusanzeigendisplay genutzt werden könne.

Waveshare 15.6 Inch LCD Display (D5)

Das Waveshare 15,6 Zoll LCD Display (D5) besitze eine *große Darstellungsfläche*, die für umfangreichere Inhalte genutzt werden könne. Teilnehmende beschrieben die Bedienung als *flüssig* und verwiesen auf eine *professionelle Optik*.

Weitere Kommentare betrafen die *Optik* und *Farbdarstellung*, die als kontrastarm wahrgenommen wurde. Die *Lücke zwischen Touchoberfläche und Display* beeinflusse die Interaktion. Die Bauweise wurde als dick beschrieben, wodurch die Haptik und der Gesamteindruck vom Display beeinflusst wurde und als *nicht zeitgemäß* beschrieben wurde.

HMI-Anwendung und allgemeine Vorschläge

Die Freitextantworten der Evaluation lieferten wertvolle Hinweise zur Benutzerfreundlichkeit und zu Verbesserungspotentialen der entwickelten HMI-Anwendung. Die häufigsten Rückmeldungen der Antworten bezogen sich auf die RGB-LED-Steuerung sowie die Navigation.

Die **Steuerung der RGB-LEDs** wurde von vielen Teilnehmenden als zentrale Herausforderung in der Nutzung beschrieben. Vor der eigentlichen Farbauswahl musste zunächst eine Animation ausgewählt werden, was in den Beobachtungen häufig nicht erfolgte. Zusätzlich wurde angegeben, dass Begriffe wie „Color Selection unavailable“

unklar formuliert wurden und nicht selbsterklärend waren. Teilweise traten auch technische Schwierigkeiten bei der Nutzung der RGB-LED-Steuerung auf. Diese waren jedoch auf den RGB-Steuerungs-Raspberry Pi zurückzuführen und wurden daher in der Evaluation nicht weiter berücksichtigt.

Auch die **Navigation** innerhalb der Benutzeroberfläche erhielt wiederholt Hinweise auf Optimierungsmöglichkeiten. Insbesondere die Sidebar-Icons wurden als unverständlich empfunden. Viele Nutzende verwendeten stattdessen wiederholt den Umweg über den Home-Button. Eine klare Kontextanzeige fehle, wodurch die Orientierung innerhalb des Systems erschwert worden sei. Auch kam es zu Verwechslungen zwischen dem Login-Button und dem Zurück-Button. Zusätzlich wurde kritisiert, dass an manchen Stellen die Möglichkeit fehle, zu vorherigen Seiten zurückzukehren, was zu einem Gefühl von Kontrollverlust führe. Dafür wurden auch Lösungen genannt, etwa das Hinzufügen der Icons auch auf dem Startbildschirm, um eine Verbindung zwischen der Sidebar und den Buttons auf der Startseite zu schaffen.

Die zentrale **Verwaltungsansicht der Stationen** bekam ebenfalls einige Kommentare. Besonders häufig wurde der Wunsch nach einer deutlicheren Statusanzeige geäußert, etwa durch farbliche Kennzeichnungen, die auf Reservierungszustände hinweisen. Weitere Vorschläge beinhalteten Benachrichtigungs-Badges für neue Reports, eine klarere Identifikation der Stationen (z. B. durch Voranstellen der ID) sowie eine Icon-basierte Darstellung von Zustandsinformationen.

Im **Reporting-System** wurden verschiedene Punkte thematisiert. So können systemweite Reports bislang nicht eindeutig einer Station zugeordnet werden, was die Nachverfolgbarkeit einschränke. Bei fehlenden Reports erscheinen leere Tabellen mit unbeschrifteten Headern, was für zusätzliche Verwirrung sorge. Zudem war es nicht möglich, Reports als „bearbeitet“ zu markieren, sie konnten lediglich gelöscht werden. Ein weiterer Verbesserungsvorschlag betraf die Verlinkung: Eine direkte Navigation von einem Report auf der systemweiten Reportansicht zur betroffenen Station wäre aus Sicht der Teilnehmenden wünschenswert.

Auf **gestalterischer Ebene** wurden mehrere Aspekte angesprochen, die zu einer eingeschränkten Nutzbarkeit führten. Dazu zählen insbesondere Kontrastprobleme, beispielsweise weiße Schrift auf farbigen Buttons, die schwer lesbar gewesen sei. Zudem fehlen Skalierungsmöglichkeiten, etwa durch Touch-Zoom oder automatische Schriftgrößenanpassung. Teilweise wurde auch von einer Informationsflut gesprochen, wenn zu viele Inhalte gleichzeitig dargestellt wurden. Das Layout passe sich nicht immer optimal an die jeweilige Bildschirmbreite an, was das Nutzungserlebnis mindere.

Dennoch wurde die Oberfläche insgesamt als „sehr übersichtlich“ beschrieben, insbesondere im Hinblick auf die Strukturierung der Hauptfunktionen. Die Darstellung von Pop-up-Benachrichtigungen wurde als angemessen prominent empfunden. Auch die funktionale Grundstruktur überzeuge.

6.4 Diskussion der Ergebnisse

Auf Grundlage der im Rahmen der Evaluation erhobenen Daten lassen sich nun Rückschlüsse auf die Eignung der getesteten Displays und die entwickelte HMI-Anwendung ziehen. Dabei ist es möglich, die Stärken und Schwächen der einzelnen Varianten in Relation zu den Anforderungen des Systems einzuordnen und ihre jeweilige Rolle für den praktischen Einsatz zu bewerten.

6.4.1 Gesamtbeurteilung der Displays

Die systematische Evaluation der fünf Touch-Displays lieferte klare Erkenntnisse über deren Eignung für den Einsatz im Remote-Labor-Kontext. Die Bewertungen zeigen deutliche Unterschiede in Benutzerakzeptanz, technischer Qualität und praktischer Anwendbarkeit.

Optimale Displaylösung

Das Waveshare 15.6 Zoll QLED Display (D1) erweist sich als optimale Wahl für das HMI. Mit einem UEQ-Score von 1,54 und einem Aussagen-Score von 4,45 führt es beide Bewertungskategorien an. Die herausragenden Eigenschaften umfassen:

- *Technische Exzellenz*: QLED-Technologie mit sauberer Farbdarstellung und scharfer Bildqualität.
- *Optimale Größe*: 15.6 Zoll bieten ausreichend Platz für alle HMI-Inhalte ohne Überforderung.
- *Touch-Responsivität*: Präzise Bedienung ohne Artefakte.
- *Intuitive Bedienbarkeit*: Klare Struktur und moderne Optik unterstützen die Benutzerführung.

Verbesserungsbedarf: Die Montage-Optionen sollten um Anwinkelungsmöglichkeiten erweitert werden. Vereinzelt wurde das Display als zu groß empfunden, was jedoch durch die gute Informationsdarstellung kompensiert wird.

6.4.2 Alternative Displaylösung

Das 9.3 Zoll Ultrawide-Display (D3) belegt mit einem UEQ-Score von 1,32 den zweiten Platz und bietet interessante Eigenschaften für spezialisierte Anwendungen:

- *Innovatives Format*: Das Ultrawide-Seitenverhältnis ermöglicht verbesserte Textlesbarkeit und wirkt interessant und neu.
- *Kompakter Formfaktor*: Guter Kompromiss zwischen Funktionalität und Platzbedarf.
- *Zukunftspotential*: Das breite Format könnte bei optimierter Software-Anpassung Vorteile bieten.

Verbesserungsbedarf: Die HMI-Software nutzt das breite Format nicht optimal aus. Farbdarstellung mit Blaustich und begrenzte Blickwinkelstabilität reduzieren die Gebrauchstauglichkeit. Sollte man die HMI-Anwendung an das Seitenverhältnis anpassen, könnte man eine erneute Evaluation zwischen diesem Display und D1 durchführen. Es wäre auch denkbar, noch ein weiteres Display mit einem breiten Seitenverhältnis wie dieses zu testen, welches jedoch etwas größer ist, um so auch von der gesteigerten Vertikalen dieses Displays zu profitieren, was ein großer Kritikpunkt an D3 war.

Displays mit geringer Empfehlung

Die drei weiteren Displays zeigen signifikante Einschränkungen für den produktiven Einsatz:

Waveshare 15.6 Zoll LCD Display (D5): Trotz guter Größe (UEQ: 1,23; Aussagen: 4,28) disqualifizieren veraltete Optik, schlechte Farbdarstellung und sichtbare Lücken zwischen Touch- und Displayebene das Gerät für moderne HMI-Anwendungen. D1 bietet dieselbe Größe bei besserer Bildqualität. Selbst bei den Kosten ist nur ein geringer Unterschied sichtbar (etwa 30$), wobei der qualitative Unterschied recht groß wahrgenommen wurde.

Raspberry Pi Touch Display 2 (D4): Das Standard-Display (UEQ: 0,94, Aussagen: 4,02) eignet sich primär nur als einfache Statusanzeige. Massive Displayränder, sichtbare

Pixelstruktur und zu geringe Größe machen es für administrative Aufgaben ungeeignet. Zudem wurde das Display durchwegs als „zu verspiegelt“ betitelt, was es für den Betrieb im Labor nur bedingt eignet.

Waveshare 5.5 Zoll AMOLED Display (D2): Trotz hervorragender Farbqualität ist es mit dem niedrigsten UEQ-Score (UEQ: 0,79; Aussagen: 4,05) für die Anforderungen unbrauchbar. Die geringe Größe führt zu Ergonomie-Problemen und Inhaltsverlust. Das Display könnte jedoch Anwendung als Statusdisplay an einem Arbeitsplatz finden. Dieser Nutzen müsste jedoch weitergehend evaluiert werden.

Zusammenfassung

Für die Implementierung im MICRO Remote-Labor wird das Waveshare 15.6 Zoll QLED Display (D1) eindeutig als die beste Wahl empfunden. Es bietet die beste Balance aus technischer Qualität, Benutzerfreundlichkeit und Zukunftssicherheit. Die Investition in die QLED-Technologie rechtfertigt sich durch deutlich bessere Nutzererfahrung, professionelles Erscheinungsbild und einer guten Helligkeit, was vor allem im gut beleuchteten Labor von Vorteil sein sollte. Alternativ wäre es auch denkbar, ein Display in der Größenordnung von etwa 12 Zoll zu evaluieren. Dieser Vorschlag wurde ebenfalls von Teilnehmenden gemacht, da der Sprung von 9.3 Zoll auf 15.6 Zoll doch recht groß sei.

Das 9.3 Zoll Ultrawide-Display (D3) sollte für zukünftige Entwicklungen im Auge behalten werden. Sobald die Software für das breite Format optimiert wurde, könnte dieses, oder ein weiteres Display in diesem Format, sich als bessere Lösung als D1 herausstellen.

6.4.3 Beurteilung der optimalen Eingabemethode

Die Ergebnisse der Auswertung zeigen deutlich, dass der Finger als Eingabemethode von den Teilnehmenden insgesamt am besten bewertet wurde und sich damit als bevorzugte Wahl für die Bedienung des HMIs herausstellt. Insbesondere in der Dimension „schneller“ erhält die Fingereingabe bei allen Displays die mit Abstand höchste Anzahl an positiven Nennungen. Auch wenn die Eigenschaft „präziser“ häufiger der Stifteingabe zugeordnet wurde, ist ihre absolute Häufigkeit gering und die Stifteingabe insgesamt wenig verbreitet. Die Eingabemethode „andere“ spielte praktisch keine Rolle und wurde nur vereinzelt als Alternative zum Finger genannt, ohne dass damit konsistent positive Eigenschaften verbunden waren.

Entscheidend für den Praxiseinsatz ist zudem, dass die Fingereingabe auf allen getesteten Displaygrößen als unkompliziert, direkt und effizient wahrgenommen wurde. Diese breite Akzeptanz macht sie zur robustesten und alltagstauglichsten Lösung für ein Touch-basiertes HMI. Auch im Hinblick auf die Anforderungen an Usability und einen möglichst niedrigen Lern- und Schulungsaufwand ist die Fingereingabe den alternativen Methoden überlegen. Sie bietet eine schnelle, intuitive Interaktion und benötigt keine zusätzlichen Werkzeuge oder Peripheriegeräte. Damit ist der Finger eindeutig als optimale Eingabemöglichkeit für das betrachtete Anwendungsszenario zu empfehlen.

6.4.4 Gesamtbeurteilung der HMI-Anwendung

Die Evaluation der HMI-Anwendung offenbarte sowohl Stärken als auch konkrete Verbesserungsbedarfe.

Stärken der aktuellen Implementierung

Die Grundarchitektur der HMI-Anwendung überzeugt durch mehrere positive Aspekte:

- *Funktionale Vollständigkeit*: Alle definierten Hauptaufgaben konnten erfolgreich ausgeführt werden.
- *Übersichtliche Struktur*: Das System wurde als übersichtlich bewertet.
- *Angemessene Benachrichtigungen*: Pop-up-Darstellungen wurden als passend prominent wahrgenommen.
- *Responsive Touch-Bedienung*: Grundlegende Touch-Interaktionen funktionieren zuverlässig.

Verbesserungsbereiche

Die **RGB-Steuerung** erfordert dringende Überarbeitung:

- *Animations-Auswahl*: Direktzugriff auf Farbauswahl ohne Animations-Vorauswahl implementieren.

- *Klarere Statusanzeigen*: „Color Selection unavailable“ durch bessere Erklärung ersetzen wie „Color Selection unavailable for this Animation“.
- *Verbesserte Fehlermeldungen*: Spezifische, handlungsorientierte Rückmeldungen bei Problemen.

Die **Benutzerführung** benötigt fundamentale Verbesserungen:

- *Icon-Beschriftung*: Tooltips oder permanente Labels für Sidebar-Elemente.
- *Kontextanzeige*: Breadcrumb-Navigation oder Seitentitel zur Orientierung.
- *Konsistente Navigation*: Die Touch-Funktionalitäten des HMI bieten die Möglichkeit, durch eine Wischgeste auf die vorherige Seite zurückzugehen und somit einen „Zurückknopf“ im ganzen System. Auf diese Funktionalität sollte man MICRO-Administratoren schulen.
- *Visuelle Verbindungen*: Icons der Sidebar auch auf der Startseite anzeigen, um dem Nutzer die Verbindung zwischen diesen klarzumachen.

Die **Stationsverwaltung** bietet Optimierungspotential:

- *Farbkodierung*: Visuelle Statusanzeigen für Reservierungs- und Betriebszustände, entweder farblich oder als deutliche Icons.
- *Benachrichtigungs-Badges*: Sofortige Erkennbarkeit von neuen Reports.

Das **systemweite Nutzermeldungs-System** erfordert funktionale Erweiterungen:

- *Zuordnungsklarheit*: Station-Zugehörigkeit in allen Report-Ansichten anzeigen.
- *Statusverwaltung*: „Bearbeitet“-Markierung zusätzlich zur Löschfunktion (Dieses muss auch im Springbackend implementiert werden).
- *Direkte Navigation*: Verlinkung von Reports der systemweiten Ansicht zu betroffenen Stationen.
- *Leere-Zustand-Behandlung*: Informative Meldungen statt leerer Tabellen.

Gestalterische Verbesserungen für bessere Zugänglichkeit:

- *Kontrastoptimierung*: Lesbarkeit bei allen Farbkombinationen sicherstellen.
- *Responsive Skalierung*: Touch-Zoom oder automatische Größenanpassung bei kleineren Displays.

Basierend auf den Ergebnissen der Evaluation lassen sich mehrere strategische Ziele ableiten, die für die Weiterentwicklung des HMI-Systems verfolgt werden sollten. Ein zentrales Ziel ist die Optimierung des *Responsive Designs*, um eine konsistente und benutzerfreundliche Darstellung auf unterschiedlichen Displaygrößen und -formaten sicherzustellen. Darüber hinaus sollte die Software weiterhin *modular erweiterbar gestaltet* werden, um künftige Laborkomponenten wie etwa eine erweiterte Robotersteuerung problemlos integrieren zu können. Zur Unterstützung der Nutzerinnen und Nutzer ist der Aufbau einer *Benutzerdokumentation*, beispielsweise in Form eines Wikis oder eines integrierten Hilfe-Systems, empfehlenswert, insbesondere für komplexere Funktionen. Schließlich sollten in regelmäßigen Abständen *Usability-Tests* mit tatsächlichen MICRO-Administratoren durchgeführt werden, um eine kontinuierliche Verbesserung der Benutzerfreundlichkeit im praktischen Einsatz sicherzustellen.

6.5 Fazit

Die systematische Evaluation des entwickelten HMI-Systems lieferte fundierte Erkenntnisse über die praktische Tauglichkeit sowohl der Hardware-Komponenten, als auch der Software-Implementierung. Mit 14 Teilnehmerinnen und Teilnehmern aus der Zielgruppe potenzieller MICRO-Administratoren konnte eine Bewertung der verschiedenen Display-Technologien und der Benutzerschnittstelle durchgeführt werden.

6.5.1 Erfüllung der Anforderungen

Die Evaluation gibt zudem weitere Aufschlüsse über die Umsetzung der Anforderungen aus Abschnitt 4.1.

Funktionale Anforderungen

Die Evaluation bestätigt die Umsetzung aller definierten funktionalen Anforderungen:

- *F-R1 - Stationsstatus-Anzeige*: Die Stationsübersicht stellt Status, Reservierungszustand und Verfügbarkeit aller MICRO-Stationen übersichtlich dar. Teilnehmende bewerteten die Informationsdarstellung als klar verständlich.
- *F-R2 - Nutzermeldungen*: Das System zeigt sowohl stationsspezifische als auch systemweite Reports an und ermöglicht deren Bearbeitung durch Löschfunktion. Verbesserungspotenzial besteht in der Zuordnungsklarheit und Statusverwaltung.

- *F-R3 - RGB-Steuerung*: Die RGB-Beleuchtung ist technisch vollständig steuerbar, jedoch erwies sich die Bedienführung als größtes UX-Problem der Evaluation.
- *F-R4 - Sicherheitsmechanismen*: Bestätigungsdialoge bei kritischen Aktionen (z. B. Report-Löschung) sind implementiert und wurden in der Evaluation erfolgreich getestet.

Nicht-funktionale Anforderungen

Auch die nicht-funktionalen Anforderungen wurden weitgehend erfolgreich umgesetzt:

- *N-R1 - Raspberry Pi Hardware*: Das System läuft stabil auf Raspberry Pi 5 mit verschiedenen Touch-Displays. Die Evaluation bestätigte die Eignung der Hardware-Plattform.
- *N-R2 - Responsive Touch-Design*: Mobile-First-Prinzipien wurden konsequent umgesetzt. Touch-Responsivität wurde mit 4,0-4,86 Punkten bewertet, Fat-Finger- und Midas-Touch-Probleme erfolgreich adressiert.
- *N-R3 - Portabilität*: Die webbasierte Vue.js-Architektur ermöglicht problemlose Portierung auf andere Plattformen und mobile Geräte ohne Code-Anpassungen.
- *N-R4 - THM Design-Compliance*: Farbschema, Typografie und Layout entsprechen THM Corporate Design und harmonieren mit dem bestehenden MICRO-System.
- *N-R5 - Intuitive Bedienbarkeit*: Das System wurde als „sehr übersichtlich“ bewertet, jedoch beeinträchtigen unklare Sidebar-Icons und RGB-Steuerung die Intuitivität.
- *N-R6 - Touch-optimierte Bedienelemente*: Vuetify-Komponenten gewährleisten Touch-gerechte Elementgrößen. Teilnehmende bewerteten die Touch-Präzision durchweg positiv (4,0-4,86 Punkte).
- *N-R7 - Flache Navigationsstruktur*: Alle Funktionen sind innerhalb von maximal zwei Interaktionen erreichbar. Die Startseite bietet direkten Zugang zu allen Hauptfunktionen.
- *N-R8 - Performance und Zuverlässigkeit*: Teilnehmende bewerteten Reaktionszeiten und Systemstabilität sehr positiv (4,14-4,71 Punkte). Animationen und Übergänge wurden als flüssig wahrgenommen.

- *N-R9* - *Wartbarkeit*: Moderne Vue.js-Architektur, automatisierte CI/CD-Pipeline und Docker-Containerisierung gewährleisten hohe Wartbarkeit. Eine modulare Komponentenstruktur erleichtert die Anpassbarkeit.
- *N-R10* - *Erweiterbarkeit*: Tab-basierte Stationsübersicht, modulares Vue-Router-System der Startseite und Navigationsleiste sowie RESTful-API-Design ermöglichen einfache Integration neuer Funktionen.

6.5.2 Display-Hardware

Die Evaluation bestätigt eindeutig die Überlegenheit des *Waveshare 15.6 Zoll QLED Displays* (D1) als optimale Lösung für Remote-Labor-HMIs. Mit dem höchsten UEQ-Score von 1,54 und einem Aussagen-Score von 4,45 setzt es sich klar von den Alternativen ab. Die QLED-Technologie bietet nicht nur technische Exzellenz in Farbdarstellung, Bildschärfe, und Helligkeit, sondern auch eine professionelle Ausstrahlung.

Besonders bemerkenswert ist die deutliche Korrelation zwischen Displaygröße und Benutzerakzeptanz: Displays unter 10 Zoll erwiesen sich durchweg als ungeeignet für administrative Aufgaben, während die 15.6 Zoll Displays die besten Bewertungen erhielten. Dies unterstreicht die Bedeutung ausreichender Darstellungsfläche für komplexe HMI-Anwendungen.

Das *9.3 Zoll Ultrawide-Display* (D3) zeigt mit seinem zweiten Platz (UEQ: 1,32) interessantes Zukunftspotential, erfordert jedoch eine grundlegende Überarbeitung der Software-Architektur zur optimalen Nutzung des außergewöhnlichen Seitenverhältnisses.

Software-Usability

Die HMI-Software demonstriert eine solide Grundarchitektur mit erfolgreicher Umsetzung aller Kernfunktionalitäten. Alle Teilnehmenden konnten die gestellten Aufgaben bewältigen, was die prinzipielle Gebrauchstauglichkeit bestätigt.

Jedoch identifizierte die Evaluation einige kritische Schwachstellen:

1. *RGB-Steuerung*: Die unintuitive Bedienführung mit notwendiger Animationsauswahl erwies sich als größtes Usability-Hindernis.
2. *Navigation*: Unverständliche Sidebar-Icons und fehlende Orientierungshilfen beeinträchtigen die Benutzerführung erheblich.

3. *Responsivität*: Teilweise unzureichende Nutzung von Whitespace.

6.5.3 Methodische Reflexion

Die gewählte Evaluationsmethodik mit kombinierten quantitativen (UEQ und Likert-Skalen) und qualitativen (Freitextantworten) Instrumenten erwies sich als effektiv für die umfassende Bewertung des HMI-Systems. Einzig war der Aufbau des Fragebogens unzureichend, sodass Teilnehmende oftmals Fragen, welche nur die Software betreffen, mehrfach beantworten mussten, was zu Frustration und einer langen Evaluationsdauer (insgesamt etwa eine Stunde) führte. Zukünftige Evaluationen sollten auf eine bessere Trennung zwischen Soft- und Hardwarefragen achten, um so unnötige Mehrfachbeantwortungen einiger Fragen zu vermeiden.

Zusätzlich ist die statistische Aussagekraft der erhobenen Daten zu diskutieren. Die Auswertung der numerischen Daten erfolgte ohne Normierung, sodass bestimmte Biases, wie etwa eine tendenziell generell sehr gute oder sehr schlechte Bewertung durch einzelne Nutzer, die Ergebnisse beeinflussen könnten. Da ein solcher Bias jedoch bei allen bewerteten Displays gleichermaßen auftritt, sollte er beim Vergleich der Displays untereinander weitgehend neutralisiert werden. Die Displays lassen sich somit weiterhin sinnvoll in ein Ranking einordnen.

Darüber hinaus wurde aus Zeitgründen bei einigen Evaluationen die Datenerhebung von mehreren Personen gleichzeitig am Evaluationsaufbau durchgeführt. Dabei wurde darauf geachtet, dass jede Person an einem unterschiedlichen Display arbeitete. Eine gegenseitige Beeinflussung der Teilnehmer kann jedoch nicht vollständig ausgeschlossen werden. Da sich eine solche Beeinflussung jedoch überwiegend darin äußert, dass Nutzer auf mögliche Missstände aufmerksam werden, und die Teilnehmer weiterhin nachvollziehbare und begründete Bewertungen abgeben konnten, wird auch dieser Bias als vernachlässigbar eingestuft. Dennoch sollte bei zukünftigen Evaluationen darauf geachtet werden, Beeinflussungen durch andere Personen sowie mögliche Ablenkungen weitestgehend zu vermeiden, um selbst minimale Biases zu reduzieren.

Der Umfang der durchgeführten Befragung beläuft sich auf 14 Teilnehmende und liegt damit innerhalb der wissenschaftlich empfohlenen Bandbreite für explorative Nutzerstudien im Bereich User Experience (UX). Gerade bei studentischen Arbeiten und praktischen Evaluationen von Prototypen wird in der Fachliteratur häufig ein Stichprobenumfang zwischen 10 und 20 Personen als ausreichend angesehen [35], sofern die Zielgruppe klar definiert ist. Die Auswahl von Studierenden aus technisch relevanten Fach-

richtungen wie Informatik spiegelt die tatsächlichen Nutzergruppen des Remote-Labors realitätsnah wider.

Mit etablierten Methoden wie dem UEQ-Fragebogen, Multiple-Choice- und offenen Fragen werden auch mit einer „kleineren“ Stichprobe valide und vergleichbare Ergebnisse erzielt. Die Größe der Stichprobe ermöglicht somit eine differenzierte Bewertung der Stärken und Schwächen des entwickelten HMIs sowie der getesteten Displays und liefert praxisnahe Optimierungsvorschläge. Zu beachten bleibt, dass die Generalisierbarkeit der Ergebnisse vor allem auf diese spezifische Zielgruppe zutrifft. Für eine breitere Übertragbarkeit auf andere Anwendungsbereiche oder tiefergehende statistische Analysen wäre ein größerer und diverserer Personenkreis erforderlich. Für das Ziel dieser Bachelorarbeit und die explorative Ausrichtung der Evaluation ist der Stichprobenumfang jedoch methodisch angemessen und entspricht gängigen wissenschaftlichen Standards.

7 Fazit und Ausblick

Das folgende Kapitel fasst die gesamte Arbeit noch einmal kurz zusammen und gibt einen Ausblick auf mögliche Weiterentwicklungen des hier erstellten HMIs.

7.1 Fazit

Im Rahmen dieser Bachelorarbeit wurde ein touchbasiertes HMI für das Remote-Labor MICRO entwickelt, implementiert und evaluiert. Ziel war es, ein benutzerfreundliches Administrationswerkzeug bereitzustellen, das wesentliche Aufgaben wie Statusüberwachung, Steuerung der RGB-Beleuchtung und das Handling von Nutzermeldungen direkt vor Ort am Laboraufbau ermöglicht. Durch den Einsatz moderner Webtechnologien (Vue.js), die containerisierte Infrastruktur mit Docker sowie die Integration automatisierter CI/CD-Prozesse konnte ein System geschaffen werden, das sowohl robust als auch flexibel erweiterbar ist.

Die Evaluation mit der Zielgruppe (MICRO-Administratoren) zeigte, dass zentrale funktionale und nicht-funktionale Anforderungen weitestgehend erfüllt wurden. Hinsichtlich der Displays erwies sich insbesondere das Waveshare 15.6 Zoll QLED Display als geeignet, da es eine gute Bildqualität, eine große Darstellungsfläche sowie eine präzise Touch-Eingabe ermöglicht. Bei kleineren Displays traten dagegen Einschränkungen auf, die sich vor allem in schlechteren Bildqualitäten und einer fehlenden Skalierbarkeit, etwa durch Touch-Zoom oder automatische Schriftgrößenanpassung, äußerten. Auf der Ebene der Software zeigten sich Schwächen vor allem in der RGB-Steuerung, deren Bedienung als wenig intuitiv beschrieben wurde, da vor der Farbauswahl zunächst eine Animation gewählt werden musste. Auch die Navigation innerhalb des Systems wies Optimierungspotenzial auf, insbesondere im Hinblick auf die Verständlichkeit der Icon-Beschriftungen und die Orientierungsmöglichkeiten.

Die eingesetzten Methoden, bestehend aus quantitativen und qualitativen Elementen (UEQ, Freitextantworten), bestätigten nicht nur die grundsätzliche Gebrauchstauglichkeit, sondern lieferten auch gezielte Hinweise für Verbesserungen. Insgesamt wur-

de das System als gut strukturiert und technisch zukunftsfähig eingeschätzt. Die Lösung stellt somit einen skalierbaren Beitrag zur Digitalisierung technischer Lehrformate dar und schafft eine belastbare Grundlage für künftige Erweiterungen und Anpassungen.

7.2 Ausblick

Für zukünftige Versionen des HMIs ergeben sich vielfältige Ansatzpunkte, um das System weiter zu verbessern und an die Anforderungen der Nutzer anzupassen.

Ein zentraler Fokus liegt auf der *Erweiterung des Funktionsumfangs*. So könnten künftig zusätzliche Funktionen integriert werden, etwa die direkte Steuerung des Portalroboters, die Überwachung und Anbindung von 3D-Druckern sowie erweiterte Schnittstellen zu Monitoring-Systemen. Dank der modularen Architektur lässt sich diese Erweiterung unkompliziert realisieren und in das bestehende System einbinden.

Die derzeit über eine Station verfügbaren Informationen sind noch sehr begrenzt und erlauben keinen Rückschluss auf den tatsächlichen Zustand, etwa auf defekte Komponenten. An einem entsprechenden *Überwachungssystem* wird aktuell gearbeitet. Sobald dieses funktionsfähig ist, sollten dessen Daten ebenfalls über das HMI verfügbar gemacht werden.

Auch die beispielhafte *Implementierung von Server-Sent-Events (SSE)*, die in Unterunterabschnitt 5.3.1 erläutert wurde, sollte weiter ausgebaut werden, um den Datenverkehr zu reduzieren. Ergänzend bietet sich eine *Optimierung des automatischen Deployments* neuer Softwareversionen an, wodurch sich ebenfalls Einsparungen beim Datenverkehr erzielen lassen.

Ein weiteres wichtiges Verbesserungspotenzial besteht in der *Optimierung der Benutzerführung*. Insbesondere die RGB-Steuerung und die Navigationslogik bieten noch Raum für eine intuitivere Gestaltung. Maßnahmen wie beschriftete Icons, visuelle Orientierungshilfen wie Breadcrumbs oder Seitentitel sowie optimierte Statusanzeigen bei Benutzerinteraktionen können die Bedienbarkeit deutlich erhöhen.

Darüber hinaus sollte die Software in puncto *Responsivität und Barrierefreiheit* weiterentwickelt werden. Funktionen wie Touch-Zoom, automatische Anpassung der Schriftgröße und eine kontrastoptimiertere Farbwahl tragen dazu bei, die Zugänglichkeit für alle Nutzergruppen zu verbessern.

Auch der *Aufbau einer umfassenden Benutzerdokumentation* ist für die weitere Nutzung empfehlenswert. Eine Online-Dokumentation in Form eines Wikis oder eines integrierten Hilfesystems erleichtert die Einarbeitung in komplexere Funktionen und unterstützt eine produktive Nutzung im Alltag.

Zudem ist eine *nutzerzentrierte Weiterentwicklung* unerlässlich. Regelmäßige UX-Tests mit MICRO-Administratoren sollten etabliert werden, um kontinuierliches, datenbasiertes Feedback in den Entwicklungsprozess einfließen zu lassen und so die Akzeptanz und Qualität des Systems im praktischen Einsatz sicherzustellen.

Schließlich empfiehlt sich eine langfristige Hardware-Optimierung und die Erkundung weiterer Displaygrößen. Insbesondere die Evaluation alternativer Displaygrößen, beispielsweise 12 Zoll als Kompromiss zwischen Kompaktheit und Übersichtlichkeit, sowie die Prüfung anderer Displayformate für spezialisierte Anwendungen könnten die Einsatzmöglichkeiten erweitern.

Zusammenfassend legt diese Arbeit das Fundament für ein flexibles, adaptierbares und administrierbares HMI im Kontext eines Remote-Labors. Sie zeigt zugleich auf, dass eine nachhaltige Weiterentwicklung maßgeblich von kontinuierlichem Nutzerfeedback, methodischer Evaluation und technischer Anpassungsfähigkeit abhängt.

Anhang

1 Fragebogen zur Displayevaluation

Display: _____

Evaluationsfragebogen – MICRO HMI-Gerät

Dieser Fragebogen dient der subjektiven Bewertung von Bildschirmen sowie der darauf dargestellten HMI-Software (**H**uman **M**achine **I**nterface). Ziel ist es, sowohl den physischen Eindruck des Displays als auch die Wahrnehmung und Bedienbarkeit der Softwareoberfläche zu erfassen. Bitte beantworten Sie die folgenden Fragen ehrlich und auf Basis Ihres Nutzungseindrucks. Antworten Sie möglichst spontan. Es ist wichtig, dass Sie nicht lange über die Fragen nachdenken, damit Ihre unmittelbare Einschätzung zum Tragen kommt. Kreuzen Sie nur **eine** Antwortmöglichkeit pro Zeile an.

Das HMI dient als zentrale Plattform zur **Überwachung und Steuerung von MICRO-Systemen** – modularen, fernsteuerbaren Experimentierplätzen für die Lehre im Bereich Embedded Systems. Über das HMI können beispielsweise Lichter am sogenannten MACRO-Schrank gesteuert oder Zustände und Daten einzelner Stationen eingesehen werden. MICRO ermöglicht die flexible Einrichtung fernsteuerbarer Versuchsaufbauten. Der MACRO-Schrank ist dabei eine Komponente der Laborinfrastruktur, die mehrere dieser MICRO-Stationen physisch enthält oder unterstützt. Innerhalb des gesamten Fragebogens beziehen sich Fragen über die Software immer auf die **Software im Kontext mit dem verwendetem Bildschirm**.

Teil 1: Aufgaben zur Nutzung der HMI Software

Zuerst sind hier einige Aufgaben aufgelistet, die sie bewältigen sollen, um sich mit dem Gerät vertraut zu machen.

- **Aufgabe 1**: Finden sie den Menüpunkt zur Steuerung der RGB-Lichter und verändern sie die Lichtfarbe.
- **Aufgabe 2**: Finden Sie den Menüpunkt zur Übersicht der Stationen und lesen Sie den aktuellen Reservierungsstatus einer Station aus.
- **Aufgabe 3**: Prüfen Sie, ob für eine Station Reports vorliegen.
- **Aufgabe 4**: Schauen sie, ob Systemweit Reports vorliegen. Finden sie dafür den Menüpunkt für die Reports Ansicht

Nachdem Sie alle Aufgaben bearbeitet oder versucht haben zu bearbeiten, können Sie die Fragen auf den folgenden Seiten beantworten.

Teil 2: Spontane Einschätzung des Gesamteindrucks

Markieren Sie den Punkt auf der Skala, der Ihrer Wahrnehmung des Gerätes am besten entspricht. Die Skala reicht von **−3 (negativ)** bis **+3 (positiv)**. Antworten sie spontan und denken sie nicht über ihre Antwort nach. Diese muss nicht logisch begründbar sein.

Adjektiv	−3	−2	−1	0	+1	+2	+3	**Adjektiv**
behindernd								unterstützend
kompliziert								einfach
ineffizient								effizient
verwirrend								übersichtlich
langweilig								spannend
uninteressant								interessant
unberechenbar								voraussagbar
konventionell								originell
herkömmlich								neuartig
überladen								aufgeräumt
technisch								intuitiv
langsam								schnell

Teil 3: Aussagen zum Bildschirm

Bitte lesen Sie jede Aussage sorgfältig durch und markieren Sie das Kästchen, das Ihrer persönlichen Einschätzung am besten entspricht. Seien sie möglichst spontan in ihrer Wahl. Kreuzen Sie nur **eine** Antwortmöglichkeit pro Zeile an.

Legende:
1: Ich stimme überhaupt nicht zu
2: Ich stimme eher nicht zu
3: Neutral
4: Ich stimme eher zu
5: Ich stimme voll zu

Aussage	1	2	3	4	5
Ich kann allgemein alle HMI-Inhalte bei dieser Größe bequem erfassen.					
Ich kann die Bedienelemente (z.B. Buttons) klar erkennen.					
Ich kann die Bedienelemente (z.B. Buttons) problemlos treffen.					
Ich kann den Text bei dieser Größe angenehm lesen.					
Ich habe nicht das Gefühl, dass Inhalte zu klein dargestellt sind.					
Ich habe nicht das Gefühl, dass Inhalte zu gedrängt dargestellt sind.					
Die Farben sehen so aus, wie ich sie erwarten würde.					
Auf mich wirken die Texte klar und scharf					
Auf mich wirken Symbole klar und scharf					
Ich empfinde die Darstellung als ruhig, flimmerfrei und angenehm.					

Das System reagiert ohne spürbare Verzögerungen auf meine Eingaben.	☐	☐	☐	☐	☐
Auf mich wirken Animationen und Übergänge flüssig und ohne Ruckler.	☐	☐	☐	☐	☐
Ich empfinde die Touch-Bedienung präzise und flüssig.	☐	☐	☐	☐	☐
Ich empfinde die Navigation durch die Menus intuitiv.	☐	☐	☐	☐	☐
Ich konnte Funktionen schnell finden.	☐	☐	☐	☐	☐
Ich wusste jederzeit, wo ich mich im System befinde.	☐	☐	☐	☐	☐
Fehlermeldungen waren verständlich und hilfreich.	☐	☐	☐	☐	☐
Das System bietet angemessene Rückmeldungen. (Popups, Ladebalken …)	☐	☐	☐	☐	☐
Ich wusste, wie ich eine Fehleingabe korrigieren konnte.	☐	☐	☐	☐	☐
Ich konnte die Lichtsteuerung gezielt einsetzen.	☐	☐	☐	☐	☐
Ich wusste, wie ich den Status einer Station abfrage.	☐	☐	☐	☐	☐
Ich wusste, wie ich Reports einzelner Stationen einsehe.	☐	☐	☐	☐	☐
Ich wusste, wie ich Reports des gesamten Systems einsehe.	☐	☐	☐	☐	☐
Ich konnte mich leicht auf meine Aufgaben konzentrieren.	☐	☐	☐	☐	☐
Ich habe sofort gesehen, ob meine Eingabe erfolgreich war.	☐	☐	☐	☐	☐
Das System hat mich gut über Fehler informiert.	☐	☐	☐	☐	☐

Teil 4: Auswahl Fragen

Bitte kreuzen Sie an, ob Sie den Aussagen zustimmen oder nicht. Begründen Sie bei Bedarf bitte anschließend ihre Wahl. Kreuzen Sie nur **eine** Antwortmöglichkeit pro Zeile an.

Frage 1.1	Stift	Finger	Andere
Meine bevorzugte Eingabemöglichkeit:	☐	☐	☐

Mehrfachnennungen bei folgender Frage möglich:

Frage 1.2	schneller	präziser	Andere
Ich empfinde meine favorisierte Eingabemöglichkeit als:	☐	☐	☐

Begründung & Nennung ihrer Wahl, wenn sie „Andere“ angekreuzt haben:

.....

.....

.....

.....

.....

Frage 2	zu hell	genau passend	zu dunkel
Die Bildschirmhelligkeit war:	☐	☐	☐

Begründung:

.....

.....

.....

.....

.....

Frage 3	zu groß	genau passend	zu klein
Die Bildschirmgröße war:	☐	☐	☐

Begründung:

.....

.....

.....

.....

.....

Frage 4	Ja	Nein
Ich fühlte mich zu irgendeiner Zeit mit der Menge an Informationen überfordert:	☐	☐

Wenn ja: Zu welchem Zeitpunkt/In welchem Menü war dies so:

.....

.....

.....

.....

.....

Teil 5: Offene Fragen

Antworten zur Software, die nicht in direktem Zusammenhang mit dem Bildschirm stehen und die Sie bereits in einem vorherigen Fragebogen genannt haben, müssen hier nicht erneut angegeben werden.

1. Was hat Ihnen an dem Bildschirm besonders gefallen?

   .....

   .....

   .....

   .....

   .....

2. Was hat Sie an dem Bildschirm gestört?

3. Gab es Aufgaben, die Sie nicht lösen konnten, oder welche, mit denen Sie Probleme hatten? Was genau was das Problem bei diesen Aufgaben?

4. Wie könnte man das gesamte Setup für eine produktivere Alltagsnutzung verbessern?

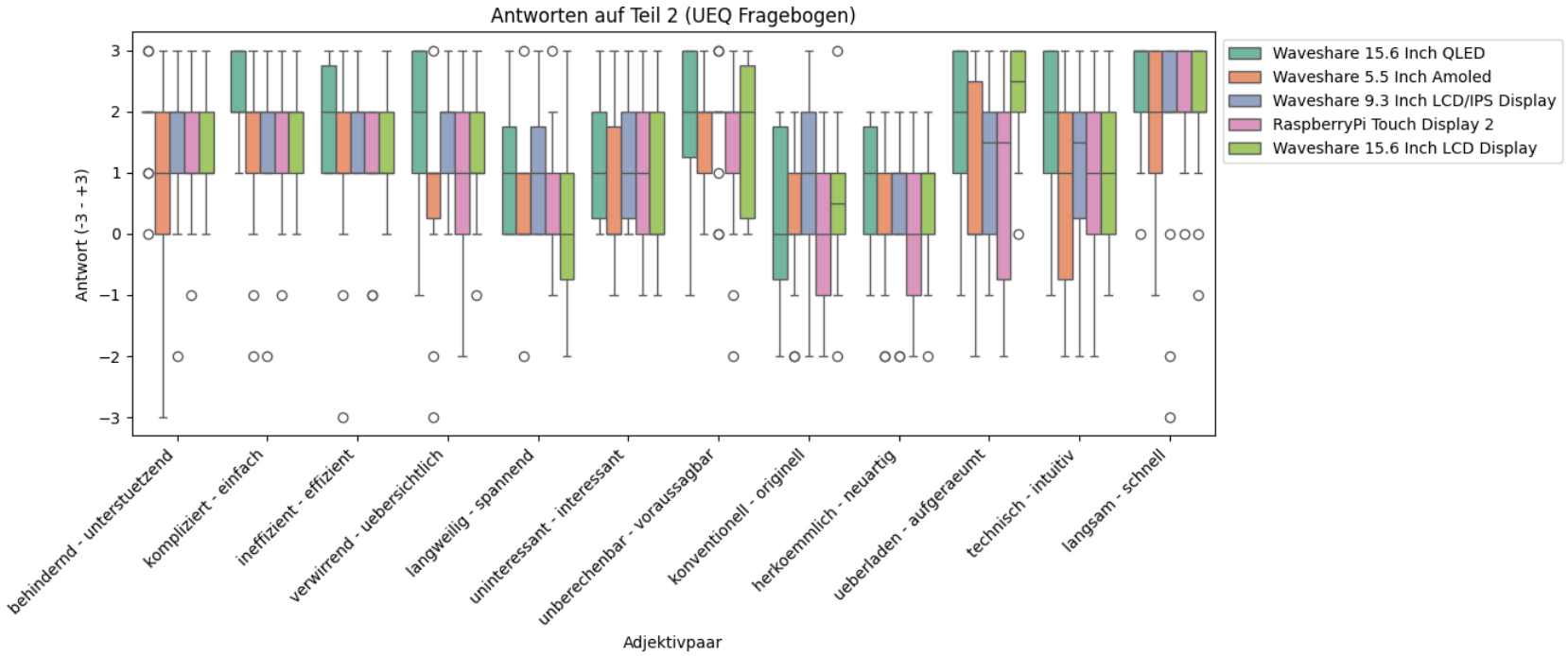
5. Haben Sie Verbesserungsvorschläge oder Wünsche für zukünftige Versionen des Bildschirmes?

6. Haben Sie Verbesserungsvorschläge oder Wünsche für zukünftige Versionen der Software? Etwa Funktionen, die sie vermisst haben?

Vielen Dank für Ihre Teilnahme!

2 Evaluationsergebnisse mit Python ausgewertet

Teil 2: UEQ Fragebogen Analyse



Detaillierte Statistiken:

	question_text	display_name	median	mittelwert	standardabweichung	minimum	maximum	anzahl
0	behindernd - unterstuetzend	RaspberryPi Touch Display 2	1.000000	1.214286	1.121714	-1	3	14
1	behindernd - unterstuetzend	Waveshare 15.6 Inch LCD Display	1.000000	1.428571	0.851631	0	3	14
2	behindernd - unterstuetzend	Waveshare 15.6 Inch QLED	2.000000	1.857143	0.770329	0	3	14
3	behindernd - unterstuetzend	Waveshare 5.5 Inch Amoled	1.000000	0.785714	1.672335	-3	3	14
4	behindernd - unterstuetzend	Waveshare 9.3 Inch LCD/IPS Display	2.000000	1.357143	1.215739	-2	3	14
5	herkoemmlich - neuartig	RaspberryPi Touch Display 2	0.000000	0.142857	1.231456	-2	2	14
6	herkoemmlich - neuartig	Waveshare 15.6 Inch LCD Display	1.000000	0.500000	1.224745	-2	2	14
7	herkoemmlich - neuartig	Waveshare 15.6 Inch QLED	1.000000	0.785714	1.050902	-1	2	14
8	herkoemmlich - neuartig	Waveshare 5.5 Inch Amoled	0.000000	0.214286	1.251373	-2	2	14
9	herkoemmlich - neuartig	Waveshare 9.3 Inch LCD/IPS Display	1.000000	0.500000	1.286019	-2	2	14
10	ineffizient - effizient	RaspberryPi Touch Display 2	1.000000	1.142857	1.027105	-1	2	14
11	ineffizient - effizient	Waveshare 15.6 Inch LCD Display	1.000000	1.428571	0.755929	0	3	14
12	ineffizient - effizient	Waveshare 15.6 Inch QLED	2.000000	1.857143	0.864438	1	3	14
13	ineffizient - effizient	Waveshare 5.5 Inch Amoled	1.000000	0.928571	1.491735	-3	3	14
14	ineffizient - effizient	Waveshare 9.3 Inch LCD/IPS Display	2.000000	1.785714	0.699293	1	3	14
15	kompliziert - einfach	RaspberryPi Touch Display 2	2.000000	1.428571	1.157868	-1	3	14
16	kompliziert - einfach	Waveshare 15.6 Inch LCD Display	2.000000	1.714286	0.994490	0	3	14
17	kompliziert - einfach	Waveshare 15.6 Inch QLED	2.000000	2.285714	0.611250	1	3	14
18	kompliziert - einfach	Waveshare 5.5 Inch Amoled	2.000000	1.285714	1.437336	-2	3	14
19	kompliziert - einfach	Waveshare 9.3 Inch LCD/IPS Display	2.000000	1.642857	1.277446	-2	3	14

	question_text	display_name	median	mittelwert	standardabweichung	minimum	maximum	anzahl
20	konventionell - originell	RaspberryPi Touch Display 2	0.000000	-0.071429	1.328057	-2	2	14
21	konventionell - originell	Waveshare 15.6 Inch LCD Display	0.500000	0.571429	1.283881	-2	3	14
22	konventionell - originell	Waveshare 15.6 Inch QLED	0.000000	0.214286	1.528125	-2	2	14
23	konventionell - originell	Waveshare 5.5 Inch Amoled	0.000000	0.214286	1.251373	-2	2	14
24	konventionell - originell	Waveshare 9.3 Inch LCD/IPS Display	1.000000	0.785714	1.368805	-2	3	14
25	langsam - schnell	RaspberryPi Touch Display 2	2.000000	2.076923	0.954074	0	3	13
26	langsam - schnell	Waveshare 15.6 Inch LCD Display	2.000000	1.928571	1.206666	-1	3	14
27	langsam - schnell	Waveshare 15.6 Inch QLED	3.000000	2.428571	0.937614	0	3	14
28	langsam - schnell	Waveshare 5.5 Inch Amoled	2.000000	1.769231	1.235168	-1	3	13
29	langsam - schnell	Waveshare 9.3 Inch LCD/IPS Display	2.000000	1.615385	2.022311	-3	3	13
30	langweilig - spannend	RaspberryPi Touch Display 2	0.000000	0.500000	1.019049	-1	3	14
31	langweilig - spannend	Waveshare 15.6 Inch LCD Display	0.000000	0.285714	1.382783	-2	3	14
32	langweilig - spannend	Waveshare 15.6 Inch QLED	1.000000	1.071429	1.071612	0	3	14
33	langweilig - spannend	Waveshare 5.5 Inch Amoled	0.000000	0.357143	1.081818	-2	3	14
34	langweilig - spannend	Waveshare 9.3 Inch LCD/IPS Display	1.000000	1.071429	1.206666	0	3	14
35	technisch - intuitiv	RaspberryPi Touch Display 2	1.000000	0.785714	1.423893	-2	3	14
36	technisch - intuitiv	Waveshare 15.6 Inch LCD Display	1.000000	1.000000	1.300887	-1	3	14
37	technisch - intuitiv	Waveshare 15.6 Inch QLED	2.000000	1.571429	1.452546	-1	3	14
38	technisch - intuitiv	Waveshare 5.5 Inch Amoled	1.000000	0.642857	1.736803	-2	3	14
39	technisch - intuitiv	Waveshare 9.3 Inch LCD/IPS Display	1.500000	1.142857	1.460092	-2	3	14

	question_text	display_name	median	mittelwert	standardabweichung	minimum	maximum	anzahl
40	ueberladen - aufgeraeumt	RaspberryPi Touch Display 2	1.500000	1.000000	1.709701	-2	3	14
41	ueberladen - aufgeraeumt	Waveshare 15.6 Inch LCD Display	2.500000	2.214286	0.974961	0	3	14
42	ueberladen - aufgeraeumt	Waveshare 15.6 Inch QLED	2.000000	1.785714	1.368805	-1	3	14
43	ueberladen - aufgeraeumt	Waveshare 5.5 Inch Amoled	0.000000	0.571429	1.869360	-2	3	14
44	ueberladen - aufgeraeumt	Waveshare 9.3 Inch LCD/IPS Display	1.500000	1.214286	1.368805	-1	3	14
45	unberechenbar - voraussagbar	RaspberryPi Touch Display 2	2.000000	1.357143	1.499084	-2	3	14
46	unberechenbar - voraussagbar	Waveshare 15.6 Inch LCD Display	2.000000	1.642857	1.215739	0	3	14
47	unberechenbar - voraussagbar	Waveshare 15.6 Inch QLED	2.000000	1.785714	1.311404	-1	3	14
48	unberechenbar - voraussagbar	Waveshare 5.5 Inch Amoled	2.000000	1.642857	0.928783	0	3	14
49	unberechenbar - voraussagbar	Waveshare 9.3 Inch LCD/IPS Display	2.000000	1.785714	0.892582	0	3	14
50	uninteressant - interessant	RaspberryPi Touch Display 2	1.000000	0.928571	1.141139	-1	3	14
51	uninteressant - interessant	Waveshare 15.6 Inch LCD Display	0.000000	0.714286	1.266647	-1	3	14
52	uninteressant - interessant	Waveshare 15.6 Inch QLED	1.000000	1.214286	0.974961	0	3	14
53	uninteressant - interessant	Waveshare 5.5 Inch Amoled	0.000000	0.642857	1.215739	-1	3	14
54	uninteressant - interessant	Waveshare 9.3 Inch LCD/IPS Display	1.000000	1.285714	1.069045	0	3	14
55	verwirrend - uebersichtlich	RaspberryPi Touch Display 2	1.000000	0.846154	1.573010	-2	3	13
56	verwirrend - uebersichtlich	Waveshare 15.6 Inch LCD Display	1.000000	1.357143	1.215739	-1	3	14
57	verwirrend - uebersichtlich	Waveshare 15.6 Inch QLED	2.000000	1.692308	1.436698	-1	3	13
58	verwirrend - uebersichtlich	Waveshare 5.5 Inch Amoled	1.000000	0.500000	1.454436	-3	3	14

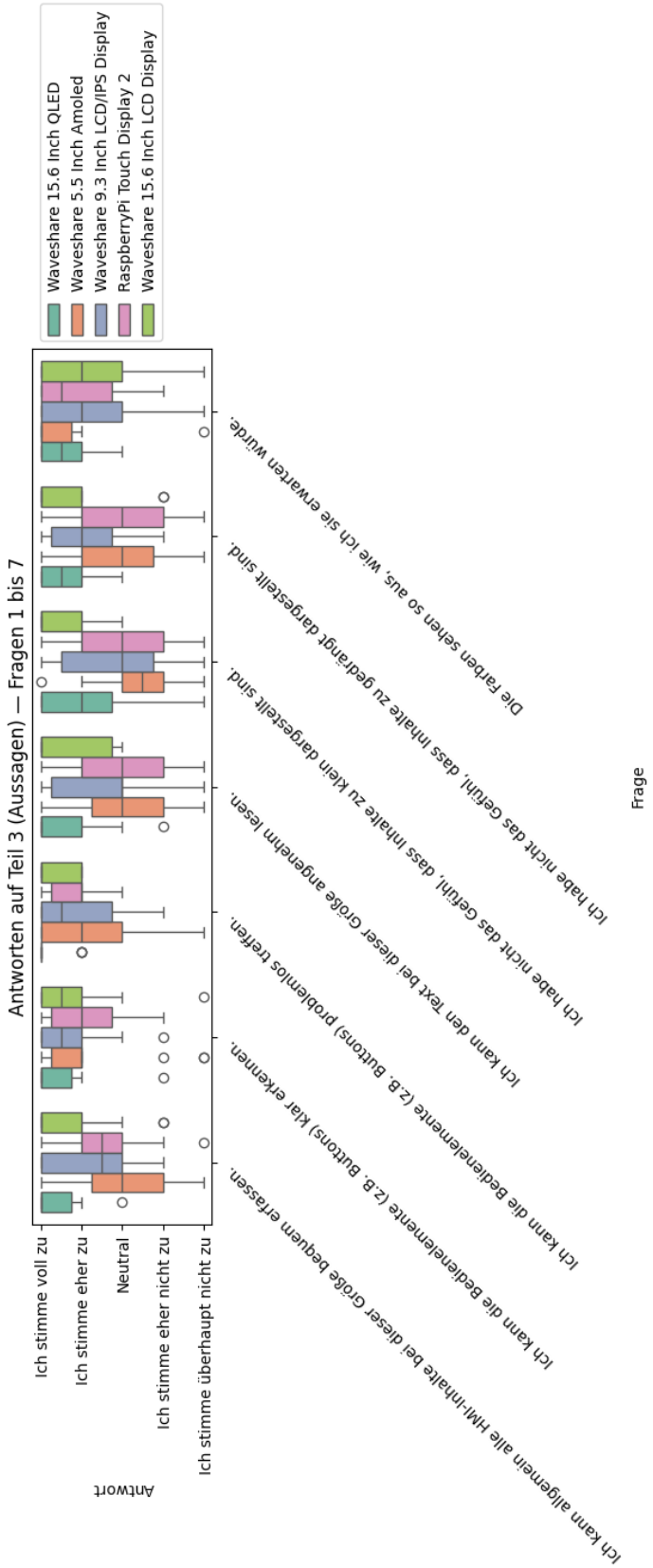
	question_text	display_name	median	mittelwert	standardabweichung	minimum	maximum	anzahl
59	verwirrend - uebersichtlich	Waveshare 9.3 Inch LCD/IPS Display	2.000000	1.642857	1.008208	0	3	14

UEQ Gesamtscores:

	display_name	UEQ_Score
0	Waveshare 15.6 Inch QLED	1.544910
1	Waveshare 9.3 Inch LCD/IPS Display	1.317365
2	Waveshare 15.6 Inch LCD Display	1.232143
3	RaspberryPi Touch Display 2	0.939759
4	Waveshare 5.5 Inch Amoled	0.790419

```
Bestes Display: Waveshare 15.6 Inch QLED mit einem UEQ-Score von 1.5449101796407185
```


Teil 3: Aussagen Analyse


Antworten auf Teil 3 (Aussagen) — Fragen 1 bis 7
Waveshare 15.6 Inch QLED
Waveshare 5.5 Inch Amoled
Waveshare 9.3 Inch LCD/IPS Display
RaspberryPi Touch Display 2
Waveshare 15.6 Inch LCD Display
Antwort
Ich stimme voll zu
Ich stimme eher zu
Neutral
Ich stimme eher nicht zu
Ich stimme überhaupt nicht zu
Frage
Ich kann allgemein alle HMI-Inhalte bei dieser Größe bequem erfassen.
Ich kann die Bedienelemente (z.B. Buttons) klar erkennen.
Ich kann die Bedienelemente (z.B. Buttons) problemlos treffen.
Ich kann den Text bei dieser Größe angenehm lesen.
Ich habe nicht das Gefühl, dass Inhalte zu klein dargestellt sind.
Ich habe nicht das Gefühl, dass Inhalte zu gedrängt dargestellt sind.
Die Farben sehen so aus, wie ich sie erwarten würde.

Antworten auf Teil 3 (Aussagen) — Fragen 8 bis 14
Antwort
Ich stimme voll zu
Ich stimme eher zu
Neutral
Ich stimme eher nicht zu
Ich stimme überhaupt nicht zu
Auf mich wirken die Texte klar und scharf
Auf mich wirken Symbole klar und scharf
Ich empfinde die Darstellung als ruhig, flimmerfrei und angenehm.
Das System reagiert ohne spürbare Verzögerungen auf meine Eingaben.
Auf mich wirken Animationen und Übergänge flüssig und ohne Ruckler.
Ich empfinde die Touch-Bedienung präzise und flüssig.
Ich empfinde die Navigation durch die Menus intuitiv.
Frage
Waveshare 15.6 Inch QLED
Waveshare 5.5 Inch Amoled
Waveshare 9.3 Inch LCD/IPS Display
RaspberryPi Touch Display 2
Waveshare 15.6 Inch LCD Display

Antworten auf Teil 3 (Aussagen) — Fragen 15 bis 21
Antwort
Ich stimme voll zu
Ich stimme eher zu
Neutral
Ich stimme eher nicht zu
Ich stimme überhaupt nicht zu
Ich konnte Funktionen schnell finden.
Ich wusste jederzeit, wo ich mich im System befinde.
Fehlermeldungen waren verständlich und hilfreich.
Das System bietet angemessene Rückmeldungen. (Popups, Ladebalken ...)
Ich wusste, wie ich eine Fehleingabe korrigieren konnte.
Ich konnte die Lichtsteuerung gezielt einsetzen.
Ich wusste, wie ich den Status einer Station abfrage.
Frage
Waveshare 15.6 Inch QLED
Waveshare 5.5 Inch Amoled
Waveshare 9.3 Inch LCD/IPS Display
RaspberryPi Touch Display 2
Waveshare 15.6 Inch LCD Display

Antworten auf Teil 3 (Aussagen) — Fragen 22 bis 26
Antwort
Ich stimme voll zu
Ich stimme eher zu
Neutral
Ich stimme eher nicht zu
Ich stimme überhaupt nicht zu
Ich wusste, wie ich Reports einzelner Stationen einsehe.
Ich wusste, wie ich Reports des gesamten Systems einsehe.
Ich konnte mich leicht auf meine Aufgaben konzentrieren.
Ich habe sofort gesehen, ob meine Eingabe erfolgreich war.
Das System hat mich gut über Fehler informiert.
Frage
Waveshare 15.6 Inch QLED
Waveshare 5.5 Inch Amoled
Waveshare 9.3 Inch LCD/IPS Display
RaspberryPi Touch Display 2
Waveshare 15.6 Inch LCD Display

```
Bewertung für Display: RaspberryPi Touch Display 2
  🥇 Beste Frage: „Auf mich wirken Animationen und Übergänge flüssig und ohne Ruckler.“ mit 4.57 Punkten
  🥉 Schlechteste Frage: „Ich habe nicht das Gefühl, dass Inhalte zu klein dargestellt sind.“ mit 3.07 Punkten

  Übersicht aller Fragen:
   - Auf mich wirken Animationen und Übergänge flüssig und ohne R... → 4.57 Punkte
   - Ich empfinde die Darstellung als ruhig, flimmerfrei und ange... → 4.5 Punkte
   - Das System reagiert ohne spürbare Verzögerungen auf meine Ei... → 4.43 Punkte
   - Ich wusste, wie ich den Status einer Station abfrage.... → 4.43 Punkte
   - Ich konnte mich leicht auf meine Aufgaben konzentrieren.... → 4.36 Punkte
   - Ich konnte die Lichtsteuerung gezielt einsetzen.... → 4.36 Punkte
   - Ich wusste, wie ich Reports einzelner Stationen einsehe.... → 4.29 Punkte
   - Ich wusste, wie ich Reports des gesamten Systems einsehe.... → 4.29 Punkte
   - Ich empfinde die Navigation durch die Menus intuitiv.... → 4.21 Punkte
   - Ich empfinde die Touch-Bedienung präzise und flüssig.... → 4.21 Punkte
   - Auf mich wirken Symbole klar und scharf... → 4.21 Punkte
   - Die Farben sehen so aus, wie ich sie erwarten würde.... → 4.14 Punkte
   - Ich kann die Bedienelemente (z.B. Buttons) problemlos treffe... → 4.14 Punkte
   - Ich wusste jederzeit, wo ich mich im System befinde.... → 4.14 Punkte
   - Auf mich wirken die Texte klar und scharf... → 4.07 Punkte
   - Das System bietet angemessene Rückmeldungen. (Popups, Ladeba... → 4.07 Punkte
   - Das System hat mich gut über Fehler informiert.... → 4.07 Punkte
   - Ich habe sofort gesehen, ob meine Eingabe erfolgreich war.... → 4.0 Punkte
   - Ich konnte Funktionen schnell finden.... → 3.93 Punkte
   - Ich kann die Bedienelemente (z.B. Buttons) klar erkennen.... → 3.93 Punkte
   - Fehlermeldungen waren verständlich und hilfreich.... → 3.79 Punkte
   - Ich wusste, wie ich eine Fehleingabe korrigieren konnte.... → 3.5 Punkte
   - Ich kann allgemein alle HMI-Inhalte bei dieser Größe bequem ... → 3.43 Punkte
   - Ich kann den Text bei dieser Größe angenehm lesen.... → 3.14 Punkte
   - Ich habe nicht das Gefühl, dass Inhalte zu gedrängt dargeste... → 3.14 Punkte
   - Ich habe nicht das Gefühl, dass Inhalte zu klein dargestellt... → 3.07 Punkte

Bewertung für Display: Waveshare 15.6 Inch LCD Display
  🥇 Beste Frage: „Auf mich wirken Animationen und Übergänge flüssig und ohne Ruckler.“ mit 4.79 Punkten
  🥉 Schlechteste Frage: „Die Farben sehen so aus, wie ich sie erwarten würde.“ mit 3.71 Punkten

  Übersicht aller Fragen:
   - Auf mich wirken Animationen und Übergänge flüssig und ohne R... → 4.79 Punkte
   - Ich kann die Bedienelemente (z.B. Buttons) problemlos treffe... → 4.64 Punkte
   - Ich wusste, wie ich den Status einer Station abfrage.... → 4.57 Punkte
   - Ich konnte mich leicht auf meine Aufgaben konzentrieren.... → 4.57 Punkte
```

- Ich empfinde die Touch-Bedienung präzise und flüssig.... → 4.54 Punkte
- Ich habe nicht das Gefühl, dass Inhalte zu klein dargestellt... → 4.5 Punkte
- Das System reagiert ohne spürbare Verzögerungen auf meine Ei... → 4.5 Punkte
- Ich empfinde die Darstellung als ruhig, flimmerfrei und ange... → 4.43 Punkte
- Ich empfinde die Navigation durch die Menus intuitiv.... → 4.38 Punkte
- Ich konnte die Lichtsteuerung gezielt einsetzen.... → 4.36 Punkte
- Auf mich wirken Symbole klar und scharf... → 4.36 Punkte
- Ich habe sofort gesehen, ob meine Eingabe erfolgreich war.... → 4.36 Punkte
- Ich habe nicht das Gefühl, dass Inhalte zu gedrängt dargeste... → 4.36 Punkte
- Ich kann den Text bei dieser Größe angenehm lesen.... → 4.29 Punkte
- Ich kann allgemein alle HMI-Inhalte bei dieser Größe bequem ... → 4.21 Punkte
- Ich kann die Bedienelemente (z.B. Buttons) klar erkennen.... → 4.21 Punkte
- Ich konnte Funktionen schnell finden.... → 4.21 Punkte
- Ich wusste, wie ich Reports des gesamten Systems einsehe.... → 4.21 Punkte
- Ich wusste, wie ich Reports einzelner Stationen einsehe.... → 4.21 Punkte
- Das System bietet angemessene Rückmeldungen. (Popups, Ladeba... → 4.14 Punkte
- Auf mich wirken die Texte klar und scharf... → 4.14 Punkte
- Ich wusste jederzeit, wo ich mich im System befinde.... → 4.07 Punkte
- Fehlermeldungen waren verständlich und hilfreich.... → 4.0 Punkte
- Das System hat mich gut über Fehler informiert.... → 3.86 Punkte
- Ich wusste, wie ich eine Fehleingabe korrigieren konnte.... → 3.79 Punkte
- Die Farben sehen so aus, wie ich sie erwarten würde.... → 3.71 Punkte

Bewertung für Display: Waveshare 15.6 Inch QLED
🥇 Beste Frage: „Ich wusste, wie ich den Status einer Station abfrage.“ mit 4.86 Punkten
🥉 Schlechteste Frage: „Ich wusste, wie ich eine Fehleingabe korrigieren konnte.“ mit 3.64 Punkten

Übersicht aller Fragen:
- Ich wusste, wie ich den Status einer Station abfrage.... → 4.86 Punkte
- Ich empfinde die Touch-Bedienung präzise und flüssig.... → 4.86 Punkte
- Auf mich wirken Animationen und Übergänge flüssig und ohne R... → 4.79 Punkte
- Ich empfinde die Darstellung als ruhig, flimmerfrei und ange... → 4.79 Punkte
- Ich kann die Bedienelemente (z.B. Buttons) problemlos treffe... → 4.79 Punkte
- Das System reagiert ohne spürbare Verzögerungen auf meine Ei... → 4.71 Punkte
- Ich wusste, wie ich Reports des gesamten Systems einsehe.... → 4.71 Punkte
- Ich konnte mich leicht auf meine Aufgaben konzentrieren.... → 4.71 Punkte
- Ich kann allgemein alle HMI-Inhalte bei dieser Größe bequem ... → 4.64 Punkte
- Ich wusste, wie ich Reports einzelner Stationen einsehe.... → 4.64 Punkte
- Auf mich wirken die Texte klar und scharf... → 4.57 Punkte
- Ich wusste jederzeit, wo ich mich im System befinde.... → 4.57 Punkte
- Ich kann die Bedienelemente (z.B. Buttons) klar erkennen.... → 4.57 Punkte

```
  - Ich konnte die Lichtsteuerung gezielt einsetzen.... → 4.5 Punkte
  - Ich habe sofort gesehen, ob meine Eingabe erfolgreich war.... → 4.43 Punkte
  - Auf mich wirken Symbole klar und scharf... → 4.43 Punkte
  - Die Farben sehen so aus, wie ich sie erwarten würde.... → 4.43 Punkte
  - Ich kann den Text bei dieser Größe angenehm lesen.... → 4.36 Punkte
  - Ich konnte Funktionen schnell finden.... → 4.36 Punkte
  - Ich habe nicht das Gefühl, dass Inhalte zu gedrängt dargeste... → 4.29 Punkte
  - Ich empfinde die Navigation durch die Menus intuitiv.... → 4.21 Punkte
  - Das System hat mich gut über Fehler informiert.... → 4.21 Punkte
  - Das System bietet angemessene Rückmeldungen. (Popups, Ladeba... → 4.0 Punkte
  - Fehlermeldungen waren verständlich und hilfreich.... → 3.86 Punkte
  - Ich habe nicht das Gefühl, dass Inhalte zu klein dargestellt... → 3.71 Punkte
  - Ich wusste, wie ich eine Fehleingabe korrigieren konnte.... → 3.64 Punkte

Bewertung für Display: Waveshare 5.5 Inch Amoled
  🥇 Beste Frage: „Ich wusste, wie ich den Status einer Station abfrage.“ mit 4.71 Punkten
  🥉 Schlechteste Frage: „Ich habe nicht das Gefühl, dass Inhalte zu klein dargestellt sind.“ mit 2.71 Punkten

  Übersicht aller Fragen:
  - Ich wusste, wie ich den Status einer Station abfrage.... → 4.71 Punkte
  - Ich konnte die Lichtsteuerung gezielt einsetzen.... → 4.71 Punkte
  - Ich konnte mich leicht auf meine Aufgaben konzentrieren.... → 4.64 Punkte
  - Auf mich wirken die Texte klar und scharf... → 4.5 Punkte
  - Die Farben sehen so aus, wie ich sie erwarten würde.... → 4.5 Punkte
  - Ich wusste, wie ich Reports einzelner Stationen einsehe.... → 4.5 Punkte
  - Ich wusste, wie ich Reports des gesamten Systems einsehe.... → 4.5 Punkte
  - Auf mich wirken Symbole klar und scharf... → 4.5 Punkte
  - Ich wusste jederzeit, wo ich mich im System befinde.... → 4.43 Punkte
  - Ich konnte Funktionen schnell finden.... → 4.36 Punkte
  - Ich empfinde die Touch-Bedienung präzise und flüssig.... → 4.36 Punkte
  - Ich habe sofort gesehen, ob meine Eingabe erfolgreich war.... → 4.29 Punkte
  - Das System hat mich gut über Fehler informiert.... → 4.14 Punkte
  - Das System reagiert ohne spürbare Verzögerungen auf meine Ei... → 4.14 Punkte
  - Ich empfinde die Navigation durch die Menus intuitiv.... → 4.14 Punkte
  - Auf mich wirken Animationen und Übergänge flüssig und ohne R... → 4.0 Punkte
  - Ich empfinde die Darstellung als ruhig, flimmerfrei und ange... → 4.0 Punkte
  - Fehlermeldungen waren verständlich und hilfreich.... → 3.93 Punkte
  - Das System bietet angemessene Rückmeldungen. (Popups, Ladeba... → 3.93 Punkte
  - Ich wusste, wie ich eine Fehleingabe korrigieren konnte.... → 3.86 Punkte
  - Ich kann die Bedienelemente (z.B. Buttons) klar erkennen.... → 3.71 Punkte
  - Ich kann die Bedienelemente (z.B. Buttons) problemlos treffe... → 3.71 Punkte
```

- Ich habe nicht das Gefühl, dass Inhalte zu gedrängt dargeste... → 3.21 Punkte
- Ich kann den Text bei dieser Größe angenehm lesen.... → 2.93 Punkte
- Ich kann allgemein alle HMI-Inhalte bei dieser Größe bequem ... → 2.86 Punkte
- Ich habe nicht das Gefühl, dass Inhalte zu klein dargestellt... → 2.71 Punkte

Bewertung für Display: Waveshare 9.3 Inch LCD/IPS Display
🥇 Beste Frage: „Ich wusste, wie ich den Status einer Station abfrage.“ mit 4.64 Punkten
🥉 Schlechteste Frage: „Ich habe nicht das Gefühl, dass Inhalte zu klein dargestellt sind.“ mit 3.14 Punkten

Übersicht aller Fragen:
- Ich wusste, wie ich den Status einer Station abfrage.... → 4.64 Punkte
- Ich wusste, wie ich Reports einzelner Stationen einsehe.... → 4.5 Punkte
- Ich wusste, wie ich Reports des gesamten Systems einsehe.... → 4.43 Punkte
- Ich wusste jederzeit, wo ich mich im System befinde.... → 4.43 Punkte
- Ich konnte Funktionen schnell finden.... → 4.36 Punkte
- Ich konnte mich leicht auf meine Aufgaben konzentrieren.... → 4.29 Punkte
- Ich konnte die Lichtsteuerung gezielt einsetzen.... → 4.29 Punkte
- Ich empfinde die Navigation durch die Menus intuitiv.... → 4.21 Punkte
- Ich kann die Bedienelemente (z.B. Buttons) klar erkennen.... → 4.21 Punkte
- Auf mich wirken die Texte klar und scharf... → 4.14 Punkte
- Ich kann die Bedienelemente (z.B. Buttons) problemlos treffe... → 4.14 Punkte
- Auf mich wirken Symbole klar und scharf... → 4.08 Punkte
- Auf mich wirken Animationen und Übergänge flüssig und ohne R... → 4.0 Punkte
- Ich habe sofort gesehen, ob meine Eingabe erfolgreich war.... → 4.0 Punkte
- Ich empfinde die Touch-Bedienung präzise und flüssig.... → 4.0 Punkte
- Ich habe nicht das Gefühl, dass Inhalte zu gedrängt dargeste... → 3.93 Punkte
- Ich empfinde die Darstellung als ruhig, flimmerfrei und ange... → 3.86 Punkte
- Das System reagiert ohne spürbare Verzögerungen auf meine Ei... → 3.86 Punkte
- Das System hat mich gut über Fehler informiert.... → 3.86 Punkte
- Das System bietet angemessene Rückmeldungen. (Popups, Ladeba... → 3.79 Punkte
- Ich kann allgemein alle HMI-Inhalte bei dieser Größe bequem ... → 3.71 Punkte
- Die Farben sehen so aus, wie ich sie erwarten würde.... → 3.57 Punkte
- Ich wusste, wie ich eine Fehleingabe korrigieren konnte.... → 3.57 Punkte
- Fehlermeldungen waren verständlich und hilfreich.... → 3.43 Punkte
- Ich kann den Text bei dieser Größe angenehm lesen.... → 3.36 Punkte
- Ich habe nicht das Gefühl, dass Inhalte zu klein dargestellt... → 3.14 Punkte

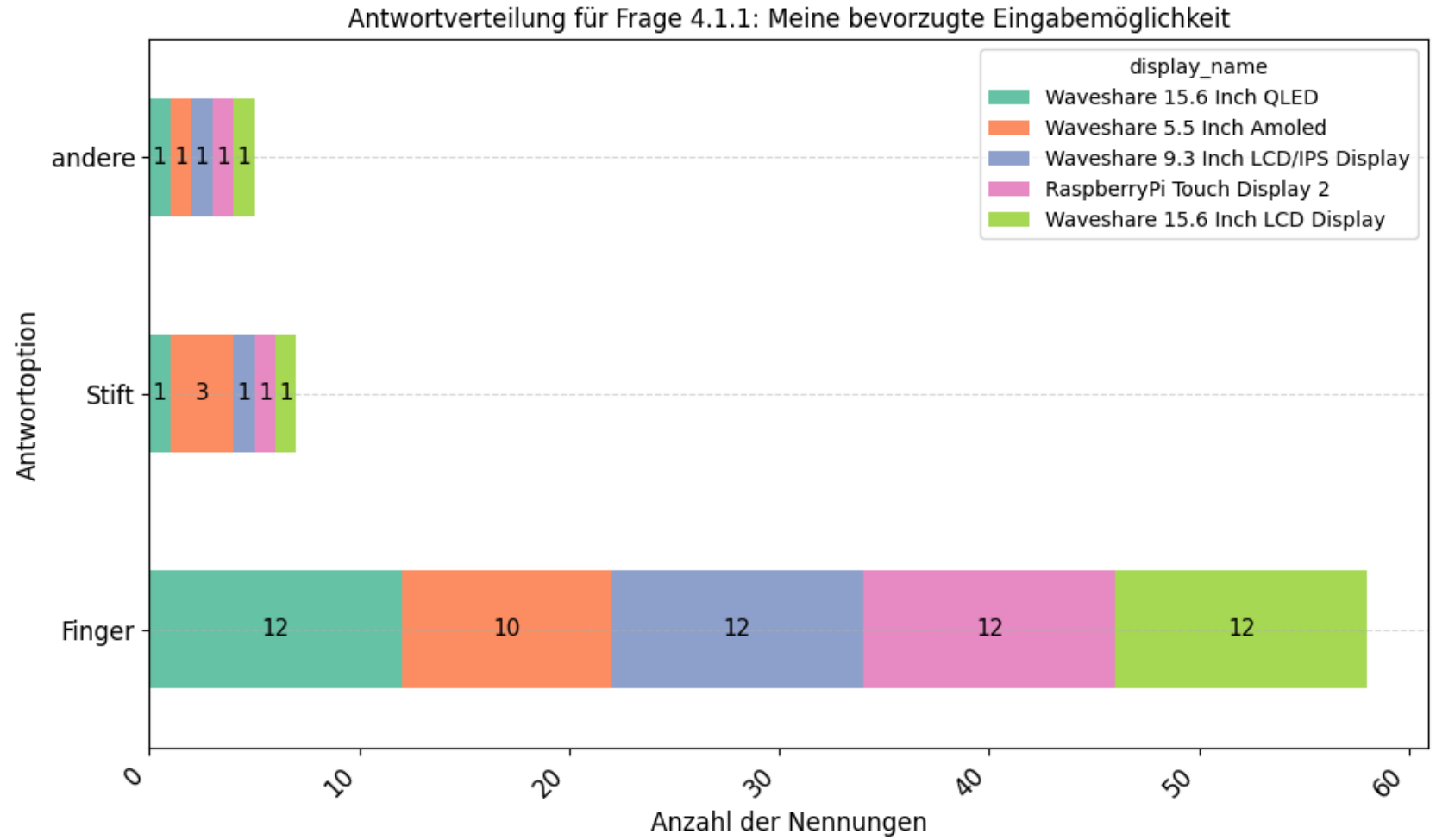
Gesamtscore (mean-Value) pro Display:
display_name
Waveshare 15.6 Inch QLED 4.45
Waveshare 15.6 Inch LCD Display 4.28

```
Waveshare 5.5 Inch Amoled            4.05
RaspberryPi Touch Display 2          4.02
Waveshare 9.3 Inch LCD/IPS Display   3.99
Name: answer_num, dtype: float64
```

Teil 4: Multiple Choice Analyse

Tabelle für Frage 4.1.1: Meine bevorzugte Eingabemöglichkeit

display_name	Waveshare 15.6 Inch QLED	Waveshare 5.5 Inch Amoled	Waveshare 9.3 Inch LCD/ IPS Display	RaspberryPi Touch Display 2	Waveshare 15.6 Inch LCD Display	Gesamt
answer						
Finger	12	10	12	12	12	58
Stift	1	3	1	1	1	7
andere	1	1	1	1	1	5



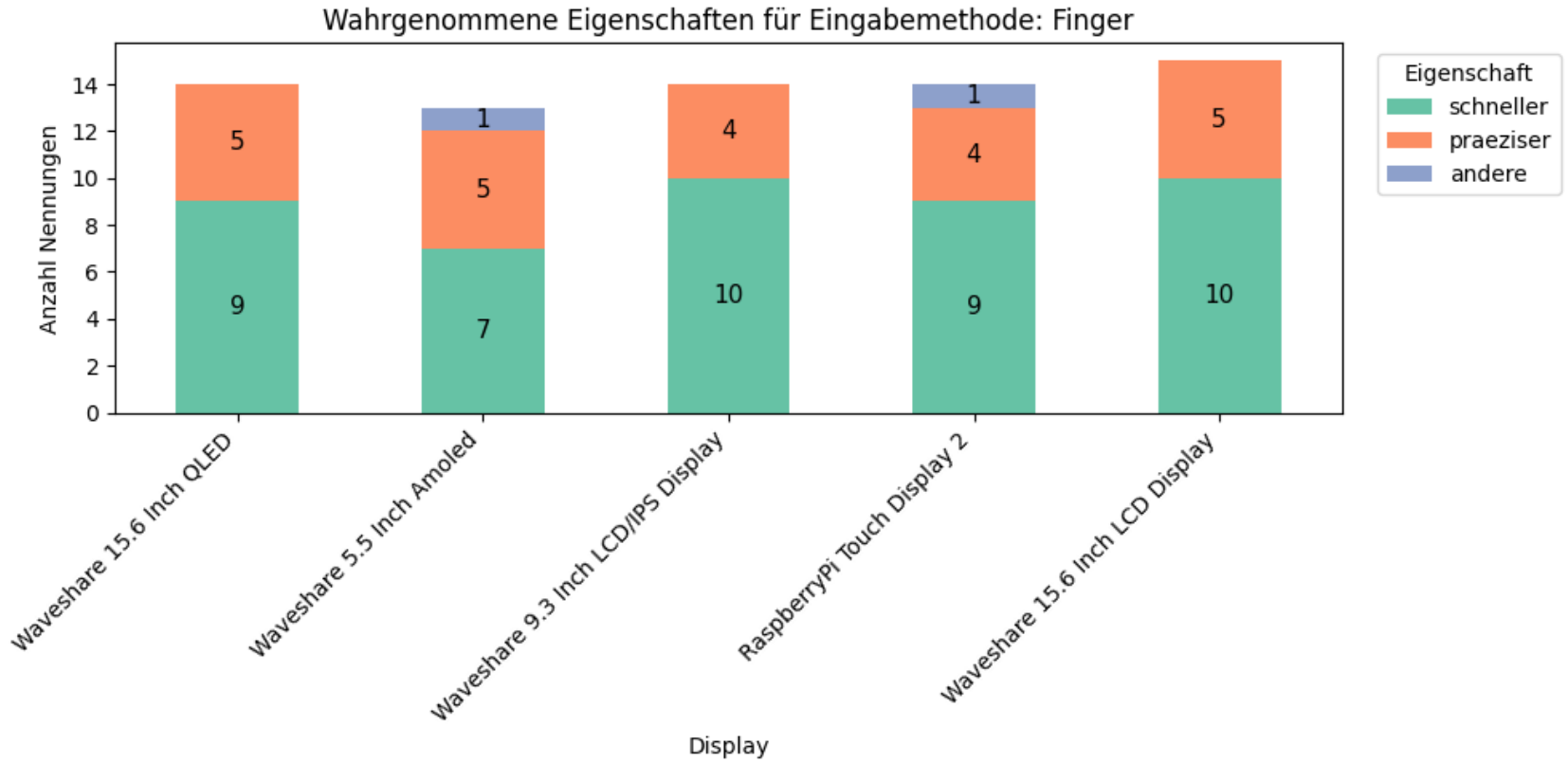
Freitextantworten:

	participant_id	display_name	answer	text
220	10	Waveshare 15.6 Inch QLED	andere	Ich bevorzuge eine Maussteuerung. Da für die meisten Aktionen diese am schnellsten und präziser. Bei einer Oberfläche, die immer weider die selben Bedienpunkte benötigt werden finde ich die Toucheingabe für sinnvoller.
225	10	Waveshare 5.5 Inch Amoled	andere	siehe 1
230	10	Waveshare 9.3 Inch LCD/IPS Display	andere	siehe 1
235	10	RaspberryPi Touch Display 2	andere	siehe 1
240	10	Waveshare 15.6 Inch LCD Display	andere	siehe 1

Tabellen für Frage 4.1.2: Ich empfinde meine favorisierte Eingabemöglichkeit als

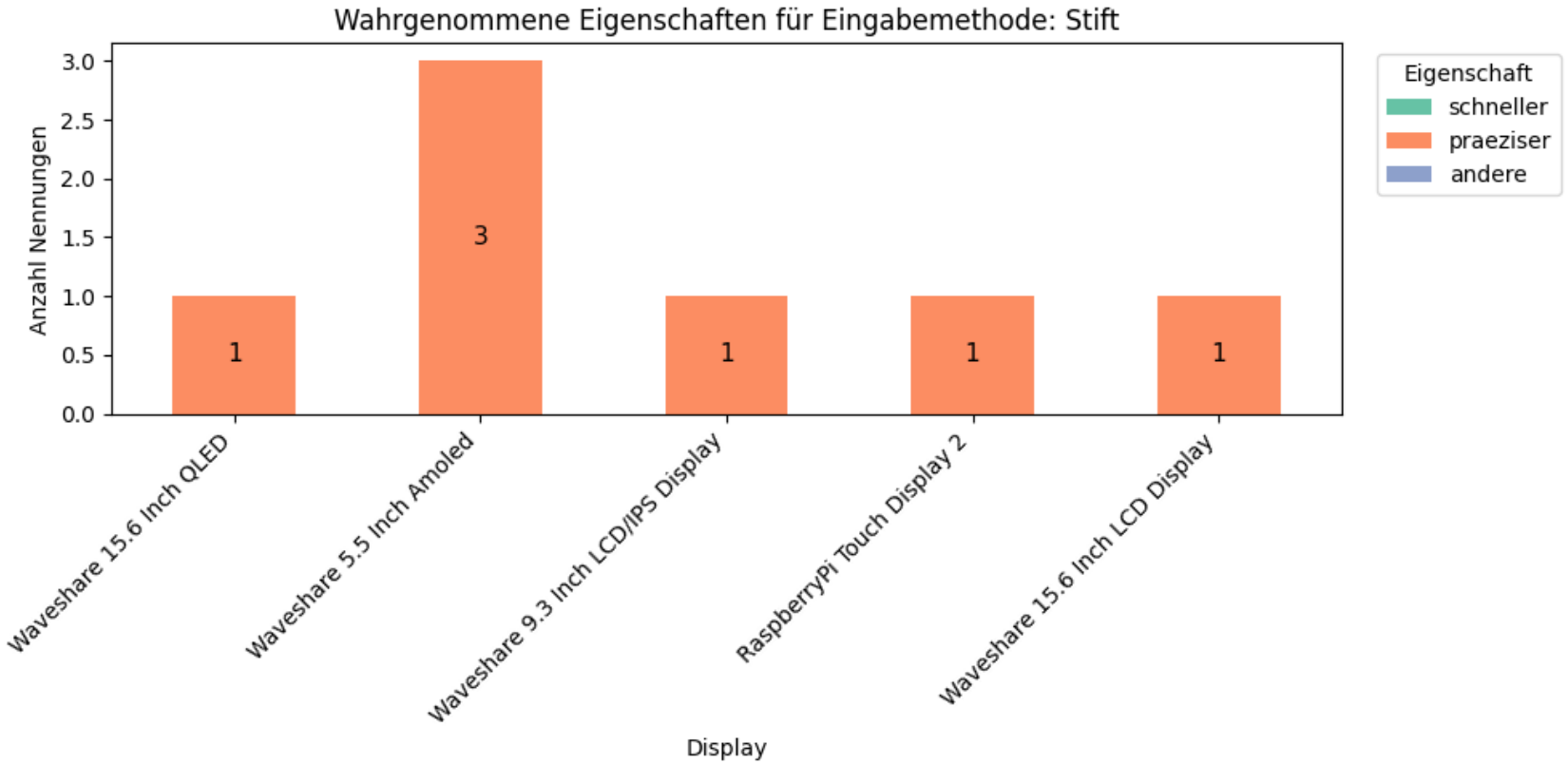
Daten für Eingabemethode: *Finger*

perceived_property	schneller	praeziser	andere
display_name			
Waveshare 15.6 Inch QLED	9	5	0
Waveshare 5.5 Inch Amoled	7	5	1
Waveshare 9.3 Inch LCD/IPS Display	10	4	0
RaspberryPi Touch Display 2	9	4	1
Waveshare 15.6 Inch LCD Display	10	5	0



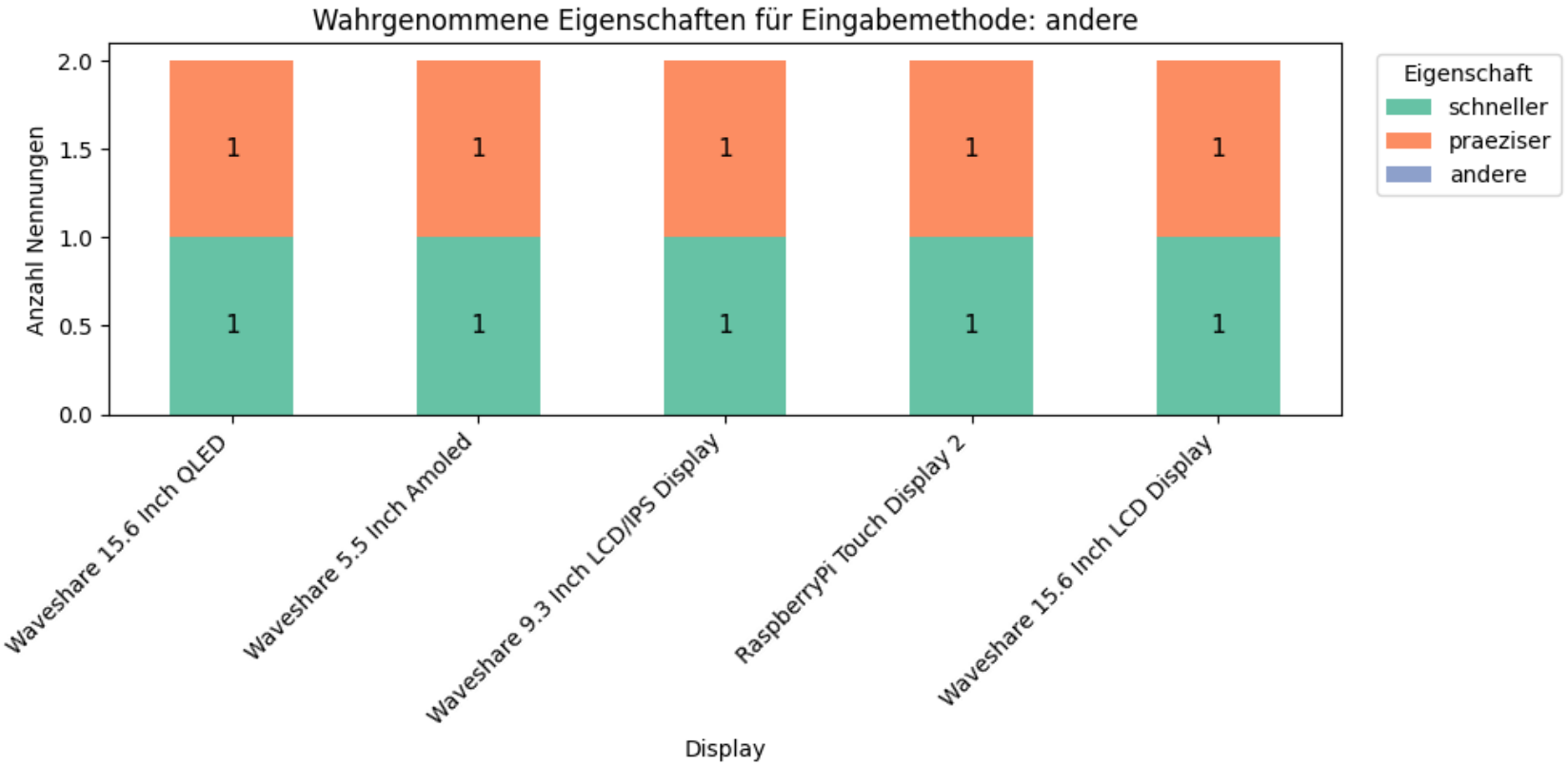
Daten für Eingabemethode: *Stift*

perceived_property display_name	schneller	praeziser	andere
Waveshare 15.6 Inch QLED	0	1	0
Waveshare 5.5 Inch Amoled	0	3	0
Waveshare 9.3 Inch LCD/IPS Display	0	1	0
RaspberryPi Touch Display 2	0	1	0
Waveshare 15.6 Inch LCD Display	0	1	0



Daten für Eingabemethode: *andere*

perceived_property display_name	schneller	praeziser	andere
Waveshare 15.6 Inch QLED	1	1	0
Waveshare 5.5 Inch Amoled	1	1	0
Waveshare 9.3 Inch LCD/IPS Display	1	1	0
RaspberryPi Touch Display 2	1	1	0
Waveshare 15.6 Inch LCD Display	1	1	0

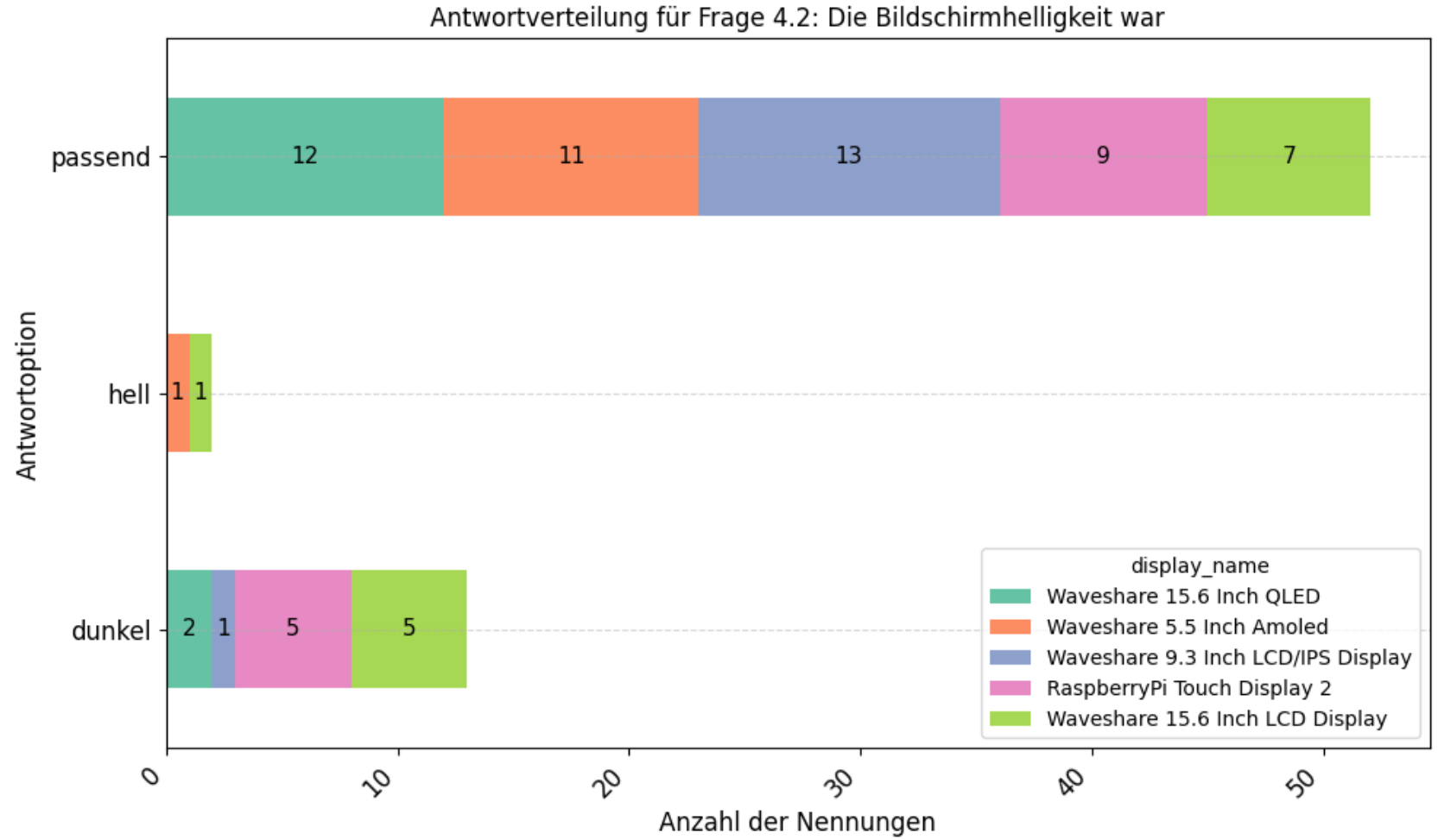


Freitextantworten:

	participant_id	display_name	answer	text
164	7	RaspberryPi Touch Display 2	andere	unwohles gefühl beim Bedienen
172	8	Waveshare 15.6 Inch QLED	schneller	Off topic: Generell wäre Mouse support ganz nett (wenn nicht am Display / Schrank) <3

Tabelle für Frage 4.2: Die Bildschirmhelligkeit war

display_name answer	Waveshare 15.6 Inch QLED	Waveshare 5.5 Inch Amoled	Waveshare 9.3 Inch LCD/ IPS Display	RaspberryPi Touch Display 2	Waveshare 15.6 Inch LCD Display	Gesamt
dunkel	2	0	1	5	5	13
hell	0	1	0	0	1	2
passend	12	11	13	9	7	52

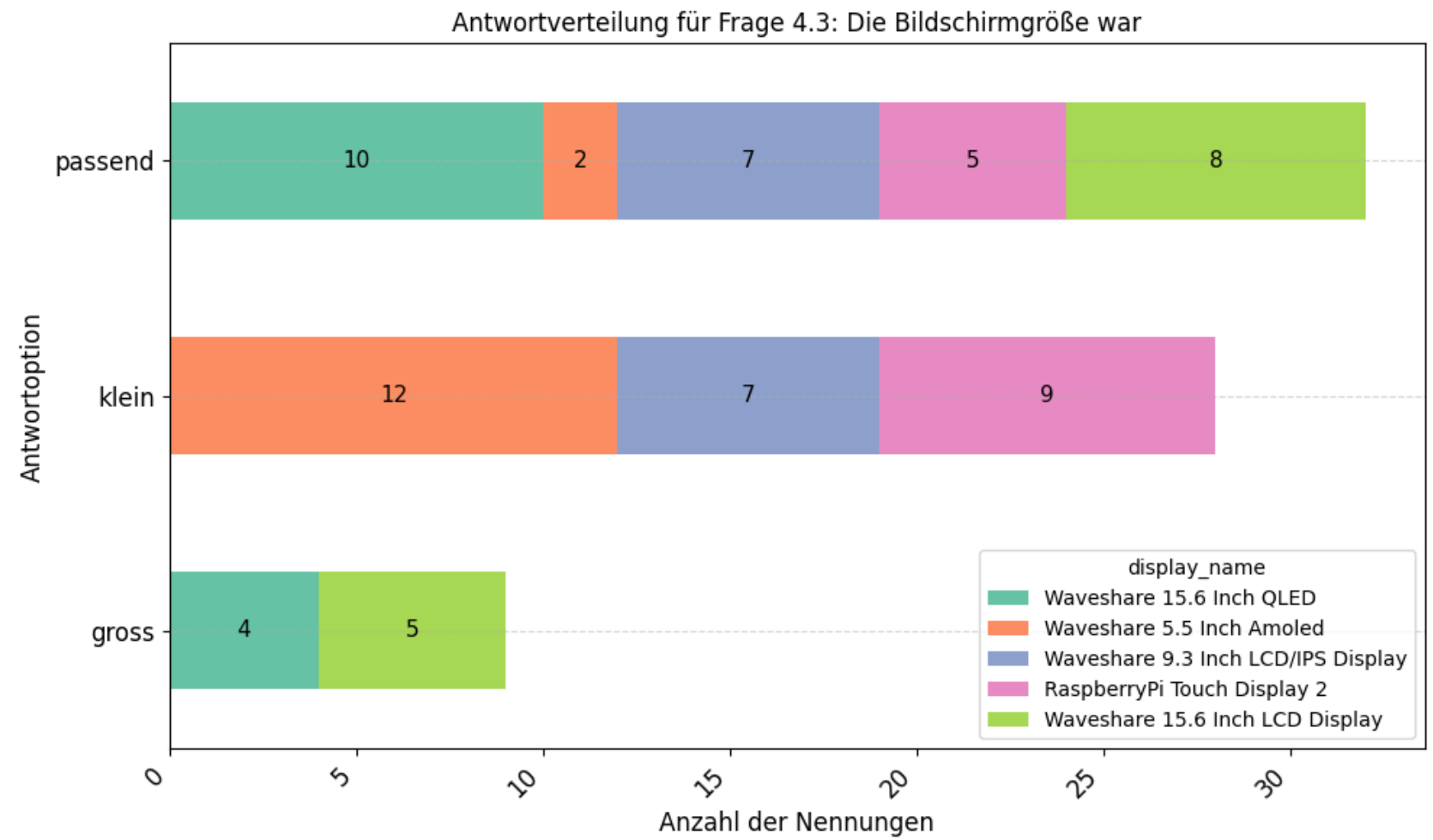


Freitextantworten:

	participant_id	display_name	answer	text
47	2	Waveshare 15.6 Inch LCD Display	dunkel	Etwas blasse Farbdarstellung
97	4	Waveshare 15.6 Inch LCD Display	passend	Ich seh alles
187	8	RaspberryPi Touch Display 2	dunkel	Gefühlt merkt man hier die Spiegelung stärker
192	8	Waveshare 15.6 Inch LCD Display	passend	Kontrast wirkt hier aber schlechter als bei anderen
197	9	Waveshare 15.6 Inch QLED	passend	Die gesamte Benutzeroberfläche war gut zu erkennen.
212	9	RaspberryPi Touch Display 2	dunkel	Ich habe die Helligkeit als etwas zu dunkel empfunden, da ich das Gefühl habe, dass das Display etwas mehr spiegelt.
217	9	Waveshare 15.6 Inch LCD Display	dunkel	wirkt wie Display 4 etwas stärker verspiegelt
222	10	Waveshare 15.6 Inch QLED	dunkel	Bei einer helleren Umgebung halte ich den Kontrast für zu niedrig
227	10	Waveshare 5.5 Inch Amoled	passend	Kontrast ausreichend
232	10	Waveshare 9.3 Inch LCD/IPS Display	dunkel	Kontrast unpassend
237	10	RaspberryPi Touch Display 2	dunkel	Passend in einem Innenraum. Zu dunkel für außenbereiche oder an Fensterbänken.
287	12	RaspberryPi Touch Display 2	dunkel	Wirkt dunkler als die anderen
292	12	Waveshare 15.6 Inch LCD Display	dunkel	Der Bildschirm reflektiert mehr als die anderen, dadurch wäre ein helleres Display hilfreich.
307	13	Waveshare 9.3 Inch LCD/IPS Display	passend	Farbton ist komisch

Tabelle für Frage 4.3: Die Bildschirmgröße war

display_name answer	Waveshare 15.6 Inch QLED	Waveshare 5.5 Inch Amoled	Waveshare 9.3 Inch LCD/ IPS Display	RaspberryPi Touch Display 2	Waveshare 15.6 Inch LCD Display	Gesamt
gross	4	0	0	0	5	9
klein	0	12	7	9	0	28
passend	10	2	7	5	8	32



Freitextantworten:

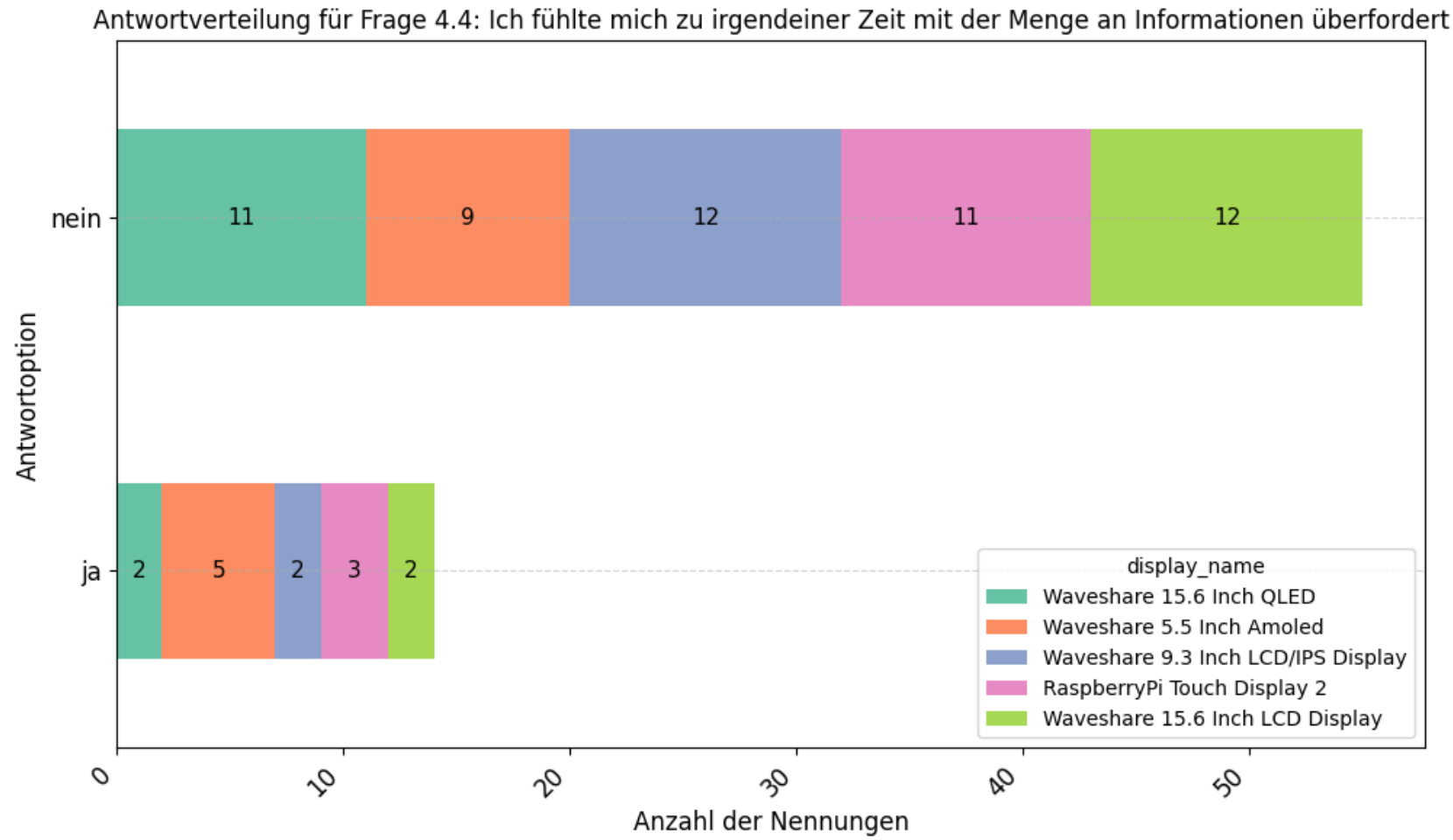
	participant_id	display_name	answer	text
3	1	Waveshare 15.6 Inch QLED	passend	Passt zum restlichen Schrank
8	1	Waveshare 5.5 Inch Amoled	klein	Nicht im Verhältnis zum Schrank. In der Anwendung nicht angenehm
18	1	RaspberryPi Touch Display 2	klein	etwas zu klein
33	2	Waveshare 5.5 Inch Amoled	klein	Textdarstellung in Station Overview zu klein
58	3	Waveshare 5.5 Inch Amoled	klein	Das Interface wirkt zu überladen für die Größe
68	3	RaspberryPi Touch Display 2	klein	Text ist zu klein
73	3	Waveshare 15.6 Inch LCD Display	gross	Bildschirm wirkt schwerfällig
83	4	Waveshare 5.5 Inch Amoled	klein	Durch die geringe Größe, habe ich das Gefühl zu nah an den Display treten zu müssen, da dieser so groß wie ein Handy ist, aber fest montiert wird.
156	7	Waveshare 5.5 Inch Amoled	klein	es ist mit grossen fingern schlecht die auswahl zu klicken.
166	7	RaspberryPi Touch Display 2	klein	rand stört extrem beim ästhetischen anschauen
174	8	Waveshare 15.6 Inch QLED	gross	Minimal zu gross, war sehr häufig viel Whitespace
178	8	Waveshare 5.5 Inch Amoled	klein	Gerade Detail-Texte waren etwas klein
183	8	Waveshare 9.3 Inch LCD/IPS Display	klein	Bitte mehr Vertikale! Damit weniger Scrollen notwendig ist\nDie Extra Horizontale wird hier nicht wirklich genutzt
188	8	RaspberryPi Touch Display 2	klein	Minimal zu klein, Scrollen zwar nicht notwendig, habe mich aber beim Text lesen dabei beobachtet näher ran zu gehen

	participant_id	display_name	answer	text
193	8	Waveshare 15.6 Inch LCD Display	gross	Wirkt etwas leer bei der Größe
198	9	Waveshare 15.6 Inch QLED	gross	Die Benutzeroberfläche wurde ziemlich gross dargestellt, was natürlich nicht schlecht ist. Jedoch hätte meiner Meinung nach auch eine geringere Bildschirmgröße gereicht, da die einzelnen UI-Elemente relativ groß waren und nicht „gedrängt" dargestellt wurden.
203	9	Waveshare 5.5 Inch Amoled	passend	Ich fand den Bildschirm nicht zu klein, da man durch Smartphones Displays mit ähnlicher Größe gewohnt ist. Größer hätte aber auch nicht geschadet.
208	9	Waveshare 9.3 Inch LCD/IPS Display	passend	Mir hat das Seitenverhältnis gut gefallen und im Vergleich hat mir die Größe besser gefallen als Display 2
213	9	RaspberryPi Touch Display 2	passend	Die Bildschirmgröße ähnelt der eines kleinen Tablets und ist somit auch sehr gewohnt.
218	9	Waveshare 15.6 Inch LCD Display	gross	ähnlich zu Display 1. Die große Größe ist nicht schlecht, jedoch für die UI nicht unbedingt nötig.
228	10	Waveshare 5.5 Inch Amoled	klein	Für lange Nutzung ungeeignet
233	10	Waveshare 9.3 Inch LCD/IPS Display	passend	Größe etwas klein aber in dem breiten Format passt es gut.
238	10	RaspberryPi Touch Display 2	passend	Für Permantente arbeiten zu klein, als Statusanzeige mit geringfügiger Konfigurationsfähigkeiten passend.
258	11	Waveshare 9.3 Inch LCD/IPS Display	passend	Weder noch, ich finde die Größe irgendwie unpassend. Weil die Inhalte noch einen größeren Rand an der Seite haben. Es ist zwar nicht zu klein, oder zu groß, aber trotzdem nicht optimal.
273	12	Waveshare 15.6 Inch QLED	passend	Alle Bedienelemente hatten eine angenehme Größe und Text war scharf und groß genug.
278	12	Waveshare 5.5 Inch Amoled	klein	Für ein montiertes Gerät zu klein. Wenn ich es in der Hand halten würde, hätte ich das Gefühl präzise genug zu sein.
283	12	Waveshare 9.3 Inch LCD/IPS Display	klein	Die erweiterte Breite im Vergleich zum kleinsten Display bringt keinen großen Vorteil, weil die Höhe hier relevanter für die Menge an dargestellter Information ist.
288	12	RaspberryPi Touch Display 2	klein	Der Bildschirm wirkte teilweise kleiner als die anderen, obwohl er größer war. Ich glaube das liegt an den sehr breiten Displayrändern.

	participant_id	display_name	answer	text
308	13	Waveshare 9.3 Inch LCD/IPS Display	klein	besser als 2, aber noch zu klein
333	14	Waveshare 9.3 Inch LCD/IPS Display	klein	das Frontend nutzt die gegebene Breite nicht immer sinnvoll. ZB. Station View

Tabelle für Frage 4.4: Ich fühlte mich zu irgendeiner Zeit mit der Menge an Informationen überfordert

display_name	Waveshare 15.6 Inch QLED	Waveshare 5.5 Inch Amoled	Waveshare 9.3 Inch LCD/ IPS Display	RaspberryPi Touch Display 2	Waveshare 15.6 Inch LCD Display	Gesamt
answer						
ja	2	5	2	3	2	14
nein	11	9	12	11	12	55



Freitextantworten:

	participant_id	display_name	answer	text
109	5	Waveshare 5.5 Inch Amoled	ja	Station übersicht / Reports
148	6	Waveshare 15.6 Inch LCD Display	ja	zu viele Stationen werden auf einmal angezeigt.
157	7	Waveshare 5.5 Inch Amoled	ja	zu viel overhead bei so einem kleinen bildschirm
224	10	Waveshare 15.6 Inch QLED	ja	Bei der Stationsübersicht, für mich keine Klaren erkennungsmerkmale zwischen den Stationen. Einfach nur eine Graue Box mit schwarzen Zeichen.
229	10	Waveshare 5.5 Inch Amoled	ja	siehe 1
234	10	Waveshare 9.3 Inch LCD/ IPS Display	ja	siehe 1
239	10	RaspberryPi Touch Display 2	ja	siehe 1
244	10	Waveshare 15.6 Inch LCD Display	ja	siehe 1
279	12	Waveshare 5.5 Inch Amoled	ja	Die Stationen-Übersicht zeigt mehr Infos auf dem Screen an, als man es von mobilen Apps o.ä. auf dem Smartphone gewöhnt ist.
299	13	Waveshare 15.6 Inch QLED	ja	Roboter Ansicht
329	14	Waveshare 5.5 Inch Amoled	ja	Station Overview
334	14	Waveshare 9.3 Inch LCD/ IPS Display	ja	Station Overview
339	14	RaspberryPi Touch Display 2	ja	Stationen, da gescrollt werden musste

Teil 5: Freitextantworten

Frage 5.1: Was hat Ihnen an dem Bildschirm besonders gefallen?

Display: Waveshare 15.6 Inch QLED

	participant_id	text
0	1	Einfach strukturiert, untuitiv Bedienbar
1	2	Die Größe, klar lesbare Inhalte und saubere Farbdarstellung
2	5	Sieht modern aus
3	7	simples user interface ohne das man viel wissen darüber muss dadurch hat man eine einfache Bedienung
4	8	Die intuitive Bedienung auf dieser Größe und einfache treffen und auswählen von Farben/Animationen.\n\nGute Farbwiedergabe.
5	9	Farben, Helligkeit, Auflösung (Alles war einfach erkennbar und klar lesbar)
6	10	Eine Touch-sensitive Oberfläche gab es keine Artefakte bei der Bedienung.
7	11	Ich finde die Größe ist sehr passend. Es wirkt nicht überladen oder sondern perfekt passend.
8	12	Passende Abstände von Elementen, passende Bedienelemente, intuitives User Interface
9	14	Der Station Overview hatte eine sinnvolle Größe

Display: Waveshare 5.5 Inch Amoled

	participant_id	text
0	1	Einfach strukturiert, untuitiv Bedienbar
1	2	Scharfe Texte und Bilddarstellung
2	4	Tolle Farben
3	6	es ist kompakt
4	8	Gute Farbwiedergabe, schnelle Response Time
5	9	Man konnte durch die geringe Größe zumindest gefühlt schneller alle Infos auf einen Blick erkennen.
6	10	Eine Touch-sensitive Oberfläche gab es keine Artefakte bei der Bedienung.
7	12	Gute Farben, Auflösung
8	14	Die Farbqualität ist deutlich besser als bei den anderen Bildschirmen

Display: Waveshare 9.3 Inch LCD/IPS Display

	participant_id	text
0	1	Ultrawide
1	2	Die Ratio
2	3	Das Format des Bildschirms hat viel Potential
3	4	War alles Top, evtl. bisschen besser als nr 4. Farben (Display) sind auch nochmal besser als bei nr 4.
4	6	breiter bildschirm -> texte waren angenehmer zu lesen
5	9	Seitenverhältnis

Display: RaspberryPi Touch Display 2

	participant_id	text
0	1	Standart Bildschirm
1	4	Die Größe wirkt passend und die Darstellung ist sehr klar.
2	6	kleiner und dadurch infos kompakter dargestellt.
3	9	Die Farbdarstellung wirkt natürlicher als zumindest Display 3
4	10	Eine Touch-sensitive Oberfläche gab es keine Artefakte bei der Bedienung.
5	11	Der Rand ist zu dick. Es sieht sehr altmodisch aus.
6	14	das klare Glas wirkt hochwertig

Display: Waveshare 15.6 Inch LCD Display

	participant_id	text
0	1	Die Größe
1	2	Größe
2	5	sieht sehr professionell aus
3	6	das es touch ist
4	9	Die Bedienung wirkt im Vergleich zu vielen anderen Bildschirmen flüssiger
5	11	Die Größe ist gut.
6	12	Die Größe

Frage 5.2: Was hat Sie an dem Bildschirm gestört?

Display: Waveshare 15.6 Inch QLED

	participant_id	text
0	3	Schrift ist bisschen zu klein für die Größe
1	4	Ein wenig zu gross. Könnte aber auch Vorteile haben. Auflösung besser als bei nr. 5 aber schlechter als bei den „kleinen“ - zumindest die Symbole wirken nicht so klar.
2	5	Ich sehe das backlight.
3	6	zu gross
4	10	Die Montagemöglichkeiten die der Monitor mitbringt halte ich für zu gering.

Display: Waveshare 5.5 Inch Amoled

	participant_id	text
0	1	zu klein
1	2	Etwas zu klein
2	4	zu klein
3	6	schrift zu klein, der bildschirm wird schnell warm was die benutzung unangenehm macht.
4	7	zu klein
5	8	Etwas zu klein für meinen Geschmack
6	9	Manche Station-Namen wurden abgeschnitten. Dabei ist vor allem dann auch die ID am Ende des Namens nicht mehr sichtbar.
7	10	Für eine einmalige oder seltene Nutzung ist mir die größe ok. Ich beforzuge aber einen größeren Screen. Besondern im zusammenhang mit einer Toucheingabe.
8	11	Er war zu klein. Es ist nicht so Augen-freundlich.
9	12	Zu klein.
10	13	die Größe

Display: Waveshare 9.3 Inch LCD/IPS Display

	participant_id	text
0	1	etwas zu klein
1	2	LED-Steuerung ging nicht
2	3	Um das Potential des breiten Formats auszunutzen, müsste das Interface angepasst werden.
3	4	Debugg Anzeige von der Schriftart etwas klein.
4	5	Blickwinkel und auflösung ist unangenehm
5	6	die höhe ist zu klein
6	7	könnte etwas größer noch sein
7	8	Farben wirken ausgewaschen im vergleich zu 1 & 2
8	10	Die Farben schienen mir alle mit einem erhöhten blau ton.
9	12	Hatte leichten Blaustich und eine geringere Blickwinkelstabilität.
10	13	Farbton

Display: RaspberryPi Touch Display 2

	participant_id	text
0	1	etwas zu klein
1	2	Der dicke Rand
2	7	Rand stört extrem und etwas klein da war Display 1 und 2 besser
3	8	Wieder schlechte Farbdarstellung, gerade bei hohem Blickwinkel auf das Display.\n\nAuch ausgewaschener als 1 & 2
4	9	stärker verspiegelt
5	10	Display Rand doch sehr groß im vergleich zu anderen Modellen.\n\nIn der Farbauswahl, war ich der Meinung die einzelnen Pixel sehen zu können, was die Qualität etwas gemindert hat. (Pixelränder sichtbar)
6	12	Zu dicke Displayränder, wirkt sehr altbacken im Vergleich zu den anderen.

Display: Waveshare 15.6 Inch LCD Display

	participant_id	text
0	1	Zu klumpig
1	2	Der Screen ist etwas dick. Größerer Bildschirm -> größere Bedienelemente. responsiv?
2	3	Bildschirm wirkt sehr massiv
3	4	Ein wenig groß, für das was er anziegen soll.
4	5	Bildschirm lässt das ganze etwas altern
5	7	Der Rand stört nach einer Zeit für das Auge
6	8	Vergleichsweise schlechte Farbwiedergabe
7	9	Die Spiegelung, die Farbdarstellung und dass zwischen Touchscreen und Display eine sichtbar große Lücke ist. Außerdem ist der Schwarzwert zumindest im Vergleich zu Display 1 ziemlich hell
8	10	Die Farbsättigung war nicht passend.
9	11	Zu dicken Rand. Sieht altmodisch aus. Die Farben sind nicht so kontrastreich, wodurch man Texte mit farbigem Hintergrund nicht so gut erkennt.
10	12	Die Auflösung ist zu niedrig und die Farbdarstellung zu blass. Insgesamt wirkt das Display alt, weil es sehr dick ist und eine Lücke zwischen Touchoberfläche und Display wahrnehmbar ist. Dadurch wirkt die Interaktion nicht so unmittelbar.
11	14	Aufhängung wacklig. Großer Abstand zwischen Glas und Bildschirm

Frage 5.3: Gab es Aufgaben, die Sie nicht lösen konnten, oder welche, mit denen Sie Probleme hatten? Was genau was das Problem bei diesen Aufgaben?

Display: Waveshare 15.6 Inch QLED

	participant_id	text
0	1	Nein, ich konnte alle Aufhaben einfach lösen
1	2	Um eine Farbe auszuwählen muss man zuerst den Modus wählen. Die Farbleiste ist aber das präsente Element
2	7	Bei der Farbauswahl fehlt das man es direkt einstellen ohne das man direkt erstmal einen modus wählen muss.\n\nBei der auswahl des Pinsels (Pipette) buggt der leicht herum
3	9	Bei der LED-Farbeinstellung fand ich es zu Beginn etwas unintuitiv, dass dort „Color Selection unavailable“ stand, wenn man keine Animation ausgewählt hat. Dort könnte man vielleicht den Color-Selector nur anzeigen, falls eine Animation ausgewählt wurde, die das unterstützt.
4	13	Fehlermeldung für die RGB-Farbauswahl. Home-Button etwas hervorheben

Display: Waveshare 9.3 Inch LCD/IPS Display

	participant_id	text
0	2	LED-Steuerung ging nicht

Display: Waveshare 15.6 Inch LCD Display

	participant_id	text
0	4	Aufgabe 1: -uneindeutige Anzeige bei der Farbauswahl, wenn man zuerst das "ausgegraute sieht (bspw. bei Rainbow)
1	6	1. ich wusste nicht das man was aussuchen sollte
2	11	Manche Buttons die farbig sind, haben einen weißen Text drauf. Diesen kann man bei dem Bildschirm sehr schwer erkennen.

Frage 5.4: Wie könnte man das gesamte Setup für eine produktivere Alltagsnutzung verbessern?

Display: Waveshare 15.6 Inch QLED

	participant_id	text
0	1	Anwinklung zur Person
1	3	Einstellbarer Winkel
2	7	Ein Wiki oder Legende wäre gut für alle Funktion und womit man was genau erreicht.
3	8	Ggf. Die Icons links auch in der Hauptansicht für die gleichen Items mitbenutzen.\n\nDen Reserviert-Zustand, ähnlich wie die Visibility, durch ein Icon repräsentieren
4	9	Bis auf die Verbesserung bei der LED-Steuerung finde ich das System sehr übersichtlich.
5	10	Kabelmontage wäre eine 180° Adapter eine gute wahl.

Display: Waveshare 5.5 Inch Amoled

	participant_id	text
0	1	Nicht einsetzbar, würde die Produktivität einschränken auf das Display bezogen.
1	9	IDs an den Anfang des Namens setzen

Display: Waveshare 9.3 Inch LCD/IPS Display

	participant_id	text
0	1	Würde ausreichen

Display: RaspberryPi Touch Display 2

	participant_id	text
0	1	etwas zu klein würde dadurch die Produktivität einschränken

Display: Waveshare 15.6 Inch LCD Display

	participant_id	text
0	5	Skalierungsmöglichkeit via Touch (Zoom) z.B. Schriftgröße
1	6	weniger info bomb
2	14	Stabile Aufhängung

Frage 5.5: Haben Sie Verbesserungsvorschläge oder Wünsche für zukünftige Versionen des Bildschirmes?

Display: Waveshare 15.6 Inch QLED

	participant_id	text
0	2	Benennung der jeweiligen Seite
1	8	Minimal kleinere Diagonale wäre ausreichend.
2	9	Anwinklung des Displays

Display: Waveshare 5.5 Inch Amoled

	participant_id	text
0	1	Größer
1	8	Etwas größere Diagonale wäre ausreichend.
2	9	vielleicht anwinkeln

Display: Waveshare 9.3 Inch LCD/IPS Display

	participant_id	text
0	1	etwas größer
1	9	Vielleicht anwinkeln
2	11	Die Farben sind sehr kühl und bläulich.

Display: RaspberryPi Touch Display 2

	participant_id	text
0	1	größer
1	9	vielleicht anwinkeln und etwas entspiegeln
2	12	Weniger Rand, mehr Display

Display: Waveshare 15.6 Inch LCD Display

	participant_id	text
0	1	Dünner
1	5	Anzeige der Inaktivität (nicht einfach blackscreen) Windows Xp Screensaver
2	8	Minimal kleinere Diagonale wäre ausreichend.
3	9	Anwinkeln und weniger verspiegeln oder Helligkeit erhöhen
4	12	4K OLED

Frage 5.6: Haben Sie Verbesserungsvorschläge oder Wünsche für zukünftige Versionen der Software? Etwa Funktionen, die sie vermisst haben?

Display: Waveshare 15.6 Inch QLED

	participant_id	text
0	2	Farbliche Statusmeldung der Stations
1	4	Mir hat ein 12/12 Zoll Display in der Auswahl gefehlt, diese „Tablet“ Größe wäre eine gute Ergänzung gewesen.
2	8	Siehe oben.\nGgf. Detail Ansicht auf dieser größe weniger kondensiert darstellen.\nManuelle Roboter Steuerung könnte gerne über Button beschriftungen verfügen als auch den Z-Achsen-Zustand preisgeben
3	9	Für die allgemeinen Reports: Es wäre gut, wenn man sehen könnte, von welcher Station ein Report kommt (Vielleicht sogar mit Verlinkung auf die entsprechende Station).
4	10	Eine zusätzliche Information bspw. Wo befinde ich mich gerade? Überschrift?
5	11	Ich find die Scroll-Bar an der Seite im Main-Menu unpassend(?)
6	12	Ich bin meist den Umweg über "Home" gegangen, weil ich nicht wusste was die Icons in der Sidebar bedeuten. Daher vielleicht beschriften.

Display: Waveshare 5.5 Inch Amoled

	participant_id	text
0	4	Ich Fand gut: Die Anzeige des Popups war präsenter als bei den anderen Bildschirmen - ich hab eher mitbekommen, dass es da ist, weil es näher an den anderen Anzeigen war.
1	11	Dachte, dass man zur Zeit keine Static Color auswählen kann, weil da einfach „unavailable“ steht. Könnte man evtl genauer beschreiben?
2	12	Bei der Report-Übersicht fehlt mir bei den Reports die Info, zu welcher Station sie gehören.

Display: Waveshare 9.3 Inch LCD/IPS Display

	participant_id	text
0	11	Weniger Rand um die Inhalte.
1	12	Beanchrichtigungsbadge, in der Station Overview, damit man schnell sehen kann, wo Reports vorliegen.
2	14	Fehlermeldungen waren sehr präsent. Bildschirmbreite wird nicht optimal genutzt.

Display: RaspberryPi Touch Display 2

	participant_id	text
0	5	Reports mit überschrift als gesamt übersicht markieren
1	11	Die Texte sind viel zu klein. Die Farben wirken kühl und sind nicht so kontrastreich. Es ist anstrengend für die Augen.
2	12	Wenn keine Reports da sind, sollten auch die Tabellen-Header nicht angezeigt werden, sondern eine kurze Meldung: "Keine Reports". \nWonach sind die Stationen sortiert?

Display: Waveshare 15.6 Inch LCD Display

	participant_id	text
0	4	Login Button bissi verwirrend, dachte es sei ein zurück Button. Generell Zurück Buttons einbauen
1	5	Report Counter in der Übersicht direkt
2	6	es wäre schön das wenn, man auf die icons auf der seite gedruükt hält gesagt bekommt wohin dich das führt.
3	11	Die Texte sind bisschen zu klein für so einen großen Bildschirm.
4	14	Gesamtfehlerübersicht bei Stationanzeige. Fehler sollten wahrscheinlich auch gelöst und nicht nur weggeklickt werden können.

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

THM Technischen Hochschule Mittelhessen

UI User Interface

UX User Experience

HMI Human-Machine-Interface

SCADA Supervisory Control and Data Acquisition

WinCC Windows Control Center

GTK GIMP-Toolkit

UEQ User Experience Questionnaire

SUS System Usability Scale

meCUE modular evaluation of key Components of User Experience

CSS Cascading Style Sheets

HTML HyperText Markup Language

XML Extensible Markup Language

JSON JavaScript Object Notation

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

URI Uniform Resource Identifier

API Application Programming Interface

SPA Single-Page-Application

MPA Multi-Page-Application

DOM Document Object Model

REST Representational State Transfer

RPC Remote Procedure Call

SSH Secure Shell

QML Qt Modelling Language

OSHMI Open Substation HMI

UART Universal Asynchronous Receiver / Transmitter

LDAP Lightweight Directory Access Protocol

SSE Server-Sent-Events

CI Continuous Integration

CD Continuous Deployment

GPL GNU General Public License

LGPL GNU Lesser General Public License

DIY do it yourself

Glossar

MICRO-Nutzer Ein Nutzer des MICRO-Systems, meistens Studenten oder Lehrende, welche MICRO in einem Modul verwenden

MICRO-Administrator Mitarbeitende im MICRO Projekt, welche das System entwickeln oder warten

DOM Der Document Object Model (DOM) ist eine standardisierte Programmierschnittstelle, die HTML- und XML-Dokumente als baumartige Struktur aus Objekten darstellt, sodass deren Inhalte und Struktur programmatisch gelesen, verändert oder gelöscht werden können. Dabei spiegelt das DOM die logische Gliederung eines Dokuments wider und erlaubt den Zugriff über viele Programmiersprachen, insbesondere JavaScript im Webumfeld [109]

URI Uniform Resource Identifier (URI) ist eine Zeichenkette, die eine Ressource im Internet eindeutig identifiziert. Eine URI kann eine Webadresse (URL), aber auch andere Arten von Identifikatoren sein, wie z.B. eine E-Mail-Adresse oder ein Dateiname [13]

HTTP HTTP ist das grundlegende Protokoll für die Kommunikation zwischen Webbrowsern und Webservern. Es ermöglicht das Laden und Anzeigen von Webseiten und anderen Ressourcen im Internet [38]

HTTPS Hypertext Transfer Protocol Secure (HTTPS) ist die sichere Variante von HTTP, bei der die Datenübertragung zwischen Webbrowser und Webserver mittels TLS (Transport Layer Security) verschlüsselt wird. Es gewährleistet Vertraulichkeit, Integrität und Authentizität der übertragenen Daten

RESTful RESTful bezeichnet eine API, die nach dem REST Prinzip von Roy Fielding [38] entworfen wurde [3]

Mobile-First-Design Mobile-First-Design ist ein Ansatz, bei dem die Gestaltung und Entwicklung einer Benutzeroberfläche zunächst für mobile Geräte erfolgt und erst anschließend für größere Bildschirme erweitert wird. Ziel ist es, die besonderen

Anforderungen mobiler Endgeräte zu berücksichtigen und eine optimale Nutzererfahrung sicherzustellen

Drag-Scrolling Drag-Scrolling ist eine Touch-Geste, bei der durch Ziehen mit dem Finger über den Bildschirm Inhalte vertikal oder horizontal gescrollt werden

GPL Die GNU General Public License (GPL) ist eine weitverbreitete Open-Source-Lizenz, die sicherstellt, dass Software frei verwendet, verändert und weiterverbreitet werden darf. Sie verpflichtet Entwickler, bei Weitergabe des Programmcodes ebenfalls den Quellcode und dieselben Freiheitsrechte bereitzustellen [43]

LGPL Die GNU Lesser General Public License (LGPL) ist eine freie Softwarelizenz, die es erlaubt, lizenzierte Bibliotheken in proprietärer Software zu verwenden, ohne dass der gesamte Quellcode offengelegt werden muss. Änderungen an der LGPL-lizenzierten Komponente selbst müssen jedoch veröffentlicht werden [42]

Root-Nutzer Unter Linux bezeichnet ein Root-Nutzer einen Nutzer mit den höchsten Berechtigungen. Unter Windows werden diese „Administratoren“genannt

Cronjob Ein Cronjob ist eine zeitgesteuerte Aufgabe unter Unix-ähnlichen Betriebssystemen, die mithilfe des *cron*-Dienstes automatisch zu definierten Zeitpunkten oder Intervallen ausgeführt wird

Literatur

[1] @gr0uch. *Swap 2 Rows Benchmark Code*. URL: https://github.com/krausest/js-framework-benchmark/blob/d81dfdb34de1f4b85d007312769168025703652 8/frameworks/keyed/s2/src/bench.js#L22 (besucht am 29. Juli 2025).

[2] Amruth Akoju. „Literature Review: To explore text input techniques for smart watches to overcome the occlusion and fat finger problem“. In: (). URL: https://www.cs.auckland.ac.nz/courses/compsci705s2c/assignments/reports/2014/Smart%20Watches%20aako822.pdf (besucht am 30. Juli 2025).

[3] Alexandru Archip u. a. „RESTful Web Services – A Question of Standards“. In: *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)*. 2018 22nd International Conference on System Theory, Control and Computing (ICSTCC). ISSN: 2372-1618. Okt. 2018, S. 677–682. DOI: 10.1109/ICSTCC.2018.8540763. URL: https://ieeexplore.ieee.org/abstract/document/8540763 (besucht am 26. Juli 2025).

[4] Ines Aubel u. a. „Adaptable Digital Labs - Motivation and Vision of the CrossLab Project“. In: *2022 IEEE German Education Conference (GeCon)*. 2022 IEEE German Education Conference (GeCon). Aug. 2022, S. 1–6. DOI: 10.1109/GeCon55699.2022.9942759. URL: https://ieeexplore.ieee.org/abstract/document/9942759 (besucht am 22. Juli 2025).

[5] Siemens Aktiengesellschaft Digital Industries Factory Automation. *Bedien- und Beobachtungssystem/PC-based Automation*. ST 80 / ST PC. Siemens Aktiengesellschaft Digital Industries Factory Automation, 2024. URL: https://www.siemens.com/de/de/produkte/automatisierung/simatic-hmi.html (besucht am 22. Juli 2025).

[6] AVEVA Group Limited. „AVEVA™ InTouch HMI“. In: ().

[7] AVEVA Group Limited. *AVEVA™ InTouch HMI - Human Machine Interface HMI Software*. URL: https://www.aveva.com/de-de/products/intouch-hmi/ (besucht am 1. Aug. 2025).

[8] AVEVA Group Limited. *Scripts - AVEVA Documentation*. URL: https://docs.aveva.com/bundle/intouch-hmi/page/712894.html (besucht am 1. Aug. 2025).

[9] Branko Balon und Milenko Simić. „Using Raspberry Pi Computers in Education“. In: *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). ISSN: 2623-8764. Mai 2019, S. 671–676. DOI: 10.23919/MIPRO.2019.8756967. URL: https://ieeexplore.ieee.org/abstract/document/8756967 (besucht am 24. Juli 2025).

[10] Emmanuele Bassi. *GTK 4 and Android - Platform / Core*. GNOME Discourse. Section: Platform. 15. Feb. 2020. URL: https://discourse.gnome.org/t/gtk-4-and-android/2808/2 (besucht am 28. Juli 2025).

[11] Christopher Beck u. a. „Introducing a group-based remote laboratory for embedded education“. In: *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*. 2023 IEEE 21st International Conference on Industrial Informatics (INDIN). ISSN: 2378-363X. Juli 2023, S. 1–6. DOI: 10.1109/INDIN51400.2023.10218010. URL: https://ieeexplore.ieee.org/abstract/document/10218010 (besucht am 21. Juli 2025).

[12] Lutz Bellmann u. a. „Digitalisierungsschub in Firmen während der Corona-Pandemie“. In: *Wirtschaftsdienst* 101.9 (1. Sep. 2021), S. 713–718. ISSN: 1613-978X. DOI: 10.1007/s10273-021-3005-3. URL: https://doi.org/10.1007/s10273-021-3005-3 (besucht am 21. Juli 2025).

[13] Tim Berners-Lee. *Universal resource identifiers in WWW: a unifying syntax for the expression of names and addresses of objects on the network as used in the world-wide web*. 1994. URL: https://www.rfc-editor.org/rfc/rfc1630 (besucht am 26. Juli 2025).

[14] Irene Bertschek. „Digitalisierung – der Corona-Impfstoff für die Wirtschaft“. In: *Wirtschaftsdienst* 100.9 (2020). Publisher: Heidelberg: Springer, S. 653–656. ISSN: 1613-978X. DOI: 10.1007/s10273-020-2732-1. URL: https://www.econstor.eu/handle/10419/231809 (besucht am 21. Juli 2025).

[15] Christine Bresnahan und Richard Blum. *Linux Essentials*. Google-Books-ID: Jdg9CgAAQBAJ. John Wiley & Sons, 15. Sep. 2015. 373 S. ISBN: 978-1-119-09206-3.

[16] Christine Bresnahan und Richard Blum. *LPIC-1: Linux Professional Institute Certification Study Guide*. Google-Books-ID: sv3zBgAAQBAJ. John Wiley & Sons, 28. Apr. 2015. 716 S. ISBN: 978-1-119-02119-3.

[17] john Brooke. „SUS: A 'Quick and Dirty' Usability Scale“. In: *Usability Evaluation In Industry*. Num Pages: 6. CRC Press, 1996. ISBN: 978-0-429-15701-1.

[18] Prithwiraj Choudhury u. a. „GitLab: work where you want, when you want“. In: *Journal of Organization Design* 9.1 (16. Nov. 2020), S. 23. ISSN: 2245-408X. DOI: 10.1186/s41469-020-00087-8. URL: https://doi.org/10.1186/s41469-020-00087-8 (besucht am 26. Juli 2025).

[19] Jelica Cincović und Marija Punt. „Comparison: Angular vs. React vs. Vue. Which framework is the best choice?“ In: *Zdravković, M., Konjović, Z., Trajanović, M.(Eds.) ICIST 2020 Proceedings* (2020), S. 250–255. URL: https://www.eventiotic.com/eventiotic/files/Papers/URL/50173409-699e-4b17-8edb-9764ecc53160.pdf (besucht am 28. Juli 2025).

[20] Manjaro GmbH und Co. KG. *Manjaro Linux Official*. URL: https://manjaro.org/ (besucht am 28. Juli 2025).

[21] Weston community. *Welcome to Weston documentation! — weston 14.0.90 documentation*. URL: https://wayland.pages.freedesktop.org/weston/ (besucht am 28. Juli 2025).

[22] The Qt Company. *Qt Success Stories | Real-World Applications & Innovations*. URL: https://www.qt.io/qt-success-stories (besucht am 28. Juli 2025).

[23] OpenJS Foundation {and} Electron contributors. *Why Electron | Electron*. URL: https://electronjs.org/docs/latest/why-electron (besucht am 28. Juli 2025).

[24] Jakob Czekansky, Justin Sauer und Diethelm Bienhaus. „Enhancing Digital Technology Education through MICRO: FPGA-Based Remote Experiments with Simulation-Driven Feedback“. In: awaiting publication. 2025.

[25] Jakob Czekansky, Justin Sauer und Diethelm Bienhaus. „Shared Sessions in MICRO: Enabling Collaborative Remote Experimentation in Embedded Systems Education“. In: DELFI. awaiting publication. 2025.

[26] Jakob Czekansky u. a. „Advancing Embedded Systems Education Through Remote Laboratory Integration in the Teaching Module Microprocessor Technology“. In: *Online Laboratories in Engineering and Technology Education: State of the Art and Trends for the Future*. Hrsg. von Dominik May, Michael E. Auer und Alexander Kist. Cham: Springer Nature Switzerland, 2024, S. 203–222. ISBN: 978-3-

031-70771-1. URL: https://doi.org/10.1007/978-3-031-70771-1_10 (besucht am 21. Juli 2025).

[27] Jakob Czekansky u. a. „Hands-On from Afar: The Future of Embedded Systems Education with the MICRO Remote Lab“. In: The Learning Ideas Conference. awaiting publication. 2025.

[28] Axel Daneels und Wayne Salter. „What is SCADA?“ In: (1999). URL: https://cds.cern.ch/record/532624/files/mc1i01.pdf (besucht am 22. Juli 2025).

[29] Ignacio Díaz-Oreiro u. a. „UX Evaluation with Standardized Questionnaires in Ubiquitous Computing and Ambient Intelligence: A Systematic Literature Review“. In: *Advances in Human-Computer Interaction* 2021.1 (2021), S. 5518722. ISSN: 1687-5907. DOI: 10.1155/2021/5518722. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/5518722 (besucht am 21. Juli 2025).

[30] Docker Inc. *Install Docker using the convenience script*. Docker Documentation. URL: https://docs.docker.com/engine/install/debian/ (besucht am 30. Juli 2025).

[31] Docker Inc. *Post-installation steps*. Docker Documentation. 200. URL: https://docs.docker.com/engine/install/linux-postinstall/ (besucht am 30. Juli 2025).

[32] Docker Inc. *Security*. Docker Documentation. 700. URL: https://docs.docker.com/engine/security/ (besucht am 30. Juli 2025).

[33] Lisa M. Dusseault und James M. Snell. *PATCH Method for HTTP*. Request for Comments RFC 5789. Num Pages: 10. Internet Engineering Task Force, März 2010. DOI: 10.17487/RFC5789. URL: https://datatracker.ietf.org/doc/rfc5789 (besucht am 26. Juli 2025).

[34] Christof Ebert u. a. „DevOps“. In: *IEEE Software* 33.3 (Mai 2016), S. 94–100. ISSN: 1937-4194. DOI: 10.1109/MS.2016.68. URL: https://ieeexplore.ieee.org/abstract/document/7458761 (besucht am 26. Juli 2025).

[35] Laura Faulkner. „Beyond the five-user assumption: Benefits of increased sample sizes in usability testing“. In: *Behavior Research Methods, Instruments, & Computers* 35.3 (1. Aug. 2003), S. 379–383. ISSN: 1532-5970. DOI: 10.3758/BF03195514. URL: https://doi.org/10.3758/BF03195514 (besucht am 14. Aug. 2025).

[36] Xinyang Feng, Jianjing Shen und Ying Fan. „REST: An alternative to RPC for Web services architecture“. In: *2009 First International Conference on Future Information Networks*. 2009 First International Conference on Future Information Networks. Okt. 2009, S. 7–10. DOI: 10.1109/ICFIN.2009.5339611. URL: https://ieeexplore.ieee.org/abstract/document/5339611 (besucht am 26. Juli 2025).

[37] Roy T. Fielding, Mark Nottingham und Julian Reschke. *HTTP Semantics*. Request for Comments RFC 9110. Num Pages: 194. Internet Engineering Task Force, Juni 2022. DOI: 10.17487/RFC9110. URL: https://datatracker.ietf.org/doc/rfc9110 (besucht am 26. Juli 2025).

[38] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000. URL: https://search.proquest.com/openview/fc2d064044b971dda476dfb429a2b344/1?pq-origsite=gscholar&cbl=18750&diss=y (besucht am 26. Juli 2025).

[39] The Apache Software Foundation. *Welcome! - The Apache HTTP Server Project*. URL: https://httpd.apache.org/ (besucht am 26. Juli 2025).

[40] frangoteam. *frangoteam/FUXA - GitHub*. 31. Juli 2025. URL: https://github.com/frangoteam/FUXA (besucht am 1. Aug. 2025).

[41] frangoteam. *FUXA Homepage*. URL: https://frangoteam.org/#downloads (besucht am 1. Aug. 2025).

[42] Inc. Free Software Foundation. *GNU Lesser General Public License, Version 3 - GNU-Projekt - Free Software Foundation*. URL: https://www.gnu.org/licenses/lgpl-3.0.de.html (besucht am 28. Juli 2025).

[43] Inc. Free Software Foundation. *The GNU General Public License v3.0 - GNU Project - Free Software Foundation*. URL: https://www.gnu.org/licenses/gpl-3.0.html.en (besucht am 28. Juli 2025).

[44] Alexandre Freitas u. a. „Pactolo Bar: An Approach to Mitigate the Midas Touch Problem in Non-Conventional Interaction“. In: *Sensors* 23.4 (Jan. 2023). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, S. 2110. ISSN: 1424-8220. DOI: 10.3390/s23042110. URL: https://www.mdpi.com/1424-8220/23/4/2110 (besucht am 30. Juli 2025).

[45] Félix García-Loro u. a. „Remote laboratory VISIR: recent advances, initiatives, federation, limitations and future“. In: *2021 IEEE Global Engineering Education Conference (EDUCON)*. 2021 IEEE Global Engineering Education Conference (EDUCON). ISSN: 2165-9567. Apr. 2021, S. 1754–1757. DOI: 10.1109/

EDUCON46332.2021.9453961. URL: https://ieeexplore.ieee.org/abstract/document/9453961 (besucht am 21. Juli 2025).

[46] GitLab Inc. *Get started with GitLab CI/CD | GitLab Docs*. URL: https://docs.gitlab.com/ci/ (besucht am 31. Juli 2025).

[47] GitLab Inc. *GitLab CI/CD variables | GitLab Docs*. URL: https://docs.gitlab.com/ci/variables/ (besucht am 31. Juli 2025).

[48] GitLab Inc. *Project access tokens | GitLab Docs*. URL: https://docs.gitlab.com/user/project/settings/project_access_tokens/ (besucht am 31. Juli 2025).

[49] Google LLC. *Accessibility: Touch Access*. URL: https://www.chromium.org/user-experience/touch-access/ (besucht am 28. Juli 2025).

[50] Google LLC. *Angular*. Angular. URL: https://angular.dev/ (besucht am 25. Juli 2025).

[51] Google LLC. *Chromium Docs - Sandbox*. URL: https://chromium.googlesource.com/chromium/src/+/main/docs/design/sandbox.md (besucht am 28. Juli 2025).

[52] Google LLC. *Flutter - Build apps for any screen*. URL: //flutter.dev/ (besucht am 28. Juli 2025).

[53] Google LLC. *Material Design*. Material Design. URL: https://m3.material.io (besucht am 3. Aug. 2025).

[54] Google LLC. *Supported deployment platforms*. URL: https://docs.flutter.dev/reference/supported-platforms (besucht am 28. Juli 2025).

[55] Ian Grout. „Remote Laboratories as a Means to Widen Participation in STEM Education“. In: *Education Sciences* 7.4 (Dez. 2017). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, S. 85. ISSN: 2227-7102. DOI: 10.3390/educsci7040085. URL: https://www.mdpi.com/2227-7102/7/4/85 (besucht am 21. Juli 2025).

[56] Ashish Gupta, Meenakshi Panda und Anoop Gupta. „Advancing API Security: A Comprehensive Evaluation of Authentication Mechanisms and Their Implications for Cybersecurity“. In: *International Journal of Global Innovations and Solutions (IJGIS)* (27. Apr. 2024). Publisher: The New World Foundation. DOI: 10.21428/e90189c8.406d2328. URL: https://ijgis.pubpub.org/pub/7drcnfbc/release/1 (besucht am 31. Juli 2025).

[57] Ingvar Gustavsson u. a. „The VISIR project – an Open Source Software Initiative for Distributed Online Laboratories“. In: REV 2007. 2007. URL: https://urn.kb.se/resolve?urn=urn:nbn:se:bth-9135 (besucht am 22. Juli 2025).

[58] Marc Hassenzahl, Michael Burmester und Franz Koller. „AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität“. In: *Mensch & Computer 2003: Interaktion in Bewegung*. Hrsg. von Gerd Szwillus und Jürgen Ziegler. Wiesbaden: Vieweg+Teubner Verlag, 2003, S. 187–196. ISBN: 978-3-322-80058-9. DOI: 10.1007/978-3-322-80058-9_19. URL: https://doi.org/10.1007/978-3-322-80058-9_19 (besucht am 24. Juli 2025).

[59] Karsten Henke u. a. „GOLDi - Grid of Online Lab Devices Ilmenau.“ In: *Int. J. Online Eng.* 12.4 (2016), S. 11–13. URL: https://www.researchgate.net/profile/Heinz-Dietrich-Wuttke/publication/301717748_GOLDi_-_Grid_of_online_lab_devices_Ilmenau/links/5a86be050f7e9b1a9548ecfa/GOLDi-Grid-of-online-lab-devices-Ilmenau.pdf (besucht am 1. Aug. 2025).

[60] Felix Hüning. *Embedded Systems für IoT*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019. ISBN: 978-3-662-57900-8. DOI: 10.1007/978-3-662-57901-5. URL: http://link.springer.com/10.1007/978-3-662-57901-5 (besucht am 24. Juli 2025).

[61] Apple Inc. *Going full screen*. Apple Developer Documentation. URL: https://developer.apple.com/design/human-interface-guidelines/going-full-screen (besucht am 25. Juli 2025).

[62] Docker Inc. *Docker: Accelerated Container Application Development*. 9. Apr. 2025. URL: https://www.docker.com/ (besucht am 27. Juli 2025).

[63] Stack Exchange Inc. *Stack Overflow Website*. Stack Overflow. URL: https://stackoverflow.com/questions (besucht am 25. Juli 2025).

[64] Stack Exchange Inc. *Technology | 2024 Stack Overflow Developer Survey*. URL: https://survey.stackoverflow.co/2024/technology (besucht am 28. Juli 2025).

[65] Shyr-Long Jeng, Wei-Hua Chieng und Yi Chen. „Web-Based Human-Machine Interfaces of Industrial Controllers in Single-Page Applications“. In: *Mobile Information Systems* 2021.1 (2021), S. 6668843. ISSN: 1875-905X. DOI: 10.1155/2021/6668843. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/6668843 (besucht am 22. Juli 2025).

[66] Marin Kaluža, Krešimir Troskot und Bernard Vukelić. „COMPARISON OF FRONT-END FRAMEWORKS FOR WEB APPLICATIONS DEVELOPMENT“. In: *Zbornik Veleučilišta u Rijeci* 6.1 (3. Mai 2018). Publisher: Polytechnic of Rijeka, S. 261–282. ISSN: 1848-1299, 1849-1723. DOI: 10.31784/zvr.6.1.19. URL: https://hrcak.srce.hr/en/clanak/294389%3F (besucht am 21. Juli 2025).

[67] A. Baki Kocabalil, Liliana Laranjo und Enrico Coiera. „Measuring User Experience in Conversational Interfaces: A Comparison of Six Questionnaires“. In: Proceedings of the 32nd International BCS Human Computer Interaction Conference. BCS Learning & Development, 1. Juli 2018. DOI: 10.14236/ewic/HCI2018.21. URL: https://www.scienceopen.com/hosted-document?doi=10.14236/ewic/HCI2018.21 (besucht am 21. Juli 2025).

[68] Stefan Krause. *Interactive Results - Comparison of Frameworks*. URL: https://krausest.github.io/js-framework-benchmark/2025/table_chrome_138.0.7204.50.html (besucht am 28. Juli 2025).

[69] Bettina Laugwitz, Theo Held und Martin Schrepp. „Construction and Evaluation of a User Experience Questionnaire“. In: *HCI and Usability for Education and Work*. Hrsg. von Andreas Holzinger. Berlin, Heidelberg: Springer, 2008, S. 63–76. ISBN: 978-3-540-89350-9. DOI: 10.1007/978-3-540-89350-9_6.

[70] OffSec Services Limited. *Get Kali*. Kali Linux. URL: https://www.kali.org/get-kali/ (besucht am 28. Juli 2025).

[71] Waveshare International Limited. *15.6inch HDMI LCD (H) (with case), 1920x1080, IPS, Raspberry Pi*. URL: https://www.waveshare.com/15.6inch-hdmi-lcd-h-with-case.htm (besucht am 27. Juli 2025).

[72] Waveshare International Limited. *15.6inch QLED Quantum Dot Display, 1920×1080, Optical Bonding Toughened Glass Panel, 100%sRGB, Metal Case, Thin and Light Design | 15.6inch QLED Monitor*. URL: https://www.waveshare.com/15.6inch-qled-monitor.htm (besucht am 27. Juli 2025).

[73] Waveshare International Limited. *5.5inch Capacitive Touch AMOLED Display, 1080×1920, HDMI, Toughened Glass Cover*. URL: https://www.waveshare.com/5.5inch-hdmi-amoled.htm (besucht am 27. Juli 2025).

[74] Waveshare International Limited. *7″ Touch Display Kit for Raspberry Pi Zero, with IPS display Expansion Board, 1024×600, 5-point Capacitive Touch | Zero-DISP-7A*. URL: https://www.waveshare.com/zero-disp-7a.htm (besucht am 27. Juli 2025).

[75] Waveshare International Limited. *9.3inch Capacitive Touch Display, High Brightness, 1600×600, Optical Bonding Toughened Glass Panel, HDMI Interface, IPS | 9.3inch 1600x600 LCD*. URL: https://www.waveshare.com/9.3inch-1600x600-lcd.htm (besucht am 27. Juli 2025).

[76] Jon Loeliger. *Versionskontrolle mit git*. O'Reilly Germany, 2009. URL: https://books.google.com/books?hl=de&lr=&id=AjOvAgAAQBAJ&oi=fnd&pg=PR7&dq=git&ots=e-xITO1r4g&sig=iP7RqQRrHwgSc7WjqDOyWVIRxb0 (besucht am 26. Juli 2025).

[77] Raspberry Pi Ltd. *Buy a Raspberry Pi Zero*. Raspberry Pi Zero. URL: https://www.raspberrypi.com/products/raspberry-pi-zero/ (besucht am 27. Juli 2025).

[78] Raspberry Pi Ltd. *Raspberry Pi 5 Product Brief*. Raspberry Pi Ltd, Jan. 2025. (Besucht am 24. Juli 2025).

[79] Raspberry Pi Ltd. *Raspberry Pi Software*. Raspberry Pi. URL: https://www.raspberrypi.com/software/ (besucht am 30. Juli 2025).

[80] Canonical Ltd. *Download Ubuntu Desktop*. Ubuntu. URL: https://ubuntu.com/download/desktop (besucht am 28. Juli 2025).

[81] Canonical Ltd. *Lubuntu – The official Lubuntu home*. URL: https://lubuntu.me/ (besucht am 28. Juli 2025).

[82] Canonical Ltd. *Ubuntu Core*. Ubuntu. URL: https://ubuntu.com/core (besucht am 28. Juli 2025).

[83] Canonical Ltd. *Xubuntu*. URL: https://xubuntu.org/ (besucht am 28. Juli 2025).

[84] Dominik May u. a. „The remote laboratory VISIR - Introducing online laboratory equipment in electrical engineering classes". In: *2020 IEEE Frontiers in Education Conference (FIE)*. 2020 IEEE Frontiers in Education Conference (FIE). ISSN: 2377-634X. Okt. 2020, S. 1–9. DOI: 10.1109/FIE44824.2020.9274121. URL: https://ieeexplore.ieee.org/document/9274121 (besucht am 1. Aug. 2025).

[85] Inc Meta Platforms. *React*. React. URL: https://react.dev/ (besucht am 25. Juli 2025).

[86] Michael Minge, Laura Riedel und Manfred Thüring. „Modulare evaluation interaktiver Technik. Entwicklung und Validierung des meCUE Fragebogens zur Messung der user experience“. In: *Grundlagen und Anwendungen der Mensch-Technik-Interaktion* 10 (2013). Publisher: Berliner Werkstatt Mensch Maschine-Systeme. Universitätsverlag der TU Be Berlin, S. 28–36. URL: https://www.researchgate.net/profile/Michael-Minge/publication/263714615_Modulare_Evaluation_interaktiver_Technik_Entwicklung_und_Validierung_des_meCUE_Fragebogens_zur_Messung_der_User_Experience/links/0c96053bbd925c3406000000/Modulare-Evaluation-interaktiver-Technik-Entwicklung-und-Validierung-des-meCUE-Fragebogens-zur-Messung-der-User-Experience.pdf (besucht am 24. Juli 2025).

[87] Technische Hochschule Mittelhessen. *Corporate Design*. URL: https://www.thm.de/site/hochschule/service/ag-kommunikation-und-marketing/corporate-design.html (besucht am 26. Juli 2025).

[88] Eduardo San Martin Morote. *Pinia | The intuitive store for Vue.js*. URL: https://pinia.vuejs.org (besucht am 31. Juli 2025).

[89] Dimitris Mourtzis, John Angelopoulos und Nikos Panopoulos. „The Future of the Human–Machine Interface (HMI) in Society 5.0“. In: *Future Internet* 15.5 (Mai 2023). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, S. 162. ISSN: 1999-5903. DOI: 10.3390/fi15050162. URL: https://www.mdpi.com/1999-5903/15/5/162 (besucht am 23. Juli 2025).

[90] Mozilla Foundation. *Window: localStorage property - Web APIs | MDN*. 23. Juni 2025. URL: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage (besucht am 31. Juli 2025).

[91] Rohit J. Nandha und Ronak D. Patel. *Linux for Everyone: Can Standardization Drive Mainstream Adoption?* 29. März 2025. DOI: 10.48550/arXiv.2503.23068. arXiv: 2503.23068[cs]. URL: http://arxiv.org/abs/2503.23068 (besucht am 25. Juli 2025).

[92] NGINX. *nginx*. URL: https://nginx.org/ (besucht am 26. Juli 2025).

[93] NGINX. *nginx documentation*. URL: https://nginx.org/en/docs/ (besucht am 26. Juli 2025).

[94] Gudmundur Bjarni Olafsson. „The Information Kiosk“. In: (). Publisher: Citeseer. URL: https://citeseerx.ist.psu.edu/document?repid=

rep1&type=pdf&doi=4cb48fd5eb2952808740855c7ffecad5e4103ebf (besucht am 24. Juli 2025).

[95] Ricardo Olsen. *riclolsen/OSHMI - GitHub*. 18. Juli 2025. URL: https://github.com/riclolsen/OSHMI (besucht am 1. Aug. 2025).

[96] Ricardo Olsen. *Using OSHMI – Part 1: Common Use Cases*. Control + S. 7. Aug. 2017. URL: https://ricolsen1supervc.wordpress.com/2017/08/07/using-oshmi-part-1-use-cases/ (besucht am 1. Aug. 2025).

[97] Ricardo Olsen. *Using OSHMI – Part 2: Installation*. Control + S. 26. Aug. 2017. URL: https://ricolsen1supervc.wordpress.com/2017/08/26/using-oshmi-part-2-installation/ (besucht am 1. Aug. 2025).

[98] Pablo Orduña u. a. „LabsLand: A sharing economy platform to promote educational remote laboratories maintainability, sustainability and adoption“. In: *2016 IEEE Frontiers in Education Conference (FIE)*. 2016 IEEE Frontiers in Education Conference (FIE). Okt. 2016, S. 1–6. DOI: 10.1109/FIE.2016.7757579. URL: https://ieeexplore.ieee.org/abstract/document/7757579 (besucht am 21. Juli 2025).

[99] Pablo Orduña u. a. „weblablib: Ein neuer Ansatz zur Einrichtung von Remote-Laboren“. In: *Labore in der Hochschullehre-Didaktik, Digitalisierung, Organisation. wbv Media GmbH & Co. KG, Bielefeld* (2021), S. 249–262. URL: https://pdfs.semanticscholar.org/e973/00e47dacecdc74ba0322d6a9fed8d63fb64c.pdf (besucht am 22. Juli 2025).

[100] Pallets. *Welcome to Flask — Flask Documentation (3.1.x)*. URL: https://flask.palletsprojects.com/en/stable/ (besucht am 31. Juli 2025).

[101] Peter Papcun, Erik Kajáti und Jiří Koziorek. „Human Machine Interface in Concept of Industry 4.0“. In: *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*. 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA). Aug. 2018, S. 289–296. DOI: 10.1109/DISA.2018.8490603. URL: https://ieeexplore.ieee.org/abstract/document/8490603 (besucht am 21. Juli 2025).

[102] Havoc Pennington. *GTK+/Gnome application development*. New Riders Indianapolis, 1999. URL: https://www.alpha2.gnuinos.org/GCAD.pdf (besucht am 28. Juli 2025).

[103] Michał Policht. *CuteHMI*. CuteHMI. URL: https://cutehmi.kde.org/ (besucht am 1. Aug. 2025).

[104] Michał Policht. *michpolicht/CuteHMI - GitHub*. 22. Juli 2025. URL: https://github.com/michpolicht/CuteHMI (besucht am 1. Aug. 2025).

[105] Babak Bashari Rad, Harrison John Bhatti und Mohammad Ahmadi. „An introduction to docker and analysis of its performance“. In: *International Journal of Computer Science and Network Security (IJCSNS)* 17.3 (2017). Publisher: International Journal of Computer Science and Network Security, S. 228. URL: https://www.researchgate.net/profile/Harrison-Bhatti/publication/318816158_An_Introduction_to_Docker_and_Analysis_of_its_Performance/links/61facc0c007fb504472fd6c7/An-Introduction-to-Docker-and-Analysis-of-its-Performance.pdf (besucht am 26. Juli 2025).

[106] Satria Ihsan Ramadhan, Erik Haritman und Didin Wahyudin. „Development of BLENODE32 Practicum Module Using ADDIE for Embedded System & IoT Practical Work“. In: *2021 3rd International Symposium on Material and Electrical Engineering Conference (ISMEE)*. 2021 3rd International Symposium on Material and Electrical Engineering Conference (ISMEE). Nov. 2021, S. 137–141. DOI: 10.1109/ISMEE54273.2021.9774195. URL: https://ieeexplore.ieee.org/abstract/document/9774195 (besucht am 22. Juli 2025).

[107] Ossama Rashad, Omneya Attallah und Iman Morsi. „A smart PLC-SCADA framework for monitoring petroleum products terminals in industry 4.0 via machine learning“. In: *Measurement and Control* 55.7 (1. Juli 2022). Publisher: SAGE Publications Ltd, S. 830–848. ISSN: 0020-2940. DOI: 10.1177/00202940221103305. URL: https://doi.org/10.1177/00202940221103305 (besucht am 22. Juli 2025).

[108] Suryaansh Rathinam. „Analysis and Comparison of Different Frontend Frameworks“. In: *Applications and Techniques in Information Security*. Hrsg. von Srikanth Prabhu, Shiva Raj Pokhrel und Gang Li. Singapore: Springer Nature, 2023, S. 243–257. ISBN: 978-981-99-2264-2. DOI: 10.1007/978-981-99-2264-2_19.

[109] Jonathan Robie. *What is the Document Object Model?* URL: https://www.w3.org/TR/WD-DOM/introduction.html (besucht am 26. Juli 2025).

[110] Rockwell Automation. *2713P-T9WD1-B | US*. Rockwell Automation. URL: https://www.rockwellautomation.com/en-us/products/details.2713P-T9WD1-B.html (besucht am 1. Aug. 2025).

[111] Rockwell Automation. *FactoryTalk View-HMI Software | FactoryTalk | UK*. FactoryTalk View HMI. URL: https://www.rockwellautomation.com/en-

gb/products/software/factorytalk/operationsuite/view.html (besucht am 1. Aug. 2025).

[112] Rockwell Automation. „Visualization Solutions Graphic Terminals Tethered Operator Terminals Industrial Computers, Monitors, and Thin Clients Related Software Selection Guide“. In: (2024).

[113] Joanna Rosak-Szyrocka u. a. „Digitalization of Higher Education Around the Globe During Covid-19“. In: *IEEE Access* 10 (2022), S. 59782–59791. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3178711. URL: https://ieeexplore.ieee.org/abstract/document/9784871 (besucht am 23. Juli 2025).

[114] Mojtaba Shahin, Muhammad Ali Babar und Liming Zhu. „Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices“. In: *IEEE Access* 5 (2017), S. 3909–3943. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2685629. URL: https://ieeexplore.ieee.org/abstract/document/7884954 (besucht am 26. Juli 2025).

[115] Siemens Aktiengesellschaft Digital Industries Factory Automation. *Maschinennahes Bedienen und Beobachten*. URL: https://www.siemens.com/de/de/produkte/automatisierung/simatic-hmi/panels.html (besucht am 1. Aug. 2025).

[116] Siemens Aktiengesellschaft Digital Industries Factory Automation. *Produktübersicht für SIMATIC HMI Panels*. (Besucht am 1. Aug. 2025).

[117] Rahul Soni. *Nginx*. Berkeley, CA: Apress, 2016. ISBN: 978-1-4842-1657-6. DOI: 10.1007/978-1-4842-1656-9. URL: http://link.springer.com/10.1007/978-1-4842-1656-9 (besucht am 26. Juli 2025).

[118] Diomidis Spinellis. „Git“. In: *IEEE Software* 29.3 (Mai 2012), S. 100–101. ISSN: 1937-4194. DOI: 10.1109/MS.2012.61. URL: https://ieeexplore.ieee.org/abstract/document/6188603 (besucht am 26. Juli 2025).

[119] Mark Summerfield. *Advanced Qt programming: creating great software with C++ and Qt 4*. Pearson Education, 2010. URL: https://books.google.com/books?hl=de&lr=&id=t_3JCgAAQBAJ&oi=fnd&pg=PT15&dq=qt+software&ots=gBWXielIfC&sig=CsGB-6_gdJhTkk2AOIQ0nZ9nb6E (besucht am 28. Juli 2025).

[120] The GTK Team. *The GTK Project - Overview of GTK and its Libraries*. URL: https://www.gtk.org/docs/architecture/index (besucht am 28. Juli 2025).

[121] Technische Universität Ilmenau. *GOLDi - How to use ECP*. URL: https://www.goldi-labs.net/index.php?Site=70 (besucht am 1. Aug. 2025).

[122] Eben Upton und Gareth Halfacree. *Raspberry Pi User Guide*. Google-Books-ID: WHPhDAAAQBAJ. John Wiley & Sons, 29. Aug. 2016. 327 S. ISBN: 978-1-119-26436-1.

[123] Valve Corporation. *Steam Deck*. Steam Deck. URL: https://www.steamdeck.com/de/ (besucht am 30. Juli 2025).

[124] Valve Corporation. *Steam Support :: Steam Deck Desktop: FAQ*. URL: https://help.steampowered.com/en/faqs/view/671A-4453-E8D2-323C (besucht am 30. Juli 2025).

[125] Valve Corporation. *SteamOS*. 22. Mai 2025. URL: https://store.steampowered.com/steamos?l=german (besucht am 30. Juli 2025).

[126] *Vue.js Rendering Mechanism*. Rendering Mechanism. URL: https://vuejs.org/ (besucht am 25. Juli 2025).

[127] Vuetify. *Button component*. Vuetify. URL: https://vuetifyjs.com/en/components/buttons/ (besucht am 30. Juli 2025).

[128] Vuetify. *Vuetify — A Vue Component Framework*. Vuetify. URL: https://vuetifyjs.com/en/ (besucht am 2. Aug. 2025).

[129] Evan You. *Vue.js*. Vue.js - The Progressive Javascript Framework | Vue.js. URL: https://vuejs.org/ (besucht am 23. Juli 2025).

[130] Evan You und Eduardo San Martin Morote. *Vue Router | The official Router for Vue.js*. URL: https://router.vuejs.org (besucht am 3. Aug. 2025).

[131] Kim Zupancic. *What is Kiosk Mode? Benefits & Use Cases of Kiosk Mode | LogMeIn*. 7. Feb. 2025. URL: https://www.logmein.com/blog/what-is-kiosk-mode-benefits-and-feature-use-cases-of-kiosk-mode (besucht am 25. Juli 2025).