WHZ Westsächsische Hochschule Zwickau
Hochschule für Mobilität

Fraunhofer IWS

Bachelorarbeit

Prototypische Implementierung eines mikrocontrollerbasierten Kommunikationssystems zur Unterstützung der Qualitätssicherungsprozesse innerhalb industrieller Umgebungen

Zur Erlangung des Grades
Bachelor of Science
im Studiengang Informatik
an der Fakultät Physikalische Technik / Informatik
der Westsächsischen Hochschule Zwickau

eingereicht von

Alec Sicheneder

geboren am 03.01.2001

Gutachter:	Prof. Dr. Tina Geweniger
Betreuer:	Alexander Kabardiadi-Virkovski, M.Eng. Fraunhofer IWS (AZOM)
Ausgabedatum:	24. Oktober 2025
Eingereicht am:	02. Januar 2026
Matrikelnummer:	41897
Seminargruppennummer:	202079

Autorenreferat

Qualitätssicherung in Fabrik- und Lagerhallen ist ein Vorgehen, das in der heutigen Zeit bei einer Vielzahl von Unternehmen durchgeführt wird. Dabei kommen die verschiedensten Instrumente und Vorgehen zum Einsatz. Neben den organisatorischen Maßnahmen um die Qualität eines Produktes oder einer Leistung sicherzustellen gibt es die technischen Maßnahmen, um eine entsprechende Qualität gewährleisten zu können.
Eine Auswahl an technischen Maßnahmen, die bei der Überprüfung von Bauteilen zum Einsatz kommt, liegt im Feld der visuellen Kontrolle. Bei der Kategorie der visuellen Kontrolle steht hauptsächlich der Einsatz von einer oder mehreren Lichtquellen und einem oder mehreren Kamerasystemen im Vordergrund. Je nachdem, wie das Licht auf die Oberfläche des Bauteils eingesetzt wird und auftrifft, sind entsprechende Merkmale und Eigenschaften der Oberfläche hervorgehoben. Somit sorgen Kratzer, Erhebungen, Vertiefungen, Prägungen, Beulen, Oberflächenstrukturen und andere Produktionsfehler für eine Beeinflussung des Bildes. Das auftreffende Licht sorgt dann für die Sichtbarkeit dieser Fehler auf dem Kamerabild durch das Hervorheben betroffener Stellen.
Innerhalb dieser Bachelorarbeit ist unter Verwendung von MQTT und MicroPython ein Kommunikationssystem entstanden, mit dem es möglich ist, eine oder mehrere solcher Lichtquellen in Intervallen anzusteuern, die im Bereich von Sekunden bis Mikrosekunden liegen. Nachfolgend wird dieses Kommunikationssystem in industriellen Umgebungen zur Unterstützung von Qualitätssicherungsprozessen eingesetzt.

Abstract

Quality assurance in manufacturing facilities such as storing facilities is a practice that is conducted by every business these days. Within there are different instruments and processes involved. Besides organisational approaches to ensure the quality of a product or a service there are technical approaches too to guarantee a desired quality.
A selection of these technical approaches to ensure the quality of a product is located in the field of visual inspection. In this area one or multiple light sources such as one or multiple camera systems are utilized to minimize or prevent production flaws. Depending on how the light falls on the surface of the component specific characteristics and attributes of this component are highlighted. Like that scratches, elevations, deepenings, embossings, bumps, surface structures and other productional flaws influence the resulting image. The impacting light creates a visibility for those flaws on the image of the camera through highlighting the damaged areas.
Within this Bachelor Thesis using MQTT and MicroPython a communication system was implemented which made it possible to regulate one or more of these light sources in intervals ranging from seconds to microseconds. Following this communication system will assist in quality assurance processes within industrial environments.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

EKI	Erklärbare künstliche Intelligenz
KI	Künstliche Intelligenz
MQTT	Message Queueing Telemetry Transport
WebREPL	Web Read-Eval-Print Loop
PWM	Pulsweitenmodulation

1 Einleitung

Im Bereich der Qualitätssicherungsverfahren lässt sich eine grundlegende Einteilung vornehmen: Einerseits gibt es die zerstörungsfreien Prüfprozesse und andererseits gibt es die zerstörenden Prüfprozesse. Innerhalb dieser Einteilung bestehen Unterkategorien: Die messtechnischen Verfahren, die statistische Prozesskontrolle, die automatisierten Prüfsysteme und die Qualitätsmanagement-Systeme. Innerhalb dieser einzelnen Kategorien gibt es bei den jeweiligen Vertretern auch Überschneidungen. So fällt die Sparte der visuellen Kontrolle in das Feld der zerstörungsfreien Prüfverfahren und wird zudem im Folgenden ebenso als automatisiertes Prüfsystem in Kombination mit Cyber Vision und künstlicher Intelligenz betrachtet. Hierbei stehen eine oder mehrere Lichtquellen sowie eine oder mehrere Kameravorrichtungen im Vordergrund, die dafür zuständig sind, eine visuelle Aufzeichnung der Oberfläche des Bauteils zu erstellen. Diese Aufzeichnung wird anschließend mittels zum Beispiel einer Bilderfassungssoftware auf ihre Merkmale und Eigenschaften ausgewertet.
Die visuelle Kontrolle gilt im Bereich einer industriellen Produktionsumgebung als eine sehr bewährte und effiziente Methode, die Qualität von Bauteilen oder Produkten sicherzustellen. Diese Bauteile oder Produkte, die dabei auf ihre Qualität untersucht werden, stammen aus den jeweiligen Fertigungssituationen, wobei der Fokus dieser Arbeit auf Rolle-zu-Rolle-Prozessen, Pressverfahren und Zerspanungsvorgängen liegt. Die gebräuchlichen Intervalle, in denen die einzelnen Bauteile beleuchtet werden, reichen von Sekunden bis Mikrosekunden und die Genauigkeit des Verfahrens befindet sich im Bereich von bis zu 10 µm.
Für gewöhnlich wird die visuelle Kontrolle der zu prüfenden Bauteile innerhalb räumlich geschlossener Systeme eingesetzt, bei denen eine nachträgliche Änderung oder Korrektur als umständlich oder unmöglich angesehen wurde. Mit dem Ansatz, der in dieser Arbeit verwendet wird, verändert sich die räumliche Zugänglichkeit zu den entsprechenden Prüfsystemen, die die Kameravorrichtungen ansteuern. Somit ändert sich die Möglichkeit für nachträgliche Änderungen oder Korrekturen innerhalb dieser Prüfsysteme.
Im Konkreten wird deshalb in dieser Arbeit untersucht, welche technischen Möglichkeiten bestehen, den Qualitätssicherungsprozess der visuellen Kontrolle durch die Implementierung eines Kommunikationssystems zur Ansteuerung der eingesetzten Lichtquellen zu verändern. Das Besondere hierbei ist der Vergleich zwischen den bisherigen Ansätzen und dem in dieser Bachelorarbeit vorgestellten System: Bisherige Ansätze, welche die visuelle Kontrolle zur Qualitätssicherung verwendet haben, haben eine Schnittstelle zur Ansteuerung der Lichtquellen und Kameravorrichtungen verwendet, die einmal eingestellt wurde und dann entsprechend der

Einstellung visuelle Aufzeichnungen erstellt hat. Das in dieser Bachelorarbeit vorgestellte System ermöglicht die präzise und fortlaufende Konfiguration der Lichtimpulse, die zur Erstellung der visuellen Aufzeichnungen eingesetzt werden, und bietet somit mehr Flexibilität beim Erstellen der visuellen Aufzeichnungen.
Somit wird bei dieser Bachelorarbeit die Frage untersucht, wie dieses Kommunikationssystem zu entwickeln ist, sodass eine oder mehrere LED-basierte Lichtquellen durch das Kommunikationssystem angesteuert werden. Zu dieser Konzeptionierung dieses Kommunikationssystems kommt die Entwicklung, Implementierung und der Einsatz bei einer LED-basierten Lichtquelle. Dadurch entsteht ein zusammengesetztes System aus Hardware, Software und Kommunikationssystem, das im Nachgang in industriellen Umgebungen zur Unterstützung der Qualitätssicherungsprozesse eingesetzt wird.
Zunächst wird auf die theoretischen Grundlagen eingegangen, die sowohl dem Prozess als auch dem Vorgehen und den involvierten Komponenten zugrunde liegen. Danach werden die Methoden und die Anforderungen des Prozesses und des Vorgehens beleuchtet. Hiernach findet die Beschreibung der experimentellen Umsetzung statt, wobei auf die Ausgangssituation, die tatsächliche Umsetzung und die Auswertung eingegangen wird. Nach der experimentellen Umsetzung werden die Ergebnisse der Arbeit zusammengefasst und ins Verhältnis mit bisherigen Umsetzungen gesetzt. Am Ende findet ein Ausblick auf weitere Veränderungen des Systems, die Anpassung anderer Systeme auf Basis der gewonnenen Erkenntnisse und die Anwendung auf andere Bereiche statt.

2 Theoretische Grundlagen

Über die Jahre haben sich die verschiedensten Qualitätssicherungsmethoden entwickelt. Während die traditionellen Methoden wie Six Sigma (SS) [Pos25, S. 223], Statistische Prozesslenkung (SPL) [Pos25, S. 223], die Taguchi-Methode [Pos25, S. 224], Umfassendes Qualitätsmanagement (UQM) [Pos25, S. 225] und die Theory of Constraints [LSL25] primär auf menschliche Ressourcen zurückgreifen und sich mit der Frage beschäftigen, wie man Prozesse so optimiert, dass eine bessere Qualität oder Leistung für den Endkunden erzeugt wird, sind durch die digitale Entwicklung der eingesetzten Technologie und das Wahrnehmen der Möglichkeiten Industrie 4.0s neue Wege und Ansätze entstanden, mit denen man Produktionsprozesse und die damit verbundene Produktqualität weiter optimiert. Zum Einen gibt es das Konzept des sogenannten „Zero-Defect-Manufacturing“ [Sou+22], bei dem die Grundidee ist, dass man jedes Bauteil auf die zuvor definierten Qualitätsmerkmale überprüft und dafür sorgt, dass so wenig Ausschuss und Produktionsressourcenverschwendung wie möglich entsteht. Zum Zweiten greift bei der Überprüfung dieser Bauteile ein weiteres Konzept, das für eine effiziente und ressourcenschonende Produktion sorgt: Die 100%-Kontrolle oder 100%-Prüfung. Bei dieser wird jedes Bauteil ganzheitlich auf mögliche Fehler, Mängel und seine Vollständigkeit überprüft [DS21].
Der dritte Bereich angesprochener Technologien und Methoden befasst sich mit jeder Herangehensweise, die „Data-Mining“, „Machine-Learning“, „Deep-Learning“ und künstliche Intelligenz (KI) sowie erklärbare und verständliche künstliche Intelligenz (EKI) bemühen. Dabei werden die Möglichkeiten und Gegebenheiten der Technologien, die durch Industrie 4.0 entstanden und entwickelt wurden, vollkommen ausgeschöpft. Durch die Verwendung dieser Möglichkeiten entstehen weitere Wege, die Qualitätsanforderungen einzuhalten und bisherige Vorgehen zu verbessern: Die Fehler-Ursachen-Analyse (FUA) [DKE15b; DKE15a], die prädiktive Instandhaltung (PI) [VV25], die visuelle Qualitätskontrolle (VQK) [DIN16] und die Prozessoptimierung (PO) [DS25].
Der Prozess, der in dieser Arbeit beleuchtet wird, basiert auf dem Konzept der visuellen Kontrolle [VF23]. Hierbei stehen eine oder mehrere Lichtquellen sowie eine oder mehrere Kameravorrichtungen im Vordergrund. Je nachdem, wie das Licht eingesetzt wird und auf die Oberfläche des zu prüfenden Bauteils auftrifft, werden die Merkmale und Eigenschaften der Oberfläche dabei sichtbar gemacht. Somit ist die Oberfläche der Bauteile sowie Werkstücke prüfbar und visuelle Aufzeichnungen lassen sich im weiteren Verlauf der Qualitätssicherung erstellen. Diese visuellen Aufzeichnungen werden hierbei an eine Bilderkennungssoftware zur Fehlerdetektion übertragen. Dadurch wird die Einhaltung der Qualitätsanforderungen im produktiven Betrieb

sichergestellt. Die Anpassung des Qualitätssicherungsprozesses der visuellen Kontrolle wird in dieser Arbeit ausschließlich bis zur Auswertung durch die Bilderkennungssoftware betrachtet.

2.1 Mikrocontroller

„Mikrocontroller sind kleine Rechnersysteme, die in einem Baustein alles beinhalten, was zur Realisierung eines Computers benötigt wird: Sie enthalten eine CPU, Speicher und auch Ein-/Ausgabeeinheiten. Die Vorteile eines Mikrocontrollers sind die kompakte Bauform und der günstige Preis. Mikrocontroller werden in unterschiedlichsten, meist kostensensitiven Anwendungen eingesetzt. Ein typisches Einsatzgebiet sind Steuerungs- und Regelungsanwendungen. In Ihrer Waschmaschine sorgt beispielsweise ein Mikrocontroller dafür, dass Sie ein Waschprogramm auswählen können. Er regelt die Wassertemperatur und steuert unter anderem die Elektronik für den Trommelmotor und die Pumpen an. Die Hersteller von Mikrocontrollern bieten eine relativ große Produktpalette an. Meistens werden die Produkte eines Herstellers in Familien unterteilt. Die Produkte einer solchen Familie besitzen in der Regel die gleiche CPU, unterscheiden sich aber im Hinblick auf die Speicherkapazität oder die integrierten Ein-/Ausgabekomponenten. Aufgrund dieser Produktvielfalt kann der Anwender den Controller auswählen, der im Hinblick auf die technischen Eigenschaften und die Kosten optimal für das geplante Einsatzgebiet geeignet ist.“ [Geh+16]

2.2 Elektromagnetische Verträglichkeit

Das Einsetzen von Mikrocontrollern in elektromagnetisch belasteten Bereichen sorgt dafür, dass unvorhergesehene Verhaltensweisen und Fehlfunktionen auftreten können, weswegen es zu beachten gilt, dass die eingesetzten Mikrocontroller für diese Umgebungen ausreichend geschützt und belastbar ausgelegt sind. Vor allem bei Hardware wie Mikrocontrollern, die durch Pins und andere analoge Schnittstellen angesteuert werden, kommt es dazu, dass Manipulationen auf Bit-Ebene dafür sorgen, dass die Hardware fehlerhafte Verhaltensmerkmale aufweist. Zudem ist die WLAN-Funkstrecke, die das zentrale Endgerät bis zum einzelnen Mikrocontroller aufweist, ebenso eine Bruchstelle, die für eine fehlerhafte Kommunikation zwischen dem Mikrocontroller und dem zentralen Endgerät sorgen kann, je nachdem, wie stark die elektromagnetische Belastung in dem eingesetzten Bereich ist. Gemäß des Gesetzes für die elektromagnetische Verträglichkeit von Betriebsmitteln (EMVG) [Bun16] ist die elektromagnetische Verträglichkeit sowie die angemessene Emission entsprechender elektromagnetischer Strahlung im alltäglichen Leben sowie im industriellen Betrieb so geregelt, dass alle elektrischen sowie elektronischen

Geräte in ihrer ordnungsgemäßen Funktion weder für elektromagnetische Beeinträchtigungen sorgen noch durch elektromagnetische Beeinträchtigungen fehlerhafte Verhaltensweisen oder Einschränkungen im Betrieb aufweisen.

2.3 Industrielle Umgebung

Der Einsatzbereich, in dem das System Verwendung findet, ist vornehmlich industriell und im Qualitätssicherungsbereich verordnet. Durch diesen Umstand ist es grundlegend, dass besondere Vorkehrungen getroffen werden, wie die Anbindung an eine passende Kommunikationsschnittstelle, die Untersuchung der elektromagnetischen Emission und die Entscheidung, an welchem Ort das System eingesetzt wird, ob in einem geschlossenen oder in einem offenen System.

Beeinträchtigung der Übertragung

Besonders bei der drahtlosen Kommunikation gibt es Bruchstellen und Potenzial für Unterbrechungen oder Fehlverhalten, die bei der drahtgebundenen Kommunikation ausgeschlossen sind. Im Fall eines mittels WLAN kommunizierenden Mikrocontrollers gibt es zum Beispiel die Störung des WLAN-Signals, über das die Befehle, die an den Mikrocontroller gesendet werden, fehlerhaft ankommen oder die Übertragung abbricht. Somit ist es essenziell, eine stabile und zuverlässige WLAN-Verbindung aufzubauen, die derartiges Verhalten vorbeugt.
Weiterhin gibt es, wie im vorhergehenden Unterkapitel erwähnt, elektromagnetische Emissionen, die die reguläre und ordnungsgemäße Funktionsweise beeinträchtigen und dafür sorgen, dass Defizite auf elektrischer Ebene geschehen, wodurch die Übertragung ebenso behindert oder vollständig verhindert wird.
Zudem gibt es neben elektrischen sowie elektronischen Bruchstellen und Herausforderungen auch menschliche Einflussfaktoren, die die Übertragungen beeinflussen: Durch einerseits das jeweilige WLAN-Sicherheitsprotokoll und andererseits die Kommunikationsprotokoll-Sicherheitsschicht ist die Kommunikationsschnittstelle geschützt. Dennoch gibt es die Möglichkeit, dass diese beiden Schichten wenig bis unzureichend eingerichtet wurden und es somit allen potenziellen menschlichen Gefahrenquellen erleichtert wird, Zugriff auf dieses System zu erhalten.

3 Anforderungen

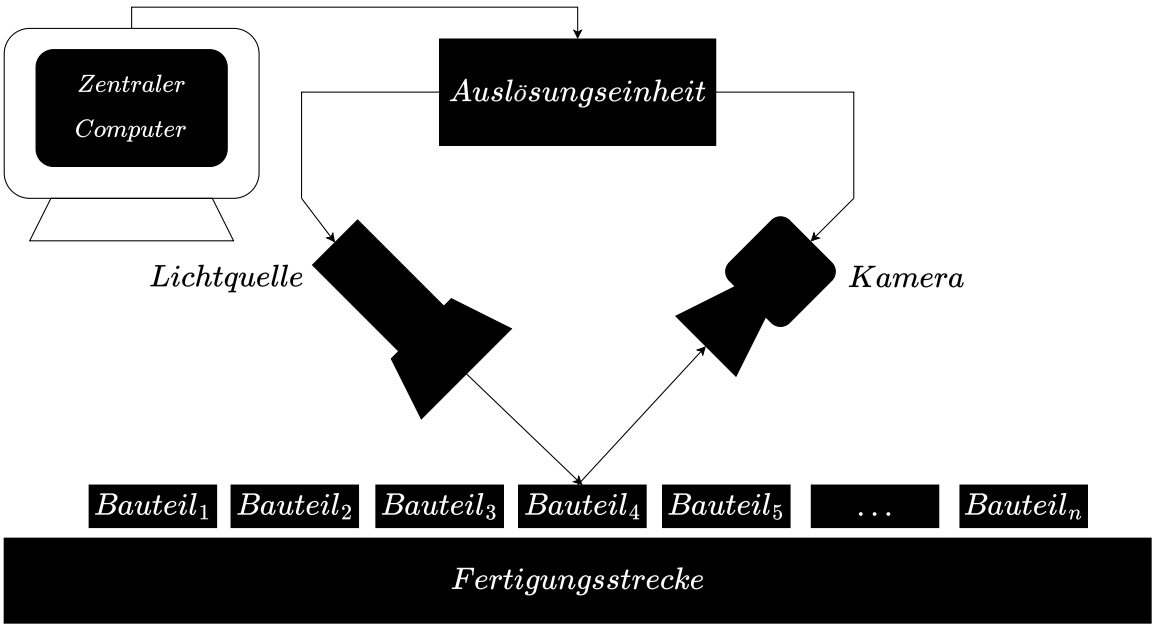


Abbildung 3.1: Grundlegender Ablauf, der visuellen Kontrolle, unter Verwendung der Lichtquelle, die das Licht emittiert (links), der Kamera, die eine visuelle Aufzeichnung des Bauteils erstellt (rechts), der Auslösungseinheit, die für die Ansteuerung der Lichtquelle und der Kamera verantwortlich ist (Mitte oben), und der Fertigungsstrecke, auf der die Bauteile transportiert und zur Prüfung bewegt werden

Durch den Aufbau, der auf der Abbildung 3.1 dargestellt ist, ergeben sich sechs Faktoren, die das Qualitätssicherungsverfahren beeinflussen:

1. Die Wiederholrate der Lichtemission und die damit verbundene Bildqualität der visuellen Aufzeichnung der Kamera.

2. Die Wiederholrate der Auslösung der Kamera - der Referenzwert, wie oft die Kamera, in einem bestimmten Zeitintervall, Aufzeichnungen erstellt.

3. Die daraus folgenden Eigenschaften der Auslösungseinheit - Die Taktfrequenz, die physischen Möglichkeiten mit der Auslösungseinheit zu kommunizieren (drahtgebunden oder drahtlos), die logischen Möglichkeiten mit der Auslösungseinheit zu kommunizieren (Programmiersprachen) sowie die Belastbarkeit und die Nutzerschaft, die sich mit der Auslösungseinheit beschäftigt, wodurch die Möglichkeit eines Austausches mit diesen Nutzern entsteht.

4. Die Geschwindigkeit der Fertigungsstrecke und damit einhergehend die Anforderung an die Kamera, an die Auslösungseinheit sowie an die Lichtquelle, ebenso mit dieser Geschwindigkeit zu arbeiten.

5. Die Einsatzumgebung - Die Zugänglichkeit zum verbauten System, die physikalischen Gegebenheiten während der Prüfung (Temperatur, Strahlung, Sauerstoffsättigung) und die Möglichkeit einer drahtlosen Kommunikation innerhalb des Prüfszenarios.

6. Die Bauteile - Je nachdem, welches Verfahren geprüft wird, ergeben sich unterschiedliche Genauigkeiten und Wiederholraten, unter denen die Bauteile geprüft werden.

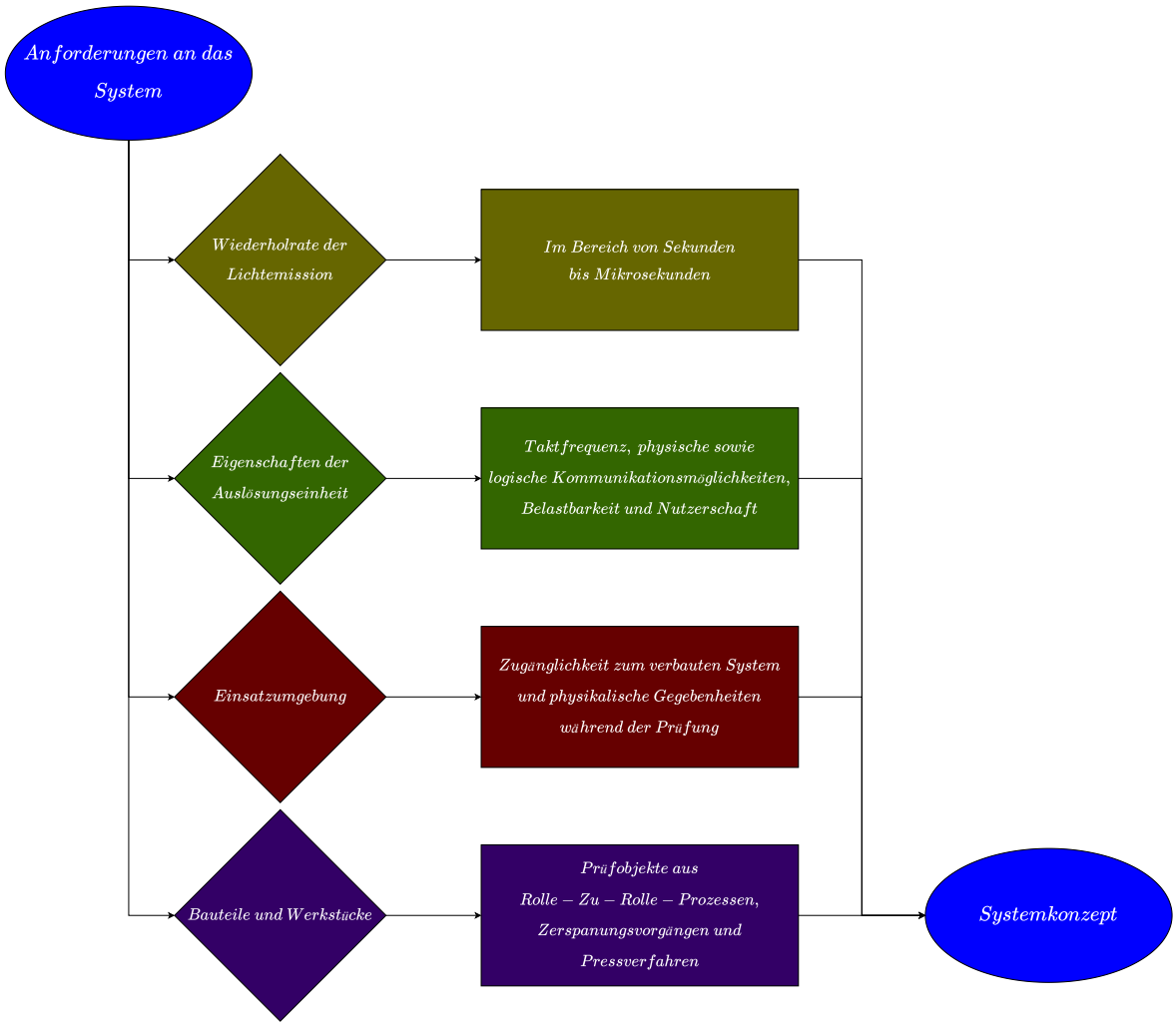


Abbildung 3.2: Grafische Darstellung der Strategie zur Konzeptionierung des Kommunikationssystems

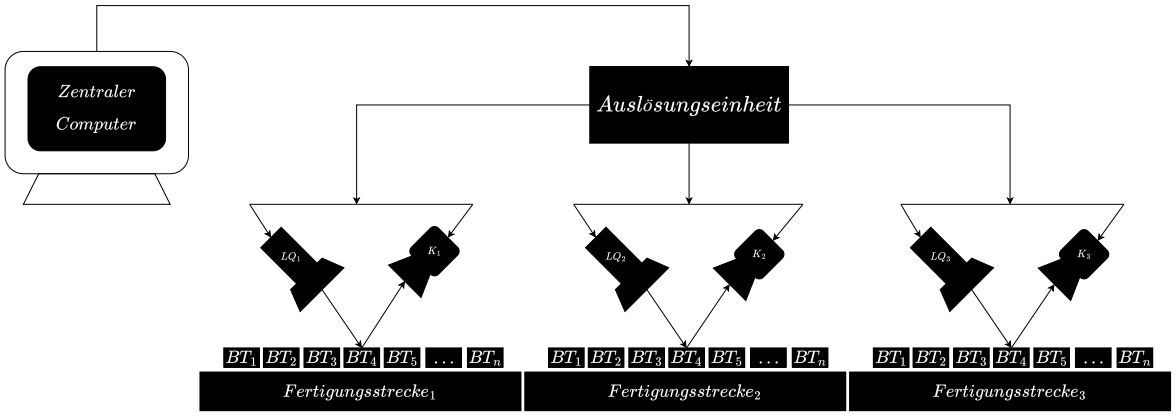


Abbildung 3.3: Aufbau eines parallel-prüfenden Systems, mit mehreren Qualitätssicherungsmodulen, die von einer Auslösungseinheit angesteuert werden

Auf der Abbildung 3.3 ist der beschriebene Aufbau zu sehen, bei dem zusätzlich zum grundlegenden Aufbau zwei weitere Qualitätssicherungsmodule durch die Auslösungseinheit angesteuert werden. Dieses Vorgehen dient der Skalierung und somit der Erhöhung der Geschwindigkeit des Qualitätssicherungssystems, weil dadurch mehrere Qualitätssicherungsmodule mittels einer Auslösungseinheit angesteuert werden.

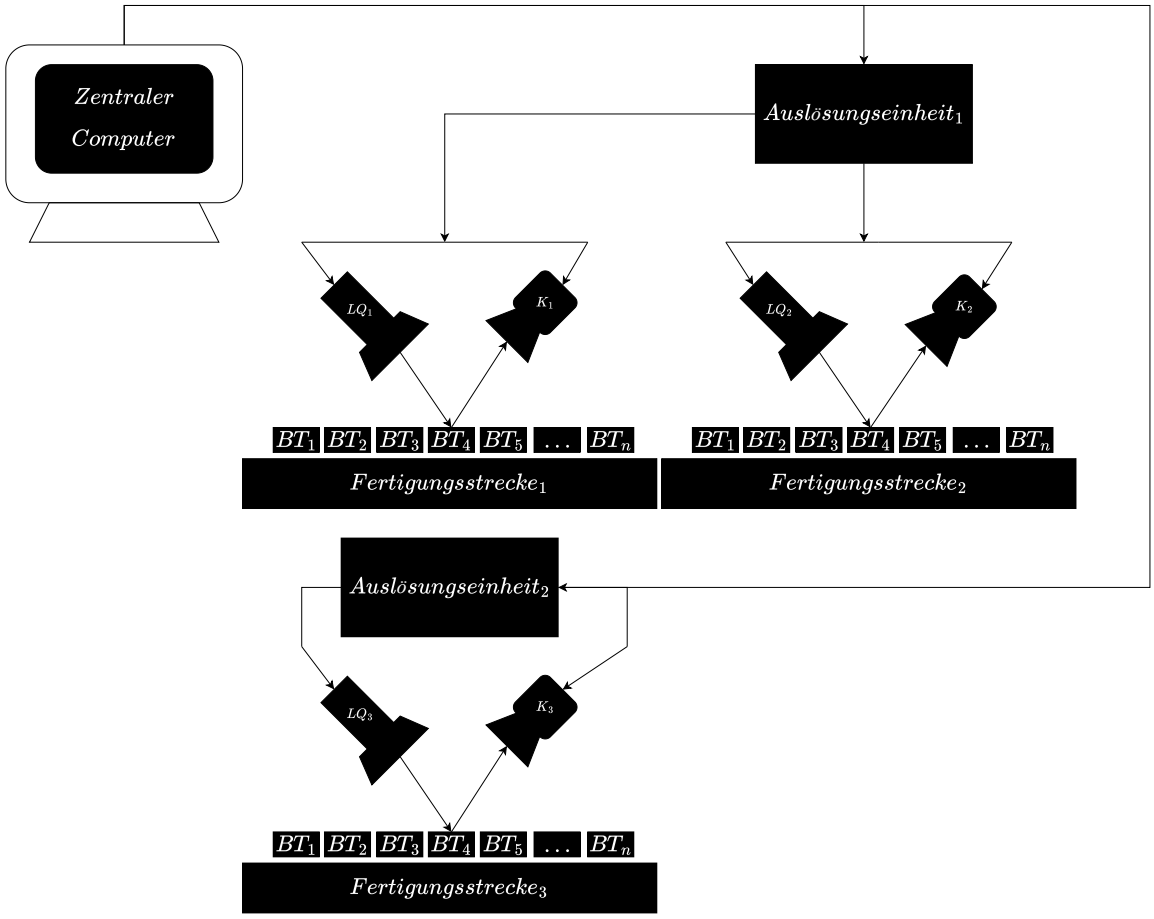


Abbildung 3.4: Aufbau eines zusammengesetzten und parallel-prüfenden Systems, mit mehreren Auslösungseinheiten, die Qualitätssicherungsmodule ansteuern

Auf der Abbildung 3.4 ist der Aufbau eines parallel-prüfenden Systems zu sehen, bei dem, zusätzlich zum Ablauf auf der Abbildung 3.3 eine weitere Auslösungseinheit, durch den zentralen Computer, angesteuert wird, die ebenso mit einem Qualitätssicherungsmodul verbunden ist. Dieses Vorgehen dient der Erhöhung der Geschwindigkeit des Qualitätssicherungssystems, weil dadurch mehrere Auslösungseinheiten und somit mehrere Qualitätssicherungsmodule angesteuert werden.

3.1 Anwendungsszenarien

Ein maßgebliches Beispiel für die Verwendung einer Auslösungseinheit innerhalb industrieller Umgebungen ist der Einsatz in räumlich geschlossenen Systemen. Hierbei wird die Auslösungseinheit wie auch das von der Auslösungseinheit angesteuerte System in eine Umgebung verbaut, wobei entweder sicherheitsbezogene oder räumliche Bedingungen dafür sorgen, dass der direkte Zugang zu der Auslösungseinheit verhindert wird. Dementsprechend steht die Anforderung einer drahtlosen Kommunikation und somit die weiterhin mögliche Ansteuer- sowie Programmierbarkeit im Vordergrund.

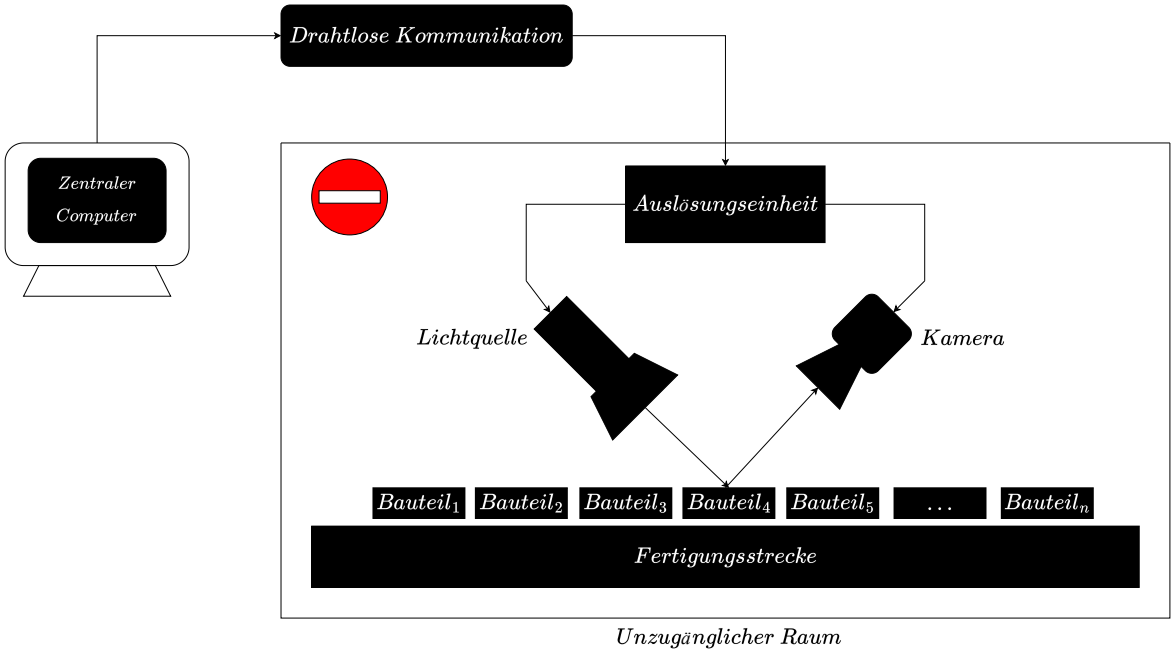


Abbildung 3.5: Anforderung, der drahtlosen Kommunikation, durch den Einsatz in unzugänglichen Systeme

Ein weiteres Beispiel für den Einsatz an unzugänglichen Orten ist die Verwendung bei Stationen oder Bereichen in für Menschen gesundheitsgefährdenden Zonen. Durch eine hohe Strahlenbelastung, hohe sowie niedrige Temperaturen oder andere Faktoren sind bestimmte Bereiche im Qualitätssicherungs- oder Industriesektor für Menschen gefährlich. Aufgrund dieser Gegebenheiten ergibt sich der Bedarf eines Systems, das in solchen Umgebungen ansteuer- sowie programmierbar ist. Dadurch wird die Gefahr beispielsweise durch eine hohe Strahlenbelastung umgangen und der Prozess bleibt weiterhin regulier- und kontrollierbar.

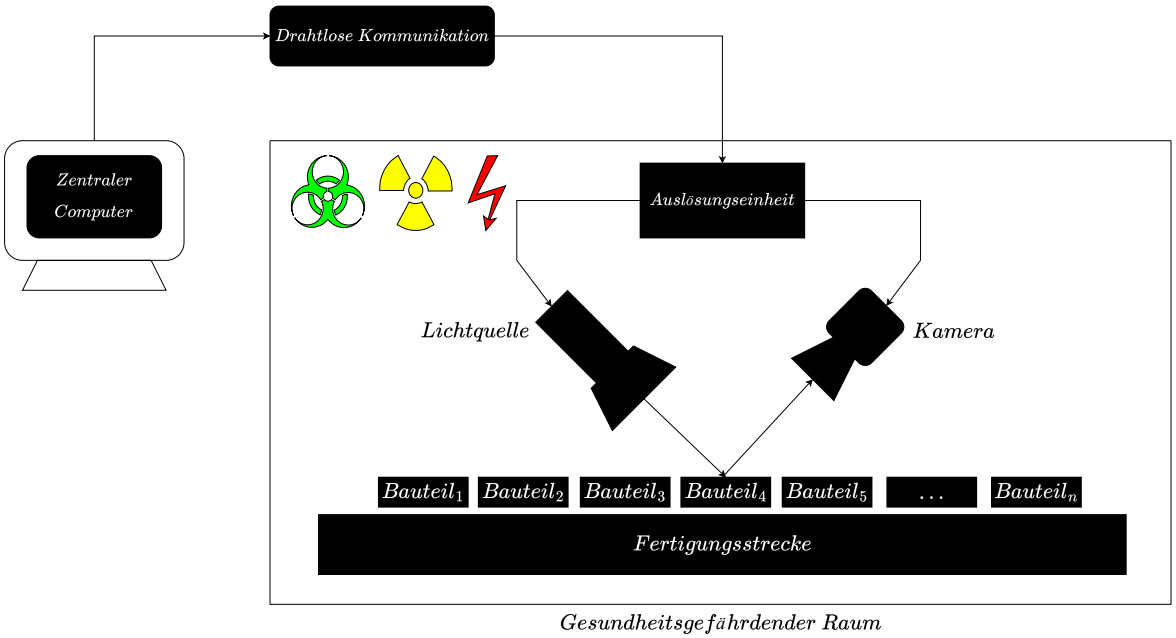


Abbildung 3.6: Anforderung, der drahtlosen Kommunikation, durch den Einsatz in gesundheitsgefährdenden Systeme

Ein drittes Beispiel, das in industriellen Qualitätssicherungsumgebungen zum Tragen kommt, sind die Limitationen, die bei kabelgebundenen Kommunikationsvarianten auftreten. Beispielsweise nutzen sich eingesetzte Materialien wie zum Beispiel Kabel mit der Zeit ab, wodurch die Möglichkeit besteht, dass eine fehlerhafte Verbindung entsteht oder die Verbindung sogar ganz entfällt. Ein weiteres Merkmal bei kabelgebundener Kommunikation ist der zusätzliche Einsatz von Komponenten wie Kabeln, um die Kommunikation zu ermöglichen, wodurch ein größerer Materialaufwand entsteht. Somit gilt dieser Aspekt unterstützend für die Anforderung einer drahtlosen Kommunikation.

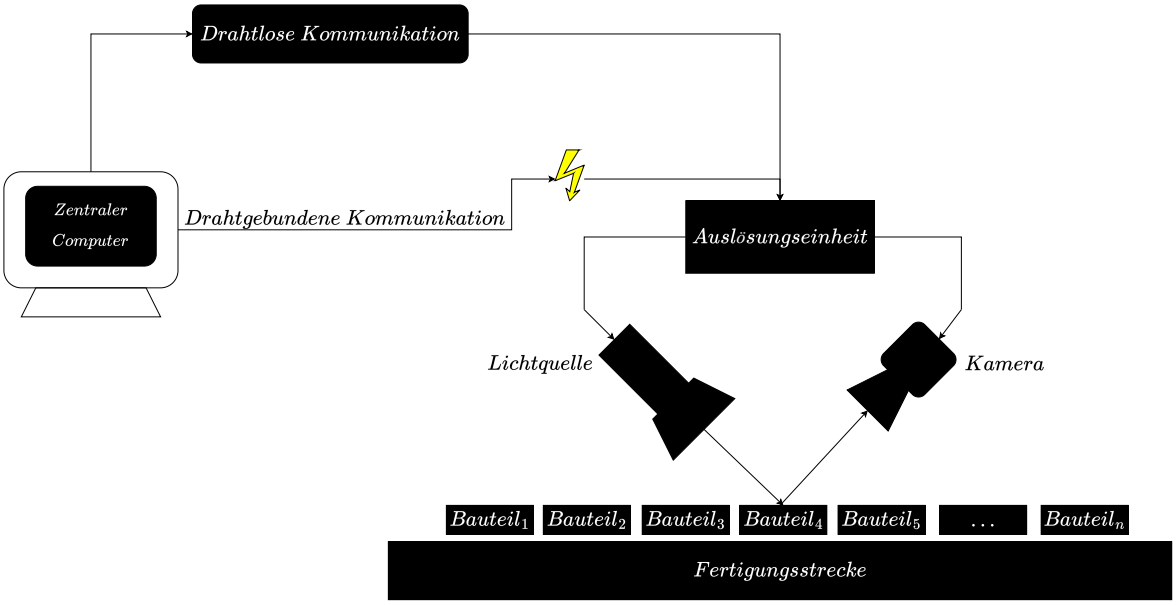


Abbildung 3.7: Anforderung, der drahtlosen Kommunikation, durch den Ausfall der drahtgebundenen Kommunikation

Zusätzlich ergibt sich durch den Einsatz mehrerer Lichtquellen, Kameras sowie Auslösungseinheiten die Herausforderung, alle Systeme präzise anzusteuern. Hierbei gilt es, alle Systeme von einer zentralen Schnittstelle anzusteuern und die größtmögliche Übersichtlichkeit sowie bestmögliche Ansteuerbarkeit zu erreichen. Durch diese drei genannten Faktoren wird die Anforderung einer drahtlosen Kommunikation zusätzlich unterstrichen.

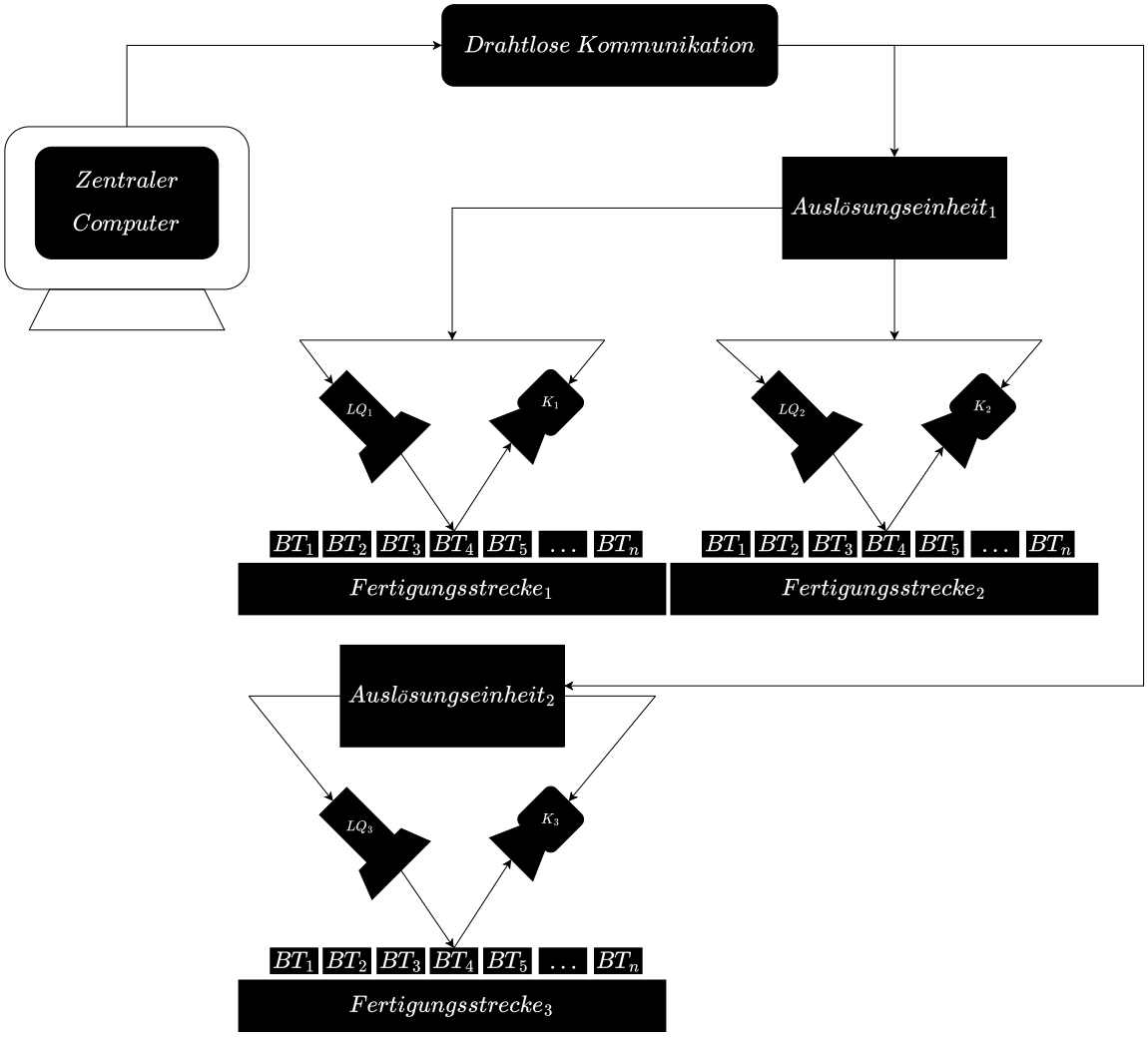


Abbildung 3.8: Anforderung, der drahtlosen Kommunikation, durch eine präzise Ansteuerung der eingesetzten Auslösungseinheiten

3.2 Prozessauswahl

3.2.1 Rolle-Zu-Rolle-Fertigungsbereich

Der grundlegende Ablauf eines Rolle-Zu-Rolle-Prozesses folgt dem Abwickeln der zu bearbeitenden Materialien, dem anschließenden Bearbeiten der ausgerollten Materialien und dem abschließenden Wiederaufrollen des Endproduktes. In Rolle-Zu-Rolle-Prozessen werden unter anderem Solarzellen, technische Folien, bedruckte oder beschichtete Textilien und Verpackungsmaterialien hergestellt. Hierbei entsteht durch die kontinuierliche Arbeitsweise der Maschinen und durch die eingesetzten Materialien eine hohe Durchsatzrate. Somit liegt das Zeitfenster für die Qualitätssicherung der zu prüfenden Materialien im Bereich von Mikrosekunden. Dabei ist

es wichtig, die höchstmögliche Genauigkeit zu erzielen. [Bun]

3.2.2 Zerspanungsbetrieb

Im Bereich der Zerspanungsfertigung gibt es mehrere Arten, Materialien und Werkstücke so zu bearbeiten, dass diese dem angeforderten Endprodukt entsprechen. Es gibt unter anderem die Möglichkeit des Fräsens der Bauteile, des Bohrens der Löcher der Werkstücke, des Drehens des Materials auf einer Drehbank und des Schleifens der Oberflächen der zu bearbeitenden Form. Dabei werden CNC-Maschinen eingesetzt, um die höchstmögliche Genauigkeit zu erzielen. Um diese Genauigkeit sowie die Qualität der bearbeiteten Materialien zu gewährleisten, ist hierbei ebenso das Höchstmaß an Genauigkeit der Prüfung erforderlich. Die Wiederholrate der Prüfung ist dabei genauso entscheidend, um den Betrieb und die Produktion so effizient wie möglich zu gestalten. Hierbei reichen die gebräuchlichen Intervalle, für die Prüfung der Bauteile, von Mikrosekunden bis Millisekunden. [DIN03a; DIN03b; DIN03i; DIN03j; DIN03k; DIN03l; DIN03m; DIN03n; DIN03o; DIN03p; DIN03c; DIN03d; DIN03e; DIN03f; DIN03g; DIN03h]

3.2.3 Pressverfahren

Unter Druckeinwirkung wird bei Pressverfahren das entsprechende Bauteil oder der Werkstoff so verändert, dass abschließend das Endprodukt die passende Form, Dichte oder Oberflächenstruktur aufweist. Vorgänge, die als Pressverfahren gelten, sind unter anderem das Schmieden, Strangpressen, Tiefziehen, Pulver- und Sinterpressen sowie Kaltpressen. Speziell um die Oberflächenstruktur zu überprüfen, bietet sich dabei ein Verfahren aus dem Bereich der visuellen Kontrolle an, mit dem es möglich ist, alle Unebenheiten oder Irritationen auf dem Endprodukt festzustellen. Für die Überprüfung der Oberflächenstruktur liegt der Zeitrahmen hierbei im Bereich von Sekunden. [DIN19]

3.3 Zusammenfassung der Anforderungen

Durch die Betrachtung der einzelnen Situationen, in denen die visuelle Kontrolle eingesetzt wird, die Darstellung des Potenzials, das mit der visuellen Kontrolle einhergeht, sowie die Erfassung der unterschiedlichen Anwendungsszenarien mit den einschränkenden Gegebenheiten der Einsatzumgebung ergeben sich folgende Anforderungen für die Entwicklung des Kommunikationssystems:

1. Das System soll drahtlos funktionieren.

2. Das System soll eine Verbindung zwischen der Auslösungseinheit und der Lichtquelle herstellen, sodass die Lichtquelle über die Auslösungseinheit angesteuert wird.

3. Das System soll die Möglichkeit bieten, mehrere Auslösungseinheiten zum gleichen Zeitpunkt anzusteuern.

4. Das System soll eine ausreichende Sicherheitsschicht bieten.

5. Das System soll eine präzise und fortlaufende Konfiguration der angesteuerten Auslösungseinheiten anbieten.

Diese Anforderungen werden im Laufe dieser Bachelorarbeit weiterhin behandelt und bei der Entwicklung des Kommunikationssystems berücksichtigt.

4 Methoden

4.1 Mikrocontrollerauswahl

Die grundlegenden Eigenschaften der Auslösungseinheit wie sie in Aufzählungspunkt 3 am Anfang des vorangegangenen Kapitels bei den Einflussfaktoren aufgezählt wurden waren Genauigkeit, Taktfrequenz und Ansteuerbarkeit. Aufgrund der Tatsache dass der ESP32-Mikrocontroller neu auf dem Markt erschienen war aufbauend auf den genannten Anforderungen und Merkmalen zum Abschluss des vorangegangenen Kapitels ebenso eine drahtlose Kommunikationsschnittstelle verbaut hatte und unter Berücksichtigung der genannten Eigenschaften erschien dieses Modell als angemessen für die Weiterentwicklung sowie die Implementierung eines effizienteren Systems auf der Grundlage eines Mikrocontrollers um die Aufgaben der Auslösungseinheit zu übernehmen. Als weiteres Entscheidungs- und Vergleichswerkzeug wurde die Tabelle 8.1 verwendet.

4.2 MicroPython

So wie es bei anderen Rechnereinheiten der Fall ist wird auch beim ESP32-Mikrocontroller eine Programmiersprache verwendet um die Anweisungen an den Mikrocontroller zu übertragen die er ausführen soll. Für die Ansteuerung des ausgewählten ESP32-Mikrocontrollers wurde die Programmiersprache MicroPython ausgewählt mit der es möglich war Python-Code in einer entsprechenden Entwicklungsumgebung zu schreiben diesen Python-Code auf den Mikrocontroller hochzuladen und ihn dann anschließend auf dem Mikrocontroller ausführen zu lassen [Geo25a]. MicroPython selbst ist wie beschrieben eine Programmiersprache sowie eine Abstraktionsschicht um Programmcode in Python zu schreiben der von Mikrocontrollern ausgeführt wird. Im Konkreten implementiert MicroPython die grundlegenden Aspekte von Python 3 und erlaubt zudem direkten Zugriff auf die Hardware des Mikrocontrollers mittels verschiedener Bibliotheken wie der „machine“-Bibliothek sowie der „mem32“-Bibliothek.
„MicroPython is written in C99 and the entire MicroPython core is available for general use under the very liberal MIT license.“ MicroPython wurde in C99 geschrieben und die fundamentalen Aspekte der Programmiersprache stehen unter der MIT-Lizenz [Geo25b]. „MicroPython is developed in the open on GitHub and the source code is available at the GitHub page and on the download page. Everyone is welcome to contribute to the project.“ Der Quellcode der Programmiersprache ist auf GitHub einsehbar [Geo05] und jeder ist dazu berechtigt an der Entwicklung mitzuwirken [Geo25a].

4.3 WebREPL

Der ESP32-Mikrocontroller bietet unter Verwendung MicroPythons eine Möglichkeit der drahtlosen Kommunikation mittels WebREPL an. WebREPL ist in dem Fall die Kommunikationsschnittstelle mit der es möglich ist drahtlos mit einem ESP32-Mikrocontroller zu interagieren. Dazu ist es fundamental das WebREPL-Kommunikationsprotokoll auf dem ESP32-Mikrocontroller zu aktivieren. Dies geschieht mittels MicroPython. Weiterhin ist ebenfalls in MicroPython ein WebREPL-Passwort zu vergeben das bei jedem Verbindungsaufbau mit dem ESP32-Mikrocontroller anzugeben ist. Dieses WebREPL-Passwort wird in einer einzelnen Datei auf dem ESP32-Mikrocontroller abgespeichert in der ausschließlich das Passwort eingetragen wird. Somit ist ein drahtloser Verbindungsaufbau mit einem einzelnen ESP32-Mikrocontroller über WebREPL möglich wenn der ESP32-Mikrocontroller entsprechend konfiguriert ist.

4.4 MQTT

Durch die Tatsache bedingt dass das Qualitätssicherungssystem vornehmlich in industriellen Umgebungen eingesetzt wird und Mikrocontroller die Aufgaben der Auslösungseinheiten übernehmen ergeben sich entsprechende Rahmenbedingungen für die Übertragung der Anweisungen an die einzelnen Mikrocontroller: Eine schwankende Bandbreite resultierend aus der drahtlosen Kommunikation eine eingeschränkte Hardwarekapazität durch den Einsatz der Mikrocontroller und eine angemessene Sicherheitsschicht um den ordnungsgemäßen Qualitätssicherungsbetrieb zu gewährleisten sowie mögliche unbefugte Eingriffe in das System zu verhindern. Auf Basis dieser drei Kriterien wurden verschiedene Netzwerkprotokolle untersucht. Am besten eignete sich MQTT für die Entwicklung des Kommunikationssystems um die Anweisungen an die Mikrocontroller zu übertragen. Gemäß [SN25] bietet MQTT Unterstützung für schwankende Netzwerke an: „Many IoT devices connect over unreliable cellular networks. MQTT's support for persistent sessions reduces the time to reconnect the client with the broker.“ Viele IoT-Geräte verbinden sich über schwankende Mobilfunknetzwerke. MQTT schafft hierbei die Unterstützung für dauerhafte Verbindungssitzungen und reduziert somit die Zeit wieder eine Verbindung mit dem „Broker“ aufzubauen. Weiterhin ist MQTT schlank und effizient: „MQTT clients are very small require minimal resources so can be used on small microcontrollers. MQTT message headers are small to optimize network bandwidth.“ MQTT-Clients sind sehr kompakt benötigen minimale Ressourcen und bieten so die Möglichkeit bei kompakten Mikrocontrollern eingesetzt zu werden. MQTT-Nachrichten haben einen geringen Metadatenanhang um so die Netzwerkbandbreite optimal zu nutzen. Zusätzlich bietet MQTT eine Sicherheitsschicht an: „MQTT makes

it easy to encrypt messages using TLS and authenticate clients using modern authentication protocols such as OAuth“ MQTT erleichtert das Verschlüsseln von Nachrichten durch TLS und authentisiert „Clients“ über moderne Authentisierungsprotokolle wie OAuth. Abschließend ist auf der Abbildung 4.1 eine beispielhafte Kommunikation dargestellt die zwischen einem MQTT-Broker und mehreren MQTT-Clients abläuft.

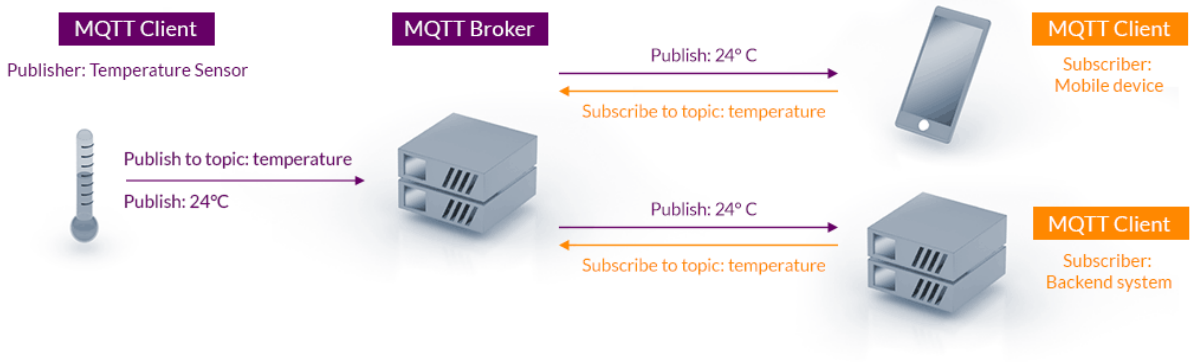


Abbildung 4.1: Grundlegende Kommunikation, zwischen einem MQTT-Broker und mehreren MQTT-Clients, beispielhaft dargestellt [SN25]

4.5 Systemarchitektur (Hardware, Software, Kommunikationsschnittstellen)

In Unterkapitel 3.1 wurden die Anwendungsszenarien definiert die als grundlegend und vordergründig für den Einsatz des Kommunikationssystems gelten. Um die definierten Anwendungsszenarien zu erreichen den ausgewählten ESP32-Mikrocontroller die beschriebene Abstraktionsschicht sowie Programmiersprache MicroPython und die drahtgebundene sowie drahtlose Möglichkeit der Kommunikation mittels MQTT miteinander zu verbinden wurde eine entsprechende Systemarchitektur entworfen welche die genannten Aspekte integriert: Auf den Mikrocontroller selbst wird MicroPython aufgespielt indem der Flash-Speicher des Mikrocontrollers gelöscht und die entsprechende MicroPython-Firmware-Datei auf den Mikrocontroller hochgeladen wird. Für die drahtgebundene Kommunikation besteht die Möglichkeit mittels eines USB-Kabels eine Verbindung zu dem Mikrocontroller aufzubauen. Für die drahtlose Kommunikation wird MQTT eingesetzt wobei entsprechende Vorbereitungen unter Verwendung MicroPythons und des Mikrocontrollers vorzunehmen sind wie es in Unterkapitel 5.2 beschrieben wird. Somit ergeben sich bei der eingesetzten Hardware (ESP32-Mikrocontroller) unter Verwendung der eingesetzten Software (MicroPython) die Möglichkeiten der drahtgebundenen Kommunikation (USB-Kabel) und drahtlosen Kommunikation (MQTT).

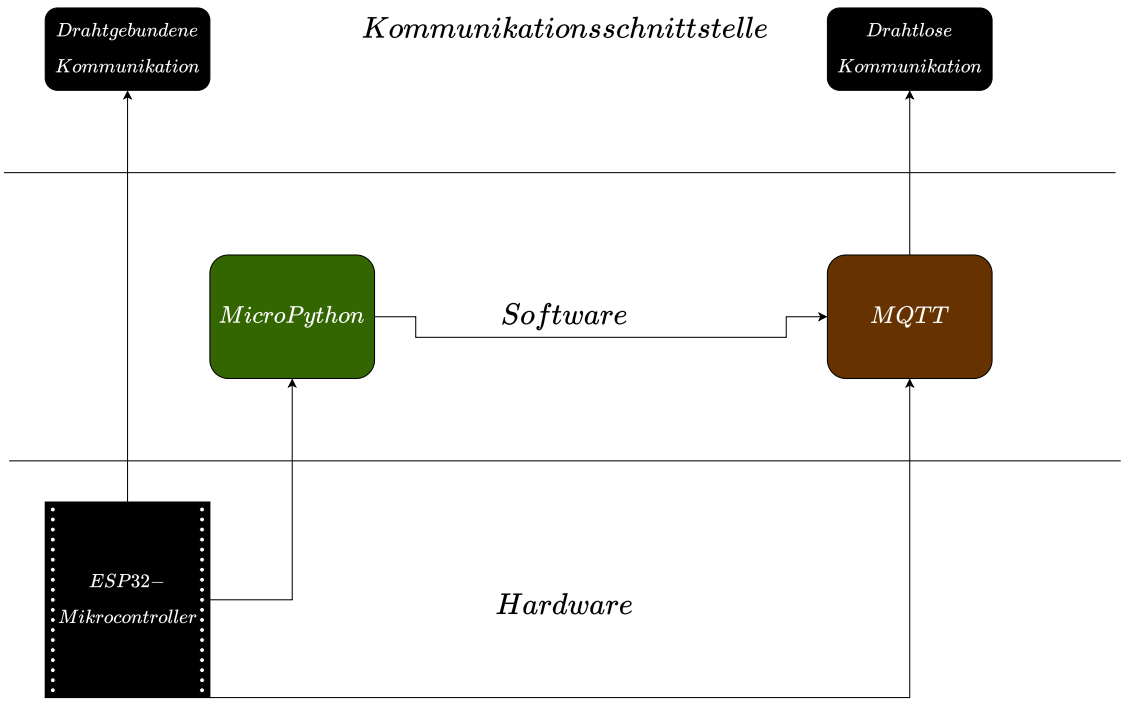


Abbildung 4.2: Systemarchitektur, unter Verwendung der eingesetzten Hardware, Software und den Kommunikationsschnittstellen

5 Experimentelle Umsetzung

Zum Abschluss des Aufbaus steht ein Qualitätssicherungssystem in Aussicht mit dem eine höhere Genauigkeit und Wiederholrate der Aufzeichnungen im gleichen Fertigungsbereich erzielt wird. Für die Erreichung dieser Vorgabe wurden sich verschiedene Mikrocontroller-Technologien angeschaut und für den Aufbau in Betracht bezogen. Die Entscheidung fiel wie in der Tabelle 8.1 dargestellt auf den ESP32-Mikrocontroller weil dieser Mikrocontroller die performantesten Ergebnisse in einem angemessenen Preis-Leistungs-Verhältnis lieferte. Zudem besitzt dieser Mikrocontroller die Möglichkeit der drahtlosen Kommunikation das ebenso einen Vorteil gegenüber anderen Mikrocontrollern darstellt.
In Kombination mit dem Mikrocontroller kommt ein Lichtemissionsgerät zum Einsatz das dafür sorgt entsprechend der eingestellten Wiederholrate Licht zu emittieren sodass für die angebrachten Kameravorrichtungen die Möglichkeit entsteht visuelle Aufzeichnungen der Prüfteile zu erstellen und weiter an die Schnittstelle zu senden die mittels beispielsweise künstlicher Intelligenz die Qualitätsüberprüfung durchführt.
Der Mikrocontroller hat hierbei die Aufgabe das Lichtemissionsgerät zum entsprechenden Zeitpunkt zu aktivieren sodass es pünktlich zum Prüfen auslöst und alle Lichtemissionsgeräte synchron zu halten um den parallelen Qualitätssicherungsvorgang zu gewährleisten. Bei diesem Aufbau kommen mehrere ESP32-Mikrocontroller zum Einsatz weswegen eine Lösung für die Ansteuerung mehrerer Mikrocontroller gefunden wurde: MQTT. MQTT als Client-Broker-Dienst sorgt dafür dass jeder Mikrocontroller der ein dafür vordefiniertes Thema abonniert hat den Inhalt jeder Nachricht die über dieses Thema versendet wird ausführt. Somit abonnieren in der Vorbereitung alle beteiligten ESP32-Mikrocontroller ein vorher konfiguriertes Thema und führen im Anschluss jeden Befehl aus der über dieses Thema gesendet wird. Für den Fall der Notwendigkeit bei einzelnen ESP32-Mikrocontrollern Veränderungen Aktualisierungen oder Zurücksetzungen vorzunehmen gibt es die besonders für diesen Anwendungsfall entwickelte Bibliothek mit der über die IP-Adresse und das entsprechende WebREPL-Passwort auf einen einzelnen ESP32-Mikrocontroller zugegriffen wird und diesem über eine Konsoleneingabe Python-Befehle gesendet werden die der Mikrocontroller direkt ausführt.

5.1 Vorgehen

Die grundlegende Konzeptionierung des Kommunikationssystems ist abgeschlossen. Um den Aufbau zu realisieren steht als Nächstes die Implementierung beziehungsweise die programmatische Umsetzung sowie die Zusammenführung aller genannten Aspekte aus dem Unterkapitel 3.3

und 4.5 an. Zudem gibt es einen Schaltplan auf dem vermerkt ist welche Pins anzusteuern sind um das Aktivieren Deaktivieren sowie die Intensität der eingesetzten LED-basierten Lichtquelle anzusteuern (siehe Unterkapitel 8.2). Auf Basis dieser Grundlagen ist ein Programm entstanden das in den folgenden Abschnitten näher beleuchtet wird.

5.2 Gestaltung

In Listing 1 ist das Programm zu sehen das dafür verantwortlich ist die entsprechenden Ports des ESP32-Mikrocontrollers anzusteuern um so die LED-basierte Lichtquelle zu aktivieren oder deaktivieren. Zudem ist hierbei ebenso die sogenannte „PWM“ also die Pulsweitenmodulation und somit die Intensität der Lichtquelle eingestellt.
Um auf die für die Ansteuerung und Programmierung der Lichtquelle notwendigen Bibliotheken zuzugreifen stehen in den Zeilen 1 - 8 die entsprechenden Import-Anweisungen der erforderlichen Bibliotheken.
Die Anforderung der drahtlosen Kommunikation mit dem ESP32-Mikrocontroller stellt eine WLAN-Verbindung in Aussicht die in den Zeilen 10 - 21 definiert und konfiguriert ist.
Die angesprochene Anpassung der Pulsweitenmodulation ist in der Zeile 23 beschrieben wobei der achtzehnte Pin für die Ansteuerung verwendet ist.
Um die Aktivierung beziehungsweise Deaktivierung der Lichtquelle zu gewährleisten sind in den Zeilen 25 sowie 26 die Pins definiert und mit den entsprechenden Parametern ausgestattet sodass sie ordnungsgemäß ihre Funktion ausfüllen.
Die Aktivierung sowie Deaktivierung der Ports welche die Pins zur Verfügung stellen erfolgt nach der Ausführung der Methode die in den Zeilen 28 - 37 zu sehen ist. Um eine saubere Trennung der beiden Funktionen des Aktivierens wie Deaktivierens herzustellen gibt es jeweils eine Hilfsmethode für das Anschalten der Lichtquelle hierbei gelistet in den Zeilen 39 und 40 sowie eine Methode für das Ausschalten hierbei gelistet in den Zeilen 42 und 43 wobei hierfür die zuvor angesprochene Hilfsmethode mit den erforderlichen Pins zum Greifen kommt.
Auf den Zeilen 45 - 48 ist eine Methode definiert die entsprechend dem als Parameter übergebenen Wert die Lichtquelle einschaltet für diesen Wert die weitere Abarbeitung des Programms pausiert und schließlich die Lichtquelle wieder ausschaltet. Der Wert ist hierbei in Sekunden anzugeben.

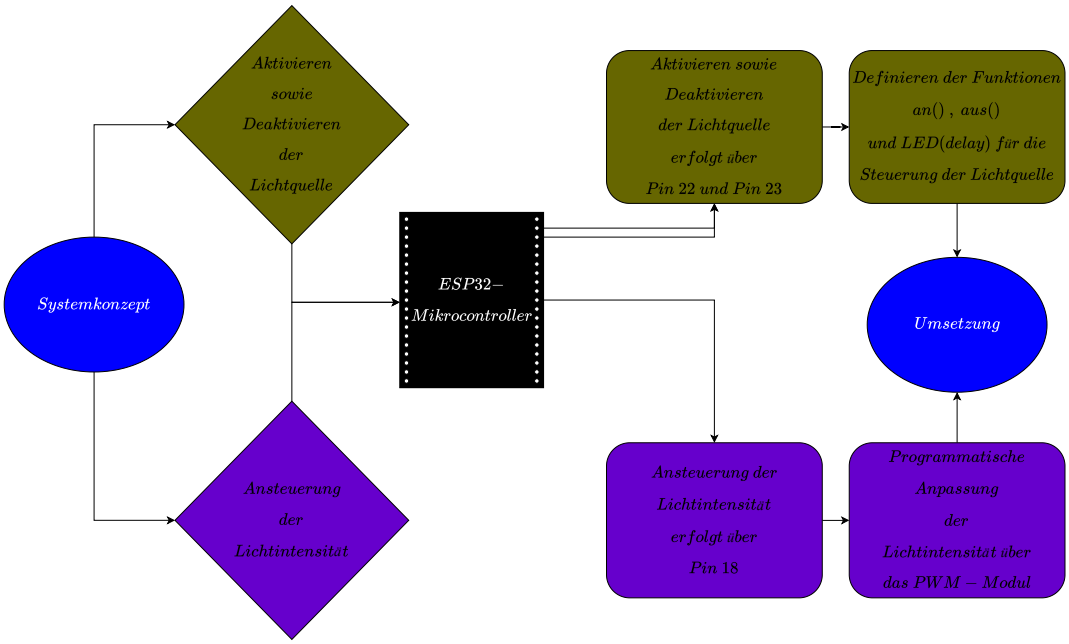


Abbildung 5.1: Anforderungen sowie die Pin-basierte Ansteuerung, der zwei grundlegenden Funktionen des ESP32-Mikrocontrollers, in einem modifizierten FlowChart-Diagramm dargestellt

Ein modifiziertes FlowChart-Diagramm zur Visualisierung der Anforderungen sowie der mit den beschriebenen Pins realisierten Umsetzung ist auf der Abbildung 5.1 zu sehen.

Nachdem die Funktionen der Ansteuerung sowie der Intensität der Lichtquelle erläutert sind geht es als Nächstes um die Kommunikation und die Übertragung der Anweisungen an den Mikrocontroller sodass der Mikrocontroller die erhaltenen Anweisungen nach Eingabe ausführt.
In Zeile 51 erfolgt die Aktivierung des WebREPL-Kommunikationsprotokolls. Das ist wichtig um die drahtlose Kommunikation zu ermöglichen.
Als weiterer wichtiger Baustein für die drahtlose Kommunikation ist eine MQTT-Verbindung aufzubauen wobei einmal ein eindeutiger Schlüssel für das Gerät das mit dem MQTT-Broker kommuniziert die Adresse des verwendeten MQTT-Brokers sowie das Thema über das die Nachrichten im Laufe der Kommunikation verschickt werden anzugeben sind. Hierfür befinden sich die entsprechenden Anweisungen auf den Zeilen 54 - 56.
Um den durch die drahtlose Kommunikation übertragenen Code schließlich auszuführen befindet sich auf den Zeilen 58 - 63 eine Hilfsmethode die einerseits eine Rückmeldung über den Code gibt der übertragen wurde und andererseits entweder den übertragenen Code direkt ausführt oder eine Fehlermeldung anzeigt wenn es bei der Ausführung zu Komplikationen kommt.
Weitere für die Kommunikation grundlegende Konfigurationen sind auf den Zeilen 65 - 68

vorgenommen. Dabei ist die Erstellung eines MQTT-Client der erste Schritt. Dieser MQTT-Client hat die Aufgabe mit dem MQTT-Broker zu kommunizieren. Für die Erstellung ist einmal der eindeutige Schlüssel des Geräts als auch die Adresse des MQTT-Brokers anzugeben. Danach folgt das Verknüpfen des MQTT-Clients mit der zuvor beschriebenen Hilfsmethode. Der dritte Schritt ist das Verbinden des MQTT-Clients mit dem Server. Nachdem der MQTT-Client mit dem Server verbunden ist abonniert er das vorher definierte Thema um über dieses Thema die Anweisungen zu empfangen die es auszuführen gilt. Für die Eingabe dieser Anweisungen eignet sich hierbei jedes Endgerät das Zugang zu dem MQTT-Broker hat und dem das Thema bekannt ist das der MQTT-Client abonniert hat.
In Zeile 70 steht die Anweisung mit der zum Start des Programms auf der Konsole ausgegeben ist dass alles ordnungsgemäß funktioniert und der MQTT-Client für die Ausführung von Anweisungen bereit ist. Das ist wichtig weil der Endnutzer so die Möglichkeit hat zu prüfen ob eine intakte Verbindung zwischen dem MQTT-Client und dem MQTT-Broker besteht.
Um über das WebREPL-Kommunikationsprotokoll zu kommunizieren findet eine entsprechende Erstellung eines WebREPL-Client in Zeile 72 statt. Bei dieser Erstellung sind einerseits die IP-Adresse über welche die Kommunikation stattfindet sowie das WebREPL-Passwort übergeben.
Ein durchgehendes und fortlaufendes Registrieren sowie Verarbeiten durch den drahtlosen Kommunikationsweg übertragener Befehle erfolgt durch die Zeilen 74 - 79. Hierbei sorgt eine while-Schleife dafür dass der MQTT-Client zuerst prüft ob neue Anweisungen angekommen sind und danach erfolgt ein Aufruf der „sleep“-Methode um dem Mikrocontroller die Möglichkeit zu geben angekommene Anweisungen abzuarbeiten. Die while-Schleife selbst steht in einem try-Block um die Verbindung des MQTT-Client jederzeit mittels „finally“ wieder aufzulösen wenn es zu Komplikationen oder Störungen bei der Übertragung kommt.

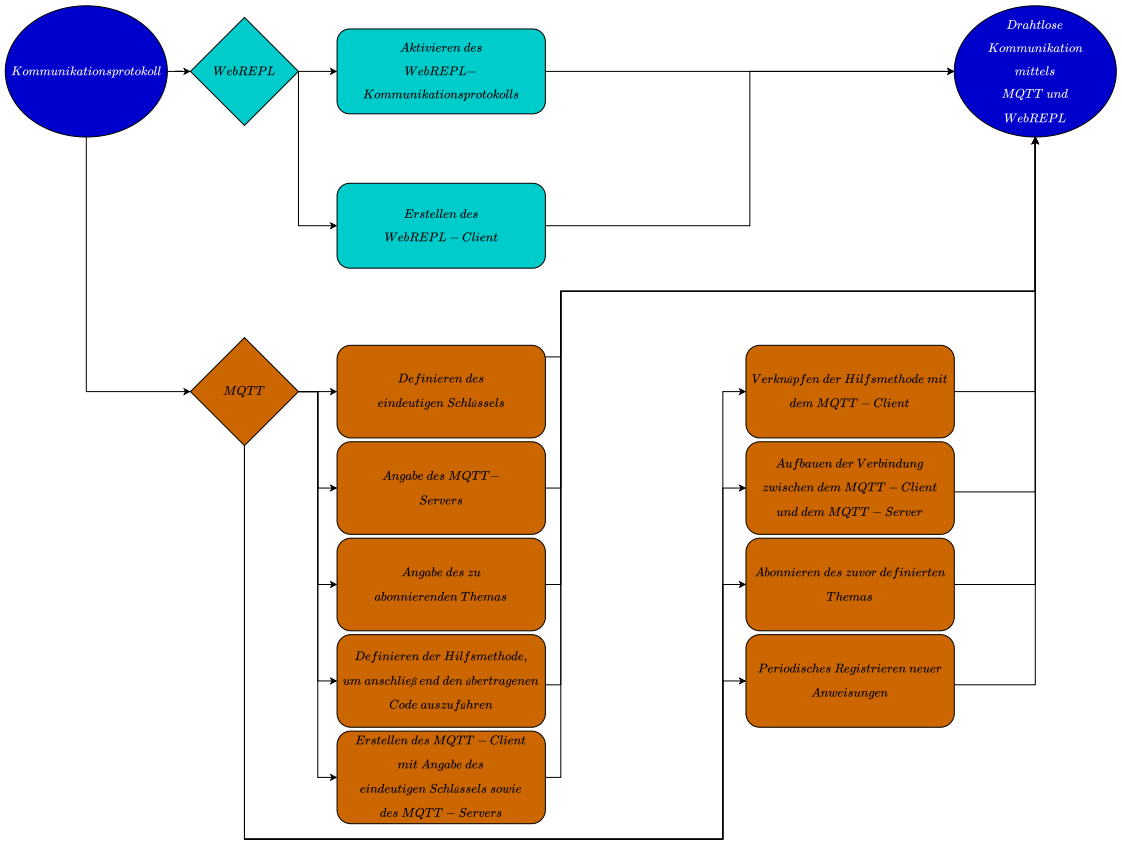


Abbildung 5.2: Anforderungen, und die entsprechenden Programmschritte, zur Realisierung einer drahtlosen Kommunikation, mittels MQTT und WebREPL, in einem FlowChart-Diagramm zusammengefasst

5.2.1 Ablauf

Der ESP32-Mikrocontroller führt jede Anweisung aus die ihm übertragen wird. Hierzu gilt es die entsprechenden Wege wahrzunehmen Anweisungen an den Mikrocontroller zu übertragen. Um dem Mikrocontroller Anweisungen zu übertragen die dieser dann anschließend ausführt gibt es zwei Möglichkeiten: Die drahtgebundene und die drahtlose Kommunikation.

Bei der drahtgebundenen Kommunikation wird ein USB-Kabel verwendet um die entsprechenden Anweisungen mittels einer geeigneten Umgebung zu übertragen. Im Rahmen dieser Arbeit wurde die Entwicklungsumgebung „Thonny“ für die drahtgebundene Übertragung angesprochener Anweisungen genutzt weil diese Entwicklungsumgebung sowohl den Zugriff auf das interne Dateisystem des ESP32-Mikrocontrollers gewährt als auch eine Konsole zur Verfügung stellt mit der Anweisungen und Befehle direkt an den ESP32-Mikrocontroller übertragen werden. Der Zugriff auf das interne Dateisystem des ESP32-Mikrocontrollers ist hierbei grundlegend um beispielsweise eine Boot-Datei wie sie in Listing 1 zu sehen ist zu erstellen und auf dem

ESP32-Mikrocontroller abzulegen sodass dieser bei jedem Neustart oder Zurücksetzen des Systems den Inhalt dieser Datei automatisch ausführt.

Um nachfolgend eine Anweisung über den drahtgebundenen Weg an den ESP32-Mikrocontroller zu übertragen sodass dieser die Anweisung ausführt ist zunächst die Umgebung aufzurufen mit der die Anweisungen eingegeben und übertragen werden. Danach sind zwei Verbindungen aufzubauen: Einerseits die physische Verbindung zwischen dem Endgerät das die Anweisungen überträgt und dem ESP32-Mikrocontroller. Andererseits ist eine logische Verbindung zwischen der Umgebung in der die Anweisungen eingegeben und übertragen werden und dem ESP32-Mikrocontroller aufzubauen sodass eine Schnittstelle besteht über welche die Kommunikation stattfindet. Im Fall der Entwicklungsumgebung Thonny ist der entsprechende Kommunikationsport auszuwählen über den die drahtgebundene Kommunikation stattfindet. Darauf folgt der Aufruf der Konsole in der Umgebung um die entsprechende Anweisung einzugeben und anschließend zu übertragen. Nachdem die Anweisung eingegeben und an den ESP32-Mikrocontroller mittels des USB-Kabels transportiert ist setzt der ESP32-Mikrocontroller die übertragene Anweisung direkt um. Somit lässt sich je nach Anweisung die Veränderung angeschlossener Peripherie wie zum Beispiel eines Sensors oder einer LED-basierten Lichtquelle direkt beobachten. Für das Suchen von fehlerhaftem Code oder das Ausgeben einzelner Werte des Mikrocontrollers selbst gibt es zudem die Möglichkeit wie es bei Entwicklungsumgebungen anderer Programmiersprachen der Fall ist Anweisungen über die Konsole einzugeben sodass diese Anweisungen für eine Ausgabe auf der Konsole sorgen. Die Ergebnisse der Ausgabe werden direkt auf der Konsole selbst angezeigt.

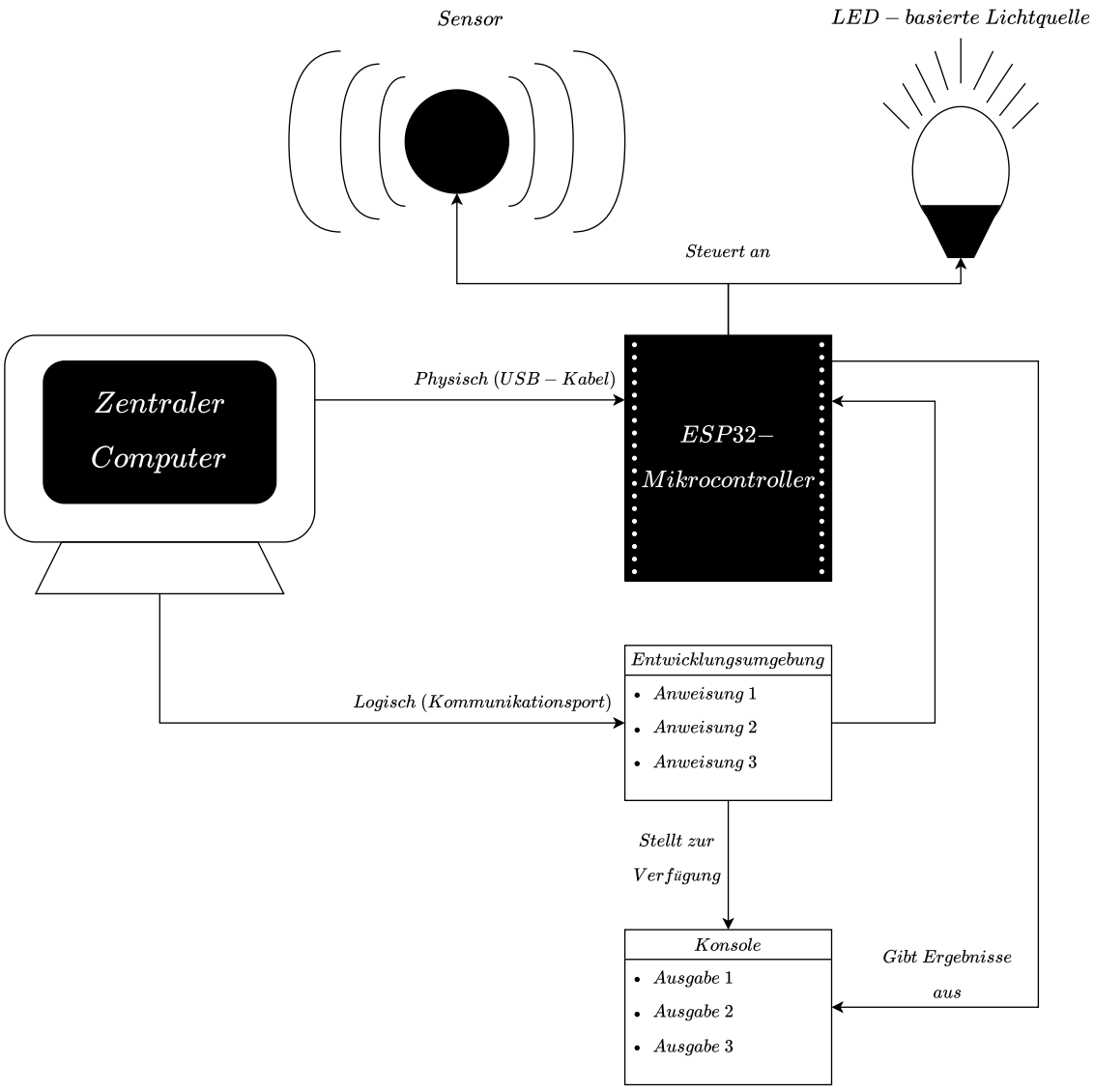


Abbildung 5.3: Grundlegende Elemente, die, bei einer drahtgebundenen Kommunikation, zum Einsatz kommen

Bei der drahtlosen Kommunikation wird das in Unterkapitel 5.2 vorgestellte Kommunikationsprotokoll verwendet das sowohl MQTT als auch WebREPL für die Übertragung der Anweisungen einsetzt.

Um hierbei eine Anweisung über den drahtlosen Weg an den ESP32-Mikrocontroller zu übertragen sodass dieser die Anweisung ausführt ist zunächst eine Anpassung der Boot-Datei vorzunehmen um eine Boot-Datei auf dem ESP32-Mikrocontroller abzulegen wie sie beispielsweise in Listing 1 zu sehen ist. Die Anpassung gilt hierbei vor allem der Konfiguration des WebREPL-Kommunikationsprotokolls sowie des nachfolgenden Erstellens und Einrichtens der

MQTT-Schnittstelle um eine drahtlose Kommunikation herzustellen. Die einzelnen Schritte um den ESP32-Mikrocontroller für eine drahtlose Kommunikation vorzubereiten sind zusätzlich auf der Abbildung 5.2 visuell aufbereitet. Nach der Vorbereitung des ESP32-Mikrocontrollers ist eine passende Anwendung aufzurufen mit der es möglich ist eine Verbindung mit einem MQTT-Broker herzustellen und über diese Verbindung eine Nachricht an ein vorher definiertes Thema zu versenden. In dieser Anwendung ist folgend die angesprochene Verbindung mit dem MQTT-Broker herzustellen. Anschließend gilt es das entsprechende Thema anzugeben und zu abonnieren über das die Kommunikation mit dem ESP32-Mikrocontroller stattfindet. Um eine Anweisung einzugeben und an den ESP32-Mikrocontroller zu übertragen ist abschließend eine Nachricht über das zuvor angegebene und abonnierte Thema zu versenden. Der Inhalt dieser Nachricht ist die Anweisung die der ESP32-Mikrocontroller nach Erhalt der Nachricht ausführt. Aufbauend darauf gibt es auch hier die Möglichkeit alle Ausgaben die entsprechend den drahtlos übertragenen Anweisungen generiert und kommuniziert werden über eine passende Umgebung hierfür eignet sich ebenso die Entwicklungsumgebung Thonny auf einer Konsole anzuzeigen.

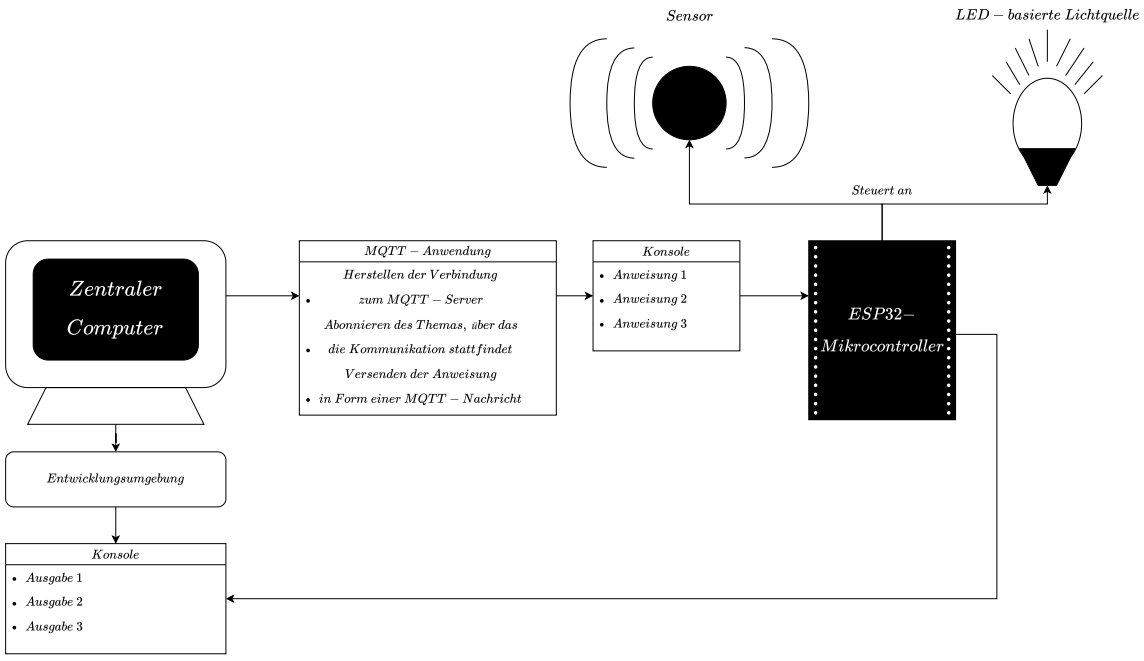


Abbildung 5.4: Grundlegende Elemente, die, bei einer drahtlosen Kommunikation, zum Einsatz kommen

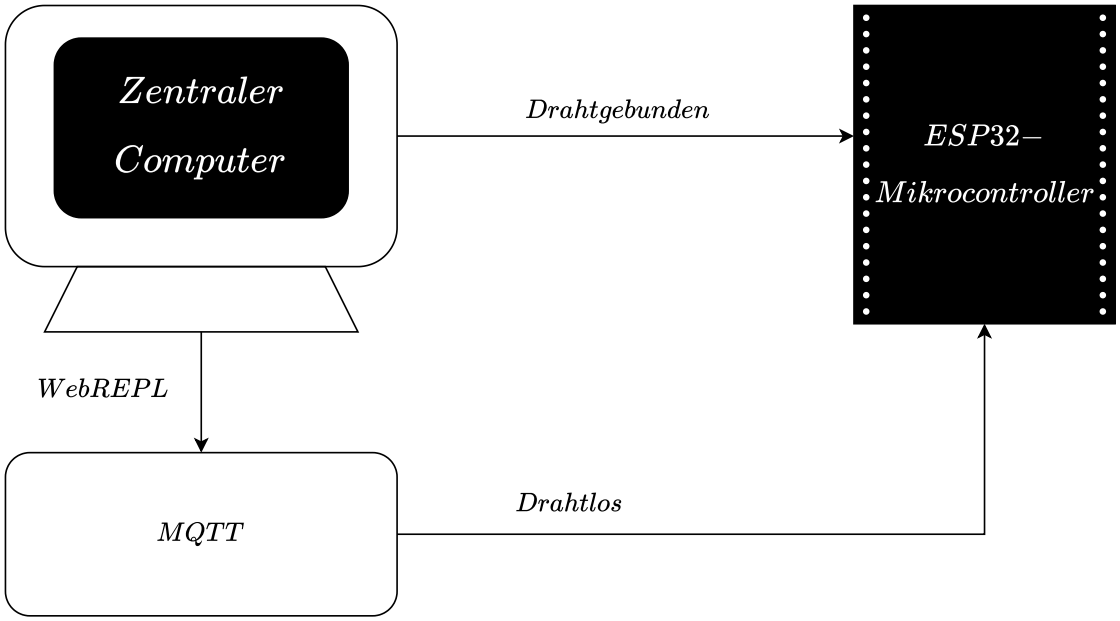


Abbildung 5.5: Kommunikationsmöglichkeiten mit dem ESP32-Mikrocontroller

Auf der Abbildung 5.5 sind beide Kommunikationsmöglichkeiten zu sehen und mit den entsprechenden „Transportmedien“ abgebildet. Im Fall der drahtgebundenen Kommunikation ist das „Transportmedium“ das USB-Kabel um die Anweisungen einer zentralen Schnittstelle an den ESP32-Mikrocontroller zu übertragen. Im Fall der drahtlosen Kommunikation sind WebREPL sowie MQTT die Transportmedien um die Anweisungen einer zentralen Schnittstelle an den ESP32-Mikrocontroller zu übertragen.

5.2.2 Aufbau

Um nun den ESP32-Mikrocontroller in Kombination mit einer LED-basierten Lichtquelle einzusetzen ist im Laufe des Bearbeitungs- und Entwicklungsprozesses ein zusammengesetztes System entstanden das die Ansteuerung der Lichtquelle ermöglicht. Dieses zusammengesetzte System besteht hauptsächlich aus dem ESP32-Mikrocontroller Stromanschluss für die Energieversorgung des Systems der LED-basierten Lichtquelle und den elektrischen Komponenten die für die Verbindung zwischen den drei Elementen zuständig sind.

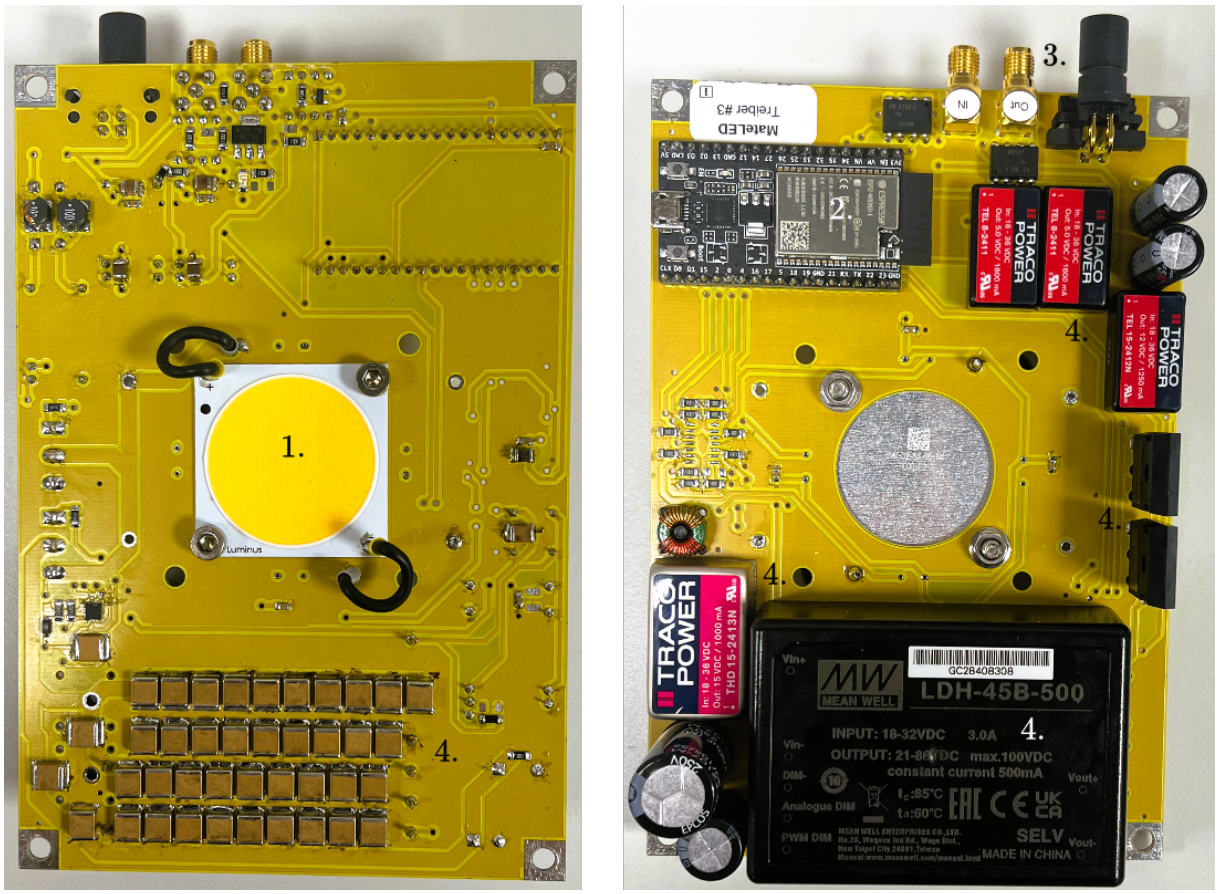


Abbildung 5.6: Das zusammengesetzte System, bestehend aus der LED-basierten Lichtquelle (1.), dem ESP32-Mikrocontroller (2.), Stromanschluss (3.) und den elektrischen Komponenten (4.)

Für die Inbetriebnahme des zusammengesetzten Systems gilt es zunächst dieses mit Strom über den Stromanschluss zu versorgen. Danach gibt es zwei Möglichkeiten der Ansteuerung des Systems die sich aus den Wegen der Kommunikation mit dem ESP32-Mikrocontroller ergeben wie sie in Unterkapitel 5.2.1 beschrieben sind: Drahtgebunden und drahtlos. Der drahtgebundene Weg setzt eine Verbindung zwischen dem System und einem Endgerät voraus das eine entsprechende Umgebung hat um Anweisungen an den Mikrocontroller zu übertragen. Diese Verbindung erfolgt beispielsweise über ein USB-Kabel das mit beiden Seiten verknüpft ist. Die Verbindung im Fall des drahtlosen Weges entsteht durch das Kommunikationsprotokoll wie es auf der Abbildung 5.4 zu sehen ist.

Um Messungen mit dem ESP32-Mikrocontroller sowie mit der LED-basierten Lichtquelle durchzuführen ist das Konzept eines experimentellen Aufbaus entstanden. Dieses Konzept besteht aus dem zusammengesetzten System das auf einer Arbeitsoberfläche fest montiert ist einem Lichtimpulsmessgerät das unmittelbar über dem System angebracht und eingestellt ist einem Oszilloskop mit dem die Messungen der elektronischen Signale des Mikrocontrollers sowie

der optischen Signale der LED-basierten Lichtquelle aufgezeichnet und visuell wiedergegeben werden sowie einem Endgerät um den Mikrocontroller anzusteuern.

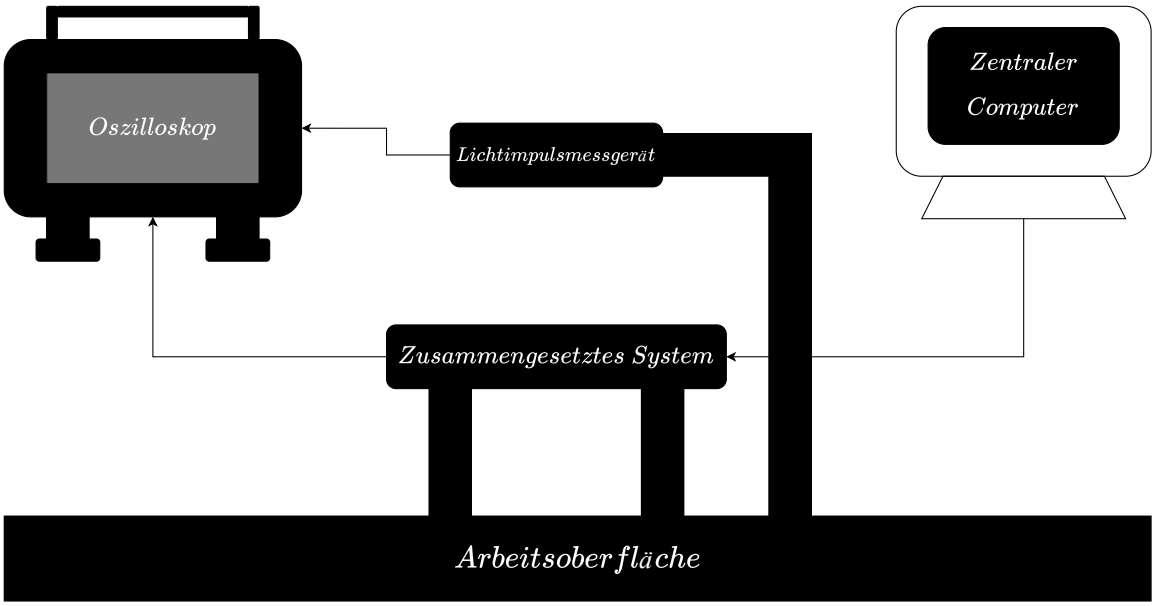


Abbildung 5.7: Experimenteller Aufbau - Konzept

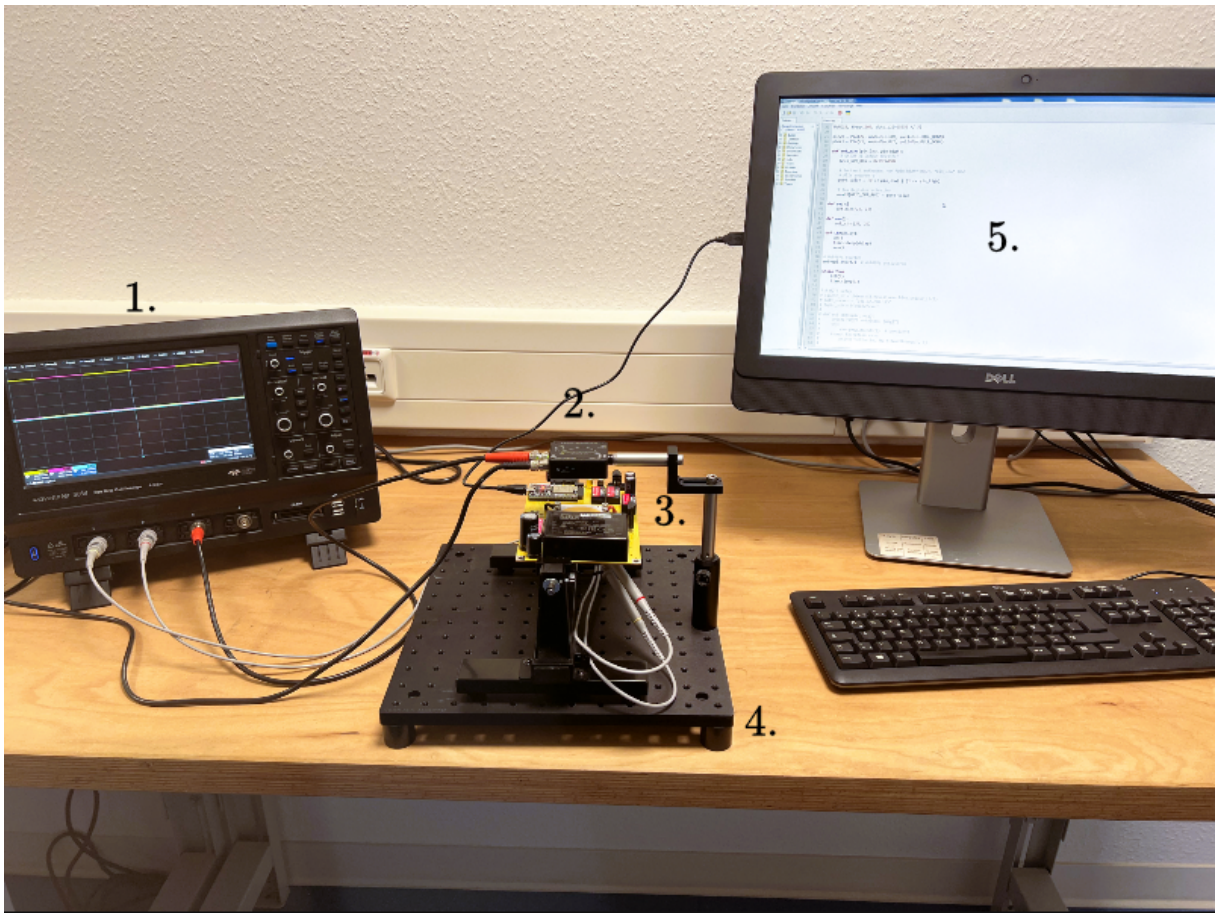


Abbildung 5.8: Experimenteller Aufbau - Tatsächliche Umsetzung (1. Oszilloskop, 2. Lichtimpulsmessgerät, 3. Zusammengesetztes System, 4. Arbeitsoberfläche, 5. Zentraler Computer)

5.3 Vergleich - Reale Umstände oder nur Simulation

Um das reale Potenzial des zusammengesetzten Systems zu untersuchen ist es wichtig das System in realen Anwendungsfällen sowie in authentischen Qualitätssicherungsprozessen einzusetzen. Im Laufe des Bearbeitungs- und Entwicklungsprozesses wurde das System für den experimentellen Aufbau der in Unterkapitel 5.2.2 näher beleuchtet wurde eingesetzt als auch für die anschließenden Messungen mit dem Oszilloskop. Somit wurde die Funktionsweise sowie der ordnungsgemäße Ablauf des Programms und das Ineinandergreifen der vorgestellten Schnittstellen wie WebREPL und MQTT in einer simulierten Umgebung verifiziert. Die Verwendung in einer industriellen Einsatzumgebung und damit in authentischen Qualitätssicherungsprozessen steht zum Zeitpunkt der Fertigstellung dieser Arbeit noch aus.

5.4 Herausforderungen/Schwierigkeiten

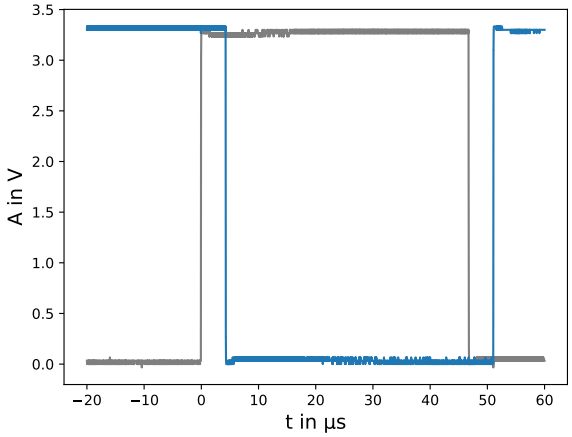
Für das Erfüllen der Anforderungen die in Unterkapitel 3.1 definiert wurden gab es während des Bearbeitungs- und Entwicklungsprozesses Schwierigkeiten und Komplikationen die es zu bewältigen galt. Unter anderem kam es zu einem Ausfall elektrischer Komponenten durch eine Stromüberlastung des zusammengesetzten Systems weil das System sowohl durch den im System verbauten Stromanschluss als auch durch den ESP32-Mikrocontroller selbst gleichzeitig mit Strom versorgt wurde. Zudem ist als die Messungen der elektrischen Signale des ESP32-Mikrocontrollers durchgeführt wurden festgestellt worden dass eine Verzögerung bei der Ansteuerung der beiden Pins entsteht die schließlich für einen Kurzschluss sowie eine fehlerhafte Ausführung des Programms sorgt.

5.4.1 Stromüberlastung

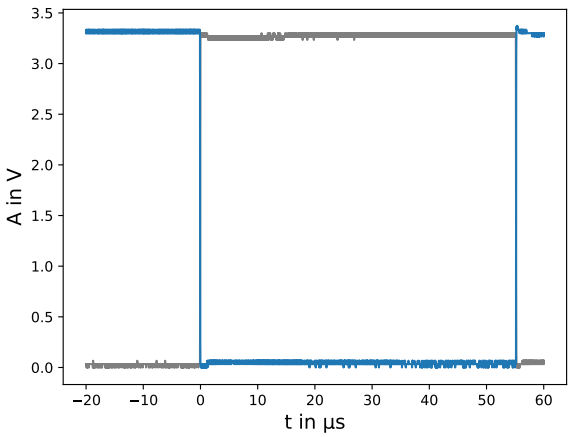
Um die ersten Messungen durchzuführen wurde wie es in Unterkapitel 5.2.2 dargelegt und visualisiert ist der experimentelle Aufbau verwendet und in Betrieb genommen. Hierzu wurde das zusammengesetzte System einerseits über den im System selbst verbauten Stromanschluss mit Strom versorgt. Dabei bekam der ESP32-Mikrocontroller die zum Arbeiten notwendige Energie zugeführt. Zusätzlich wurde der Mikrocontroller selbst über ein USB-Kabel an das ansteuernde Endgerät angeschlossen wodurch ebenso eine Stromversorgung für den Mikrocontroller entstand. Das führte dazu dass eine Stromüberlastung im Mikrocontroller stattfand elektrische Komponenten des Mikrocontrollers beschädigt wurden und der Mikrocontroller als Folge dessen ausgetauscht wurde. So wurde entschieden dass im tatsächlichen Betrieb wie auch bei den Messungen ausschließlich die drahtlose Kommunikation für die Übertragung von neuen Anweisungen verwendet wird um diese Komplikation zu beheben und weitere energetische Überlastungen zu vermeiden.

5.4.2 Divergenz elektrischer Signale

Um die LED-basierte Lichtquelle über den ESP32-Mikrocontroller anzusteuern wurden gemäß des Schaltplans und wie es zudem auf der Abbildung 5.1 zu sehen ist die Pins 22 und 23 über das Programm wie es in Listing 1 beschrieben ist entsprechend aktiviert und deaktiviert. Während des Bearbeitungs- und Entwicklungsprozesses wurden gleichermaßen Messungen des elektrischen Signals beider Pins durchgeführt wodurch festgestellt wurde dass sich der Wert der Pins zeitlich versetzt änderte.



(a) Zusammenspiel, beider elektrischen Signale der angesteuerten Pins, unter Verwendung der Pin-basierten Ansteuerung



(b) Zusammenspiel, beider elektrischen Signale der angesteuerten Pins, unter Verwendung der Port-basierten Ansteuerung

Abbildung 5.9: Unterschied zwischen Pin- und Port-basierter Programmierung, durch die zeitliche Verzögerung zwischen dem Aktivieren des ersten Signals und Deaktivieren des zweiten Signals sowie zwischen dem Deaktivieren des ersten Signals und Aktivieren des zweiten Signals

Somit ergab sich wie es auf der Abbildung 5.9a zu sehen ist ein zeitlicher Abstand zwischen dem Aktivieren des einen Pins und dem Deaktivieren des anderen Pins. Dieser zeitliche Abstand führte bei kleineren Zeitintervallen des Lichtimpulses dazu dass sowohl die ordnungsgemäße Funktion des zusammengesetzten Systems beeinträchtigt wurde als auch eine Beschädigung der elektrischen Komponenten auftrat weil die zeitliche Verzögerung zwischen dem Umschalten beider Pins für einen Stromfluss sorgte der die elektrischen Komponenten belastete. Um diesen Umstand zu lösen wurde statt der bisherigen Pin-basierten Programmierung eine Port-basierte Programmierung verwendet bei welcher der Wert der beiden Pins je nach Anwendungsfall vorher definiert und abschließend in einem Aufruf gesetzt wurde. Somit ergab sich ein gleichzeitiges Umschalten der beiden Pins und die zeitliche Verzögerung wurde so minimal dass der elektrische Fluss maßgeblich durch das Programm gesteuert wurde.

Unterschied zwischen Pin-basierter und Port-basierter Programmierung

Der ESP32-Mikrocontroller verfügt über insgesamt 38 Pins. Diese Pins haben verschiedene Funktionen. So gibt es zum Beispiel Pins die ausschließlich als Eingang dienen und es gibt Pins die sowohl als Eingang als auch als Ausgang dienen. Das bedeutet dass die Pins die ausschließlich als Eingangspins definiert sind nur die Möglichkeit haben Signale angeschlossener Peripherie zu registrieren und diese Signale an den Mikrocontroller weiterzuleiten sodass dieser damit arbeitet. Die Pins die sowohl als Eingangs- und auch als Ausgangspins definiert sind haben die Möglichkeit wie die Eingangspins Signale angeschlossener Peripherie zu registrieren und an den

Mikrocontroller weiterzuleiten. Zudem bieten diese Pins die Option Signale auszusenden über elektrische Kommunikation. Exemplarisch ist hierbei das Anschalten einer LED zu erwähnen indem ein solcher Eingangs- und Ausgangspin aktiviert wird. Auf der Abbildung 8.8 sind alle 38 Pins des ESP32-Mikrocontrollers dargestellt. Farblich kodiert sind die einzelnen Funktionsgruppen dieser Pins. Diese Funktionsgruppen dienen als Einteilung in beispielsweise Eingangspins, Ein- und Ausgangspins, Massepins, Pins für die serielle Kommunikation sowie Pins die haptische Signale registrieren. Zusätzlich zu den Funktionsgruppen gibt es die Möglichkeit mehrere Pins über einen Port anzusprechen. Ein Port ist in dem Fall eine virtuelle Repräsentation mehrerer Pins. Somit ist ein Konfigurationsaufruf eines Ports ausreichend um zeitgleich mehrere Pin-Werte zu ändern. Daraus ergeben sich zwei Möglichkeiten der Programmierung: Einerseits gibt es die Möglichkeit einen einzelnen Pin anzusteuern und zu programmieren. Andererseits gibt es die Möglichkeit einen Port anzusteuern und zu programmieren womit mehrere Pins angesteuert und programmiert werden.
Bei der zuvor beschriebenen Divergenz elektrischer Signale wurden zunächst die einzelnen Pins angesteuert. Dies führte dazu dass zuerst ein Pin deaktiviert und danach ein anderer Pin aktiviert wurde. Das Resultat davon war die zeitliche Verzögerung wie sie auf Abbildung 5.9a grafisch aufbereitet ist. Um diese zeitliche Verzögerung so geringfügig wie möglich zu gestalten wurde sich daraufhin entschieden den entsprechenden Port der unter anderem die beiden Pins adressiert zu programmieren. Also wurde die vorherige Pin-basierte Programmierung durch eine Port-basierte Programmierung ersetzt wodurch die zeitliche Verzögerung gesenkt wurde.

5.4.3 Zeitliche Verzögerung des Abschaltens der LED-basierten Lichtquelle

Um die LED-basierte Lichtquelle in den entsprechenden Intervallen zu pulsieren gibt es die zwei Hilfsmethoden „an“ und „aus“ die jeweils für das Aktivieren beziehungsweise Deaktivieren der Lichtquelle sorgen. Diese beiden Hilfsmethoden sind im Listing 1 auf den Zeilen 39–43 gelistet. Zudem gibt es die Methode „LED“ mit einem Parameter „delay“ auf den Zeilen 45–49 die entsprechend dieses Parameters die Lichtquelle aktiviert für diesen zeitlichen Abstand die weitere Abarbeitung des Programms pausiert und die Lichtquelle schließlich deaktiviert. Bei einer Pulsdauer von zehn Mikrosekunden weist der gesamte Vorgang des Aktivierens sowie des Deaktivierens somit sechzig Mikrosekunden auf bis die Periode abgeschlossen ist. Dies umfasst das Aktivieren das Pausieren und das Deaktivieren der Lichtquelle. Bei zehn hintereinander durchgeführten Perioden des Pulsierens wurde festgestellt dass die Zeiten vom Aktivieren bis zum abschließenden Deaktivieren variieren. So weist der gesamte Vorgang bei fünf Perioden eine Dauer von sechzig Mikrosekunden bei zwei Perioden eine Dauer von siebzig Mikrose-

kunden bei zwei Perioden eine Dauer von zweihundertdreißig Mikrosekunden und bei einer Periode eine Dauer von fünfhundertsechzig Mikrosekunden auf. Hierfür gibt es zwei mögliche Erklärungen: Einerseits besteht die Möglichkeit dass der ESP32-Mikrocontroller beim Abarbeiten der MicroPython-Befehle durch die sogenannten internen „Interrupts“ also interne Unterbrechungen angehalten wird bis die aktuelle interne Unterbrechung abgeschlossen ist. Diese internen Unterbrechungen sind essenziell für die Funktionsweise des Mikrocontrollers weil damit beispielsweise die drahtlose Kommunikation realisiert ist bei der in regelmäßigen Abständen geprüft wird ob die Verbindung weiterhin besteht und welche Informationen übertragen werden. Andererseits besteht die Möglichkeit dass eine Verzögerung durch das Interpretieren der MicroPython-Befehle entsteht die der Mikrocontroller abarbeitet. MicroPython wie es in Unterkapitel 4.2 beschrieben ist stellt eine Hardware-Abstraktionsschicht dar die den geschriebenen MicroPython-Code in die entsprechenden Anweisungen übersetzt die der Mikrocontroller ausführt. Bei dieser Abstraktion und der damit einhergehenden Umwandlung des eingegebenen MicroPython-Codes ist es realistisch dass ein zeitlicher Versatz beim Einlesen und Ausführen der Befehle entsteht.

Für die Untersuchung der ersten Möglichkeit wurde mittels der „machine“-Bibliothek die Funktionalität der internen Unterbrechungen für den Zeitraum der Abarbeitung des Programmabschnitts deaktiviert. Das Verhalten der einzelnen Periodendauern blieb in der nachgelagerten Betrachtung gleich. Für die Untersuchung der zweiten Möglichkeit ist MicroPython als Abstraktionsschicht selbst zu ersetzen und durch die Anforderung begründet dass MicroPython weiterhin zu verwenden ist wurde davon abgesehen MicroPython auf dem Mikrocontroller zu ersetzen.

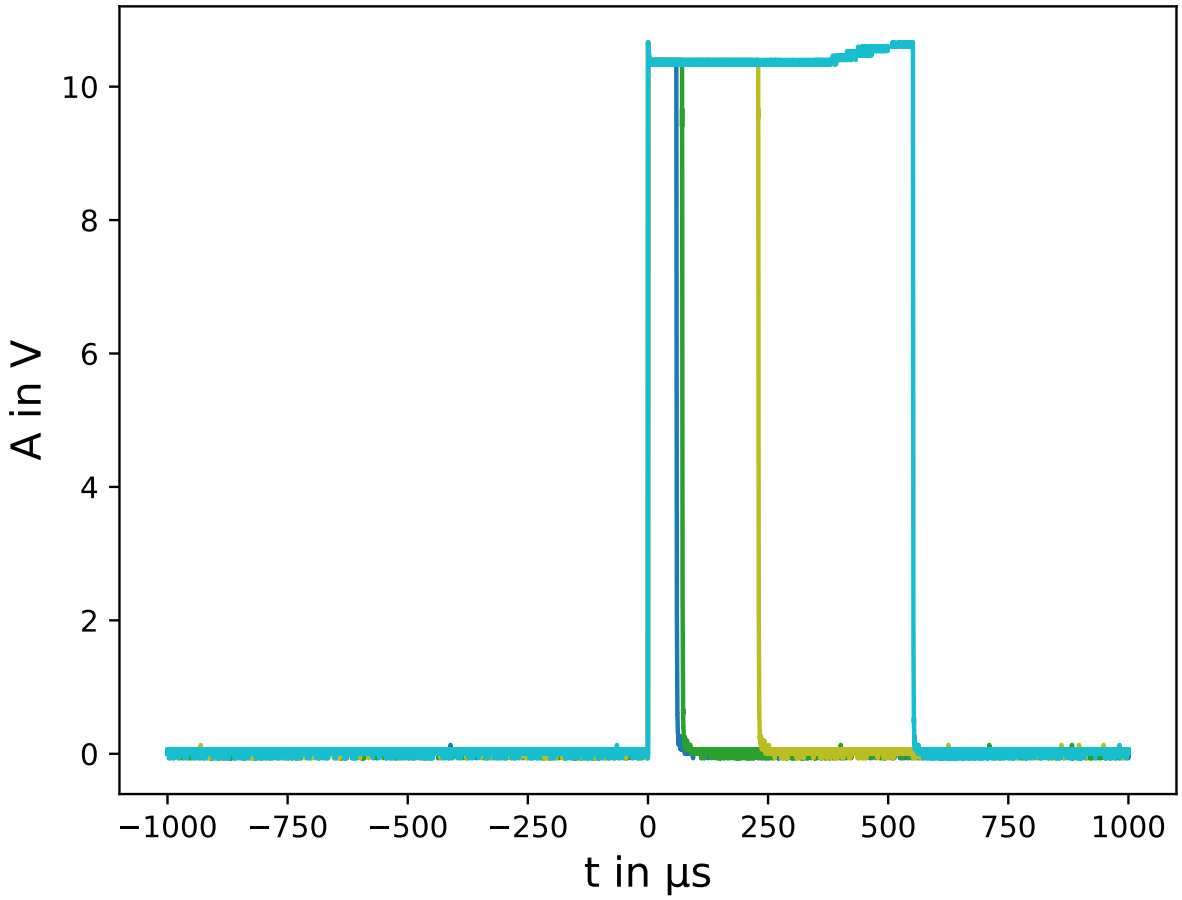


Abbildung 5.10: Lichtimpulsperioden und die ausgehende Energie, vom Aktivieren der LED-basierten Lichtquelle bis zum vollständigen Erlöschen der Lichtquelle, bei einer eingestellten Lichtimpulsdauer von fünf Mikrosekunden - 60 Mikrosekunden (Dunkelblau), 70 Mikrosekunden (Grün), 230 Mikrosekunden (Limette) und 560 Mikrosekunden (Türkis)

5.5 Fehleranalyse

Um die Anforderungen so wie sie in Unterkapitel 3.1 definiert wurden umzusetzen wurden an den entsprechenden Schnittstellen Anpassungen vorgenommen die zu einer Veränderung des zu diesem Zeitpunkt bisherigen Verhaltens des Systems führten. Vor allem im Kontext der drahtlosen Kommunikation gab es besagte Anpassungen um das Kommunikationssystem mit den Anforderungen zu realisieren. Zum Beispiel wurde durch die Verwendung MQTTs der Kommunikationsport blockiert über den zuvor die drahtgebundene Kommunikation sowie die drahtlose Kommunikation über andere Medien stattfand weswegen daraus resultierte dass in diesem Fall ausschließlich die Kommunikation über MQTT möglich ist. Um eine andere Möglichkeit der Kommunikation mit dem Mikrocontroller wahrzunehmen galt es dann die

bestehende MQTT-Verbindung aufzulösen und den Mikrocontroller über einen anderen Weg anzusteuern entweder drahtgebunden oder direkt über WebREPL. Im Falle WebREPLs gibt es eine eigenständige Bibliothek mit der unter Verwendung von WebREPL selbst und eines Websockets eine drahtlose Verbindung zu einem ESP32-Mikrocontroller aufgebaut wird über die mittels der MicroPython-Programmiersprache Befehle an den Mikrocontroller gesendet werden die dieser direkt ausführt. Um den Abschluss eines eingegebenen Befehls zu definieren wurde sich dafür entschieden die Konsoleneingabe selbst als Vergleichswert zu benutzen um zu kontrollieren ob weitere Zeichen eingegeben wurden. Dabei wurde eine leere Befehlseingabezeile als Bedingung für den Abschluss eines eingegebenen Befehls verwendet.

5.5.1 Blockade eines Ports durch die Nutzung MQTTs

Um eine drahtlose Kommunikation mittels MQTT zu realisieren, wurden wie in Unterkapitel 5.2 beschrieben, die erforderlichen Vorbereitungen im Programm vorgenommen. Die, für die Kommunikation mit dem MQTT-Broker, relevanten Abschnitte befinden sich im Listing 1 auf den Zeilen 53 - 79. Durch die Nutzung des MQTT-Clients der mittels einer while-Schleife dauerhaft überprüft ob eine neue Anweisung über das abonnierte Thema gesendet wurde wird dauerhaft der entsprechende Kommunikationsport verwendet über den die Kommunikation mit dem MQTT-Broker stattfindet. Somit ist der Kommunikationsport für eine drahtgebundene Kommunikation sowie eine drahtlose Kommunikation durch einen anderen Weg blockiert.

5.5.2 Blockade lösen durch Trennen von MQTT

Um die zuvor beschriebene Blockade des Kommunikationsports über den die drahtlose Kommunikation mit dem MQTT-Broker realisiert ist, zu lösen ist das laufende Programm zu beenden und der Kommunikationsport wieder freizugeben. Dazu ist eine entsprechende Entwicklungsumgebung aufzurufen mit der es möglich ist den eingesetzten ESP32-Mikrocontroller drahtgebunden anzusteuern. In dieser Entwicklungsumgebung ist das laufende Programm zu beenden entweder über eine integrierte Funktion der Entwicklungsumgebung oder über die Konsoleneingabe in der Entwicklungsumgebung. Dadurch wird der verwendete Kommunikationsport wieder freigegeben und der Mikrocontroller ist erneut für alle Möglichkeiten der Kommunikation verfügbar.

5.5.3 Vergleich der Konsolenausgabe bei Websocket

Im Laufe der Bearbeitungs- und Entwicklungszeit wurde eine eigenständige Python-Bibliothek implementiert, mit der es möglich ist unter Verwendung WebREPLs und eines Websockets eine

direkte Verbindung mit einem ESP32-Mikrocontroller herzustellen. Diese Verbindung sorgt dafür dass alle Anweisungen die mithilfe einer entsprechenden Eingabeaufforderung eingegeben werden an den verbundenen Mikrocontroller übertragen durch die MicroPython-Umgebung übersetzt und direkt durch den Mikrocontroller ausgeführt werden. Die Eingabe der Anweisungen die abschließend an den Mikrocontroller übertragen werden erfolgt hierbei über eine entsprechende Eingabeaufforderung. Um den Abschluss einer Anweisung programmatisch zu definieren wurde die Eingabeaufforderung verwendet. Dazu wurde der Anfang der Eingabeaufforderungszeile als Vergleichswert verwendet sodass diese definierte Zeichenfolge für das Ausführen der eingegebenen Anweisung sorgt. Im Konkreten ist die while-Schleife die in diesem Fall durchgehend ausgeführt wird dafür zuständig dass alle übertragenen Anweisungen gelesen und an den Mikrocontroller übertragen werden. Die Zeichenfolge „>>>“ stellt die Kennzeichnung des Abschlusses der Anweisung dar und die abgeschlossene Anweisung wird an den Mikrocontroller übertragen der diese direkt ausführt.

6 Zusammenfassung

Um die bestehenden Qualitätssicherungsprozesse in industriellen Umgebungen zu verbessern und die im Unterkapitel 3.1 definierten Anforderungen sowie Herausforderungen zu lösen wurde innerhalb dieser Bachelorarbeit ein Kommunikationssystem entwickelt das sowohl drahtgebunden als auch drahtlos eine Kommunikation mit dem Lichtemissionsmodul ermöglicht. Dieses Lichtemissionsmodul wird in der Qualitätssicherung dafür eingesetzt um die zu prüfenden Bauteile in der entsprechenden Wiederholrate zu beleuchten sodass visuelle Aufzeichnungen der Bauteile möglich werden. Zusätzlich ist im Laufe der Bearbeitungs- und Entwicklungszeit dieser Bachelorarbeit das zusammengesetzte System entstanden wie es im Unterkapitel 5.2.2 beschrieben und auf der Abbildung 5.6 zu sehen ist das im weiteren Verlauf für die nachfolgende Entwicklung und Erweiterung der eingesetzten Qualitätssicherungseinheit verwendet wird.
Um das Lichtemissionsmodul drahtgebunden anzusteuern gilt es den ESP32-Mikrocontroller beispielsweise mit einem USB-Kabel an ein Endgerät anzuschließen mit dem es möglich ist über eine entsprechende Entwicklungsumgebung auf den Mikrocontroller zuzugreifen und ihm Anweisungen über diese Entwicklungsumgebung zu übertragen. Die drahtlose Ansteuerung ist einerseits mittels WebREPL sowie einem Websocket und andererseits über MQTT möglich. Die Kommunikation unter Nutzung von WebREPL und einem Websocket funktioniert durch das Aktivieren des WebREPL-Kommunikationsprotokolls auf dem Mikrocontroller selbst die Einrichtung eines WebREPL-Passwortes auf dem Mikrocontroller und durch das Verwenden des Programms das in Listing 2 aufgeführt ist. Damit wird eine drahtlose Verbindung unter Angabe der WebREPL-Adresse des Mikrocontrollers sowie des WebREPL-Passworts des Mikrocontrollers zum entsprechenden Mikrocontroller hergestellt. Für die Kommunikation in Kombination mit MQTT gilt es auf dem eingesetzten Mikrocontroller so wie es am Anfang des Unterkapitels 5.2 beleuchtet worden ist die Boot-Datei hochzuladen in der die grundlegenden Konfigurationen und Einstellungen der MQTT-Verbindung vorgenommen sind. Abschließend ist eine Anwendung aufzurufen mit der es möglich ist eine Verbindung mit einem MQTT-Broker herzustellen und über diese Verbindung unter Angabe des Namens dieses Themas eine Nachricht an ein Thema zu senden. Diese Nachricht wird durch die auf dem Mikrocontroller eingerichtete MQTT-Schnittstelle ausgelesen als MicroPython-Anweisung interpretiert und direkt ausgeführt. Somit ist es möglich eine Kommunikation sowohl drahtgebunden als auch drahtlos mit dem Lichtemissionsmodul herzustellen und dieses entsprechend anzusteuern sowie Anweisungen zu übertragen.

6.1 Veränderung bei bisherigen Verfahren

Die grundlegende Funktion die im Rahmen dieser Bachelorarbeit betrachtet und bearbeitet wurde um eine Anpassung des Verfahrens zu realisieren war die Implementierung eines Kommunikationssystems. Dieses Kommunikationssystem stellt eine Verknüpfung der einzelnen Komponenten her die bei der Beleuchtung eines zu prüfenden Bauteils das in einem Qualitätssicherungsverfahren untersucht wird eingesetzt werden. Dadurch lässt sich das zusammengesetzte System sowohl drahtgebunden als auch drahtlos direkt ansteuern und es besteht die Möglichkeit jederzeit Änderungen an den Einstellungen des Systems vorzunehmen.

6.2 Vergleich mit anderen Cyber Vision-Verfahren

Um einen direkten Vergleich der aktuellen Ergebnisse mit bisherigen Cyber-Vision-Verfahren zu erzielen werden exemplarisch im Folgenden die eingesetzten Verfahren behandelt die einerseits aus [HBK24] und [Cav24] stammen.
In [HBK24] wurde der Aufbau mittels 25 Lichtstreifen und 30 Monochrom-Kameras realisiert um die zu prüfenden Bauteile in dem verwendeten Qualitätssicherungsverfahren zu untersuchen. Dabei sind die Lichtstreifen durchgehend aktiviert und die Monochrom-Kameras zeichnen fortlaufend den Qualitätssicherungsprozess auf dessen Ergebnisse anschließend mittels künstlicher Intelligenz ausgewertet werden. Dementsprechend wird alles einmal eingerichtet und aktiviert um die Produktion sowie die Qualitätssicherung zu gewährleisten. Ein entsprechendes Kommunikationssystem zur Ansteuerung der einzelnen Komponenten ist hierbei außenvor.

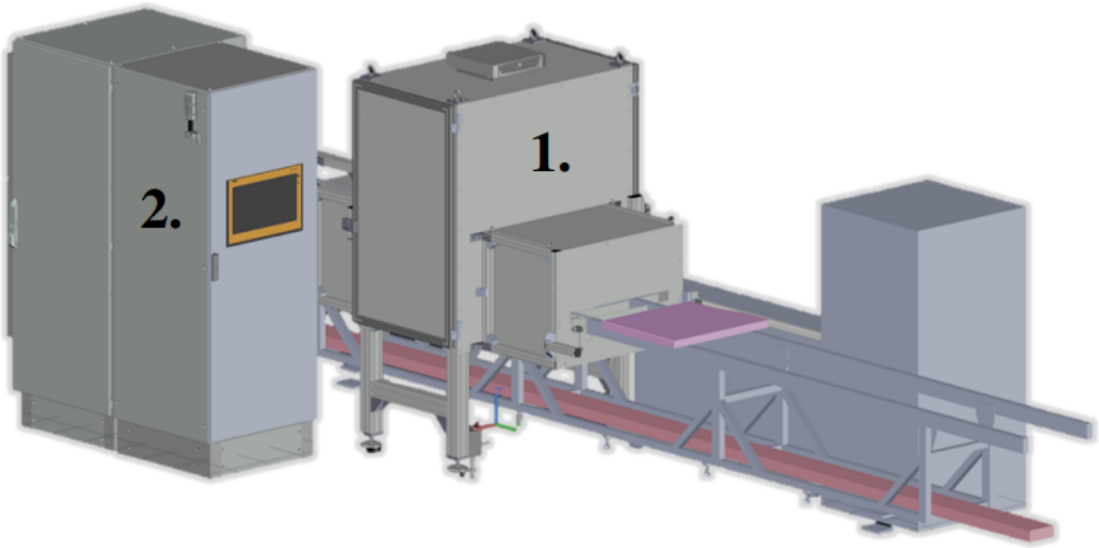


Abbildung 6.1: Abbildung des, in [HBK24] verwendeten, Aufbaus (1. Auf dem Fließband montierte Inspektionsumgebung, 2. Berechnungs- und Untersuchungseinheit)

In [Cav24] wurde der Aufbau mittels einer Lichtfläche die mit weißen LEDs ausgestattet war und einer Kamera realisiert um die zu prüfenden Bauteile in dem verwendeten Qualitätssicherungsverfahren zu untersuchen. Auch hierbei ist die Lichtfläche durchgehend aktiviert und die Kamera zeichnet konstant den Qualitätssicherungsprozess auf dessen Ergebnisse anschließend mittels künstlicher Intelligenz in diesem Fall „Deep Learning“ ausgewertet werden. Auch hierbei wird alles einmal eingerichtet und aktiviert um die dauerhafte Produktion sowie die Qualitätssicherung zu gewährleisten. Darausfolgend bleibt hierbei ebenso die Verwendung eines ansteuernden Kommunikationssystems zur exakten Ansteuerung der Lichtquelle aus.

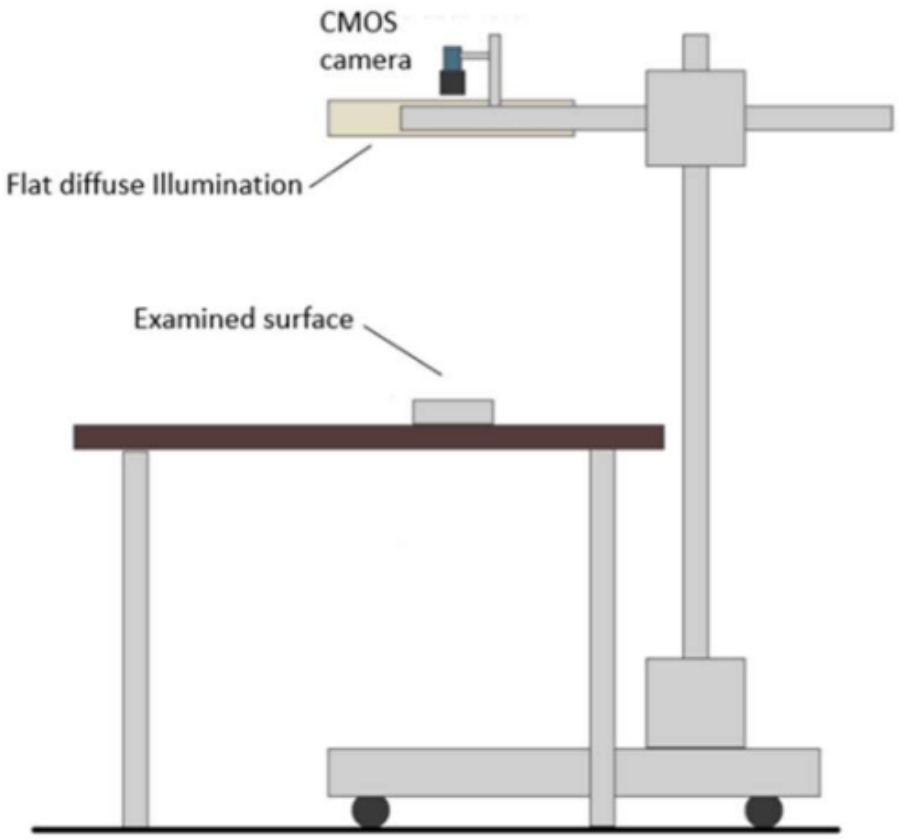


Abbildung 6.2: Abbildung des, in [Cav24] verwendeten, Aufbaus

6.3 Vorteile des entwickelten Kommunikationssystems

Durch den Vergleich im vorangegangenen Unterkapitel und durch eine eingehende Betrachtung des entwickelten Kommunikationssystems ergeben sich folgende Vorteile gegenüber vergleichbaren Qualitätssicherungsverfahren bei denen die Komponenten durchgehend aktiv sind:

- Vor allem beim Einsatz mehrerer Lichtquellen ist es möglich alle Lichtquellen synchron oder individuell anzusteuern indem die MQTT-Verbindung entsprechend angepasst wird.
- Basierend auf der Möglichkeit jederzeit Anweisungen an den ESP32-Mikrocontroller zu übertragen lässt sich somit die Pulsweitenmodulation und damit auch die Lichtintensität der verbundenen Lichtquelle einstellen.
- Im Vergleich zu dauerhaft aktiven Komponenten ist der Energieverbrauch durch das exakte Ansteuern der verbundenen Lichtquelle bei Qualitätssicherungsverfahren die das

entwickelte Kommunikationssystem einsetzen geringer.

- Im Vergleich zu dauerhaft aktiven Komponenten ist die Wärmeentwicklung durch das exakte Ansteuern der verbundenen Lichtquelle und den damit einhergehenden Pausenzeiten der verbundenen Lichtquelle bei Qualitätssicherungsverfahren die das entwickelte Kommunikationssystem einsetzen geringer.
- Die eingesetzte Energie die bei Qualitätssicherungsverfahren im Bereich der visuellen Kontrolle durch die Beleuchtung des zu prüfenden Bauteils verwendet wird lässt sich konzentrierter und fokussierter auf ein zeitlich kleineres Intervall anwenden sodass eine bessere Ausgangssituation für die visuelle Aufzeichnung entsteht.

6.4 Nachteile des entwickelten Kommunikationssystems

Durch den Vergleich im vorangegangenen Unterkapitel und durch eine eingehende Betrachtung des entwickelten Kommunikationssystems ergeben sich folgende Nachteile gegenüber vergleichbaren Qualitätssicherungsverfahren bei denen die Komponenten durchgehend aktiv sind:

- Um das entwickelte Kommunikationssystem zu verwenden gilt es im Vorfeld die entsprechende Entwicklung und Einstellung des Kommunikationssystems vorzunehmen.
- Um das entwickelte Kommunikationssystem zu verwenden ist eine entsprechende Infrastruktur erforderlich. (ein Endgerät ein Transportmedium und eine Anwendung zum Ansteuern des Kommunikationssystems)
- Um das entwickelte Kommunikationssystem zu verwenden ist es wichtig dass der Anwender das Wissen hat um die Verbindung zwischen der Infrastruktur und dem zusammengesetzten System aufzubauen und einzurichten sowie Anweisungen an das System zu übertragen.

7 Ausblick

Das grundlegende Kommunikationssystem zum Ansteuern der LED-basierten Lichtquellen die bei der Qualitätssicherung im Bereich der visuellen Kontrolle eingesetzt werden ist abgeschlossen. Nachfolgend gilt es das bestehende Kommunikationssystem weiterzuentwickeln in tatsächliche Qualitätssicherungsverfahren zu integrieren und weitere Komponenten die bei Qualitätssicherungsverfahren eingesetzt werden durch das Kommunikationssystem anzusteuern.

7.1 Weiterentwicklung des Systems

Um das Potenzial des Kommunikationssystems weiter auszuschöpfen gibt es verschiedene Möglichkeiten der Weiterentwicklung:

- Skalierung: Durch den Einsatz mehrerer zusammengesetzter Systeme bestehend aus der LED-basierten Lichtquelle den elektrischen Komponenten dem ESP32-Mikrocontroller und dem Stromanschluss so wie es in Unterkapitel 5.2.2 erläutert und auf der Abbildung 5.6 dargestellt ist ergibt sich ein größerer Bereich der beleuchtet wird und somit für Qualitätssicherungsverfahren verfügbar ist.
- Leistung der Komponenten: Durch den Einsatz eines Mikrocontrollers mit einer höheren Prozessortaktrate einer Lichtquelle mit einer höheren Lichtintensität oder elektrischer Komponenten die leistungsfähiger sind wird das Potenzial eines Qualitätssicherungsverfahrens weiter ausgeschöpft bei dem das Kommunikationssystem eingesetzt wird.
- Diversifizieren der angesteuerten Komponenten: Durch die Client-Broker-Architektur und das Abonnement-Modell die mit der Verwendung MQTTs einhergehen ist ein hoher Grad an Individualisierung und Kategorisierung der einzelnen Komponenten realisierbar; für jedes abonnierte Thema ist es hierbei möglich eine andere Anweisung an die angesteuerten Mikrocontroller zu übertragen und somit die einzelnen zusammengesetzten Systeme je nach Anwendungsfall zu kategorisieren.

7.2 Veränderung von Verfahren aufgrund der gewonnenen Erkenntnisse

Um Qualitätssicherungsverfahren im Bereich der visuellen Kontrolle mit diesem Kommunikationssystem auszustatten und die in Kapitel 6 beleuchteten Vorteile gegenüber Qualitätssicherungsverfahren ohne Kommunikationssystem zu erhalten ist es grundlegend die entsprechende Infrastruktur einzurichten. Darausfolgend lassen sich Verfahren wie sie beispielsweise in

[HBK24] und in [Cav24] zu finden sind präziser ansteuern. Somit ist auch da eine Anpassung und gegebenenfalls eine Verbesserung der eingesetzten Ausgangslage möglich.

7.3 Veränderung von anderen Systemen

Um bestehende Systeme im Bereich der visuellen Qualitätssicherung mit der in dieser Bachelorarbeit vorgestellten Herangehensweise zu verknüpfen und damit eine Veränderung sowie eine potenzielle Verbesserung zu erreichen gibt es entsprechende Kriterien die es zu erfüllen gilt: Eine Kommunikationseinheit die wie in der vorgestellten Herangehensweise der ESP32-Mikrocontroller als Schnittstelle zwischen der Lichtquelle und dem ansteuernden Endgerät fungiert sodass Anweisungen die ankommen direkt ausgeführt werden. Ein Transportmedium das eingegebene Anweisungen an die Kommunikationseinheit überträgt. Dabei gibt es die Möglichkeit der drahtgebundenen sowie der drahtlosen Übertragung wobei die drahtlose Übertragung gewisse Bruchstellen und Herausforderungen löst wie sie in Unterkapitel 3.1 beschrieben worden sind die bei der Verwendung der drahtgebundenen Übertragung aufkommen. Und im Fall der drahtlosen Kommunikation ist eine zentrale Schnittstelle zu verwenden welche die eingegebenen Anweisungen an die verbundenen Kommunikationseinheiten überträgt wie im Rahmen dieser Bachelorarbeit das Kommunikationsprotokoll MQTT.

7.4 Anwendung auf andere Bereiche

Wie am Anfang dieses Kapitels beschrieben gilt es um das bestehende Kommunikationssystem weiterzuentwickeln und dessen Potenzial sowie Möglichkeiten zu erforschen weitere Komponenten die bei Qualitätssicherungsverfahren eingesetzt werden durch das Kommunikationssystem anzusteuern. Im Bereich der visuellen Kontrolle ist es zum Beispiel denkbar das aufzeichnende Element wie eine Kamera durch das Kommunikationssystem anzusteuern. In anderen Bereichen der Qualitätssicherung gibt es die Möglichkeit andere eingesetzte Instrumente wie akustische Quellen oder mechanische Gelenke durch das Kommunikationssystem anzusteuern. Und auch außerhalb der Qualitätssicherung stellt der Einsatz des Kommunikationssystems in verschiedenen Anwendungsgebieten durchaus eine Veränderung und hypothetische Verbesserung dar: in der Regelung von Steuereinheiten beim Bedienen von Türen und Fenstern sowie beim Auslesen von Sensorwerten.

Literatur

[Ard25] Arduino Società a responsabilità limitata. *Arduino Portenta H7-Datenblatt*. Englisch. Dez. 2025. URL: https://docs.arduino.cc/resources/datasheets/ABX00042-ABX00045-ABX00046-datasheet.pdf.

[Bun] Bundesrepublik Deutschland. *Rolle-Zu-Rolle-Verfahren*. Deutsch. Begriffsdefinition. Deutschland. URL: https://www.energieforschung.de/de/glossar/Rolle-zu-Rolle-Verfahren.

[Bun16] Bundesrepublik Deutschland. *Elektromagnetische Verträglichkeit von Betriebsmitteln (Elektromagnetische-Verträglichkeit-Gesetz-EMVG)*. Deutsch. Dez. 2016. URL: https://www.gesetze-im-internet.de/emvg_2016/EMVG.pdf.

[Cav24] Georgio Cavaliere. „Comparative use of systems to detect surface defects in die-cast components using advanced vision systems applying artificial intelligence“. Englisch. Diss. Bozen-Bolzano: Free University of Bozen-Bolzano, Juli 2024.

[DIN03a] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-0*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007032.

[DIN03b] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-1*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007033.

[DIN03c] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-11*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007042.

[DIN03d] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-12*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007043.

[DIN03e] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-13*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007044.

[DIN03f] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-14*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007045.

[DIN03g] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-15*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007046.

[DIN03h] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-17*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007047.

[DIN03i] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-2*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007034.

[DIN03j] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-3*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007035.

[DIN03k] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-4*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007036.

[DIN03l] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-5*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007037.

[DIN03m] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-6*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007038.

[DIN03n] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-7*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007039.

[DIN03o] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-8*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007040.

[DIN03p] DIN-Normenausschuss Technische Grundlagen (NATG). *DIN 8589-9*. Deutsch. N Norm ST Norm. Deutschland, Sep. 2003. URL: https://www.nautos.de/O7K/search/item-detail/DE30007041.

[DIN16] DIN-Normenausschuss Materialprüfung (NMP). *DIN EN 13018*. N Norm ST Norm. Deutschland, Juni 2016. URL: https://www.nautos.de/O7K/search/item-detail/DE30066107.

[DIN19] DIN-Normenausschuss Werkzeugmaschinen (NWM). *DIN EN ISO 16092-1*. Deutsch. N Norm ST Norm VW Gilt in Verbindung mit anderen Dokumenten. Deutschland, Aug. 2019. URL: https://www.nautos.de/O7K/search/item-detail/DE30074251.

[DKE15a] DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE. *DIN EN 60812*. DC Entwurf N-E Norm-Entwurf WH In Überarbeitung (vorgesehener Ersatz). Deutschland, Aug. 2015. URL: https://www.nautos.de/O7K/search/item-detail/DE30063039.

[DKE15b] DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE. *DIN EN 62740*. N Norm ST Norm. Deutschland, Nov. 2015. URL: https://www.nautos.de/O7K/search/item-detail/DE30063468.

[DS21] DIN-Normenausschuss Qualitätsmanagement und Statistik und Zertifizierungsgrundlagen (NQSZ). *DIN 55350*. N Norm ST Norm. Deutschland, Okt. 2021. URL: https://www.nautos.de/O7K/search/item-detail/DE30090746.

[DS25] DIN-Normenausschuss Qualitätsmanagement und Statistik und Zertifizierungsgrundlagen (NQSZ). *DIN EN ISO 9001*. DC Entwurf N-E Norm-Entwurf WH In Überarbeitung (vorgesehener Ersatz). Deutschland, Sep. 2025. URL: https://www.nautos.de/O7K/search/item-detail/DE30106815.

[Esp22] Espressif Systems (Shanghai) Company, Limited. *ESP32-DevKitC*. Hardwaredokumentation. Jan. 2022. URL: https://docs.espressif.com/projects/arduino-esp32/en/latest/_images/esp32_devkitC_pinlayout.png.

[Esp25a] Espressif Systems (Shanghai) Company, Limited. *ESP32-Datenblatt*. Englisch. Dez. 2025. URL: https://documentation.espressif.com/esp32_datasheet_en.pdf.

[Esp25b] Espressif Systems (Shanghai) Company, Limited. *ESP8266-Datenblatt*. Englisch. Dez. 2025. URL: https://documentation.espressif.com/0a-esp8266ex_datasheet_en.pdf.

[Geh+16] Winfried Gehrke u. a. *Digitaltechnik*. Deutsch. 7. Auflage. Springer-Lehrbuch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016. ISBN: 978-3-662-49730-2 978-3-662-49731-9. DOI: 10.1007/978-3-662-49731-9.

[Geo05] Damien George. *MicroPython*. Programmers: _:n32. Dez. 205. URL: https://github.com/micropython/micropython (besucht am 03. 12. 2025).

[Geo25a] Damien George. *MicroPython*. Englisch. Dokumentation. Nov. 2025. URL: https://micropython.org/ (besucht am 14. 11. 2025).

[Geo25b] Damien George. *MicroPython - MIT-Lizenz*. Lizenz. Jan. 2025. URL: https://github.com/micropython/micropython/blob/master/LICENSE (besucht am 03. 12. 2025).

[HBK24] Sebastian Hunger, Michael Breiter und Claudia Klein. „Data acqusition challenges in AI-driven surface inspection: A proven solution proposal on coated sheet metal parts“. Englisch. In: (März 2024). Publisher: UB HSU. DOI: 10.24405/15310.

[lab17] labplus-cn. *umqtt.simple - MQTT Client function*. Englisch. Softwaredokumentation. Juni 2017. URL: https://github.com/micropython/micropython-lib/blob/mqtt/umqtt.simple/umqtt/simple.py (besucht am 02. 12. 2025).

[LSL25] João Victor Rojas Luiz, Fernando Bernardi De Souza und Octaviano Rojas Luiz. „Theory of Constraints and Industry 4.0: mutual contributions and research perspectives“. Englisch. In: *Production* 35 (2025), S. 15. ISSN: 1980-5411, 0103-6513. DOI: 10.1590/0103-6513.20250032.

[Mic25a] Microchip Incorporated. *PIC32MX-Familie-Datenblatt*. Englisch. Dez. 2025. URL: https://ww1.microchip.com/downloads/en/devicedoc/pic32mx_datasheet_v1_61143a.pdf.

[Mic25b] Microchip Incorporated. *PIC32MX5XX/6XX/7XX-Familien-Datenblatt*. Englisch. Dez. 2025. URL: https://ww1.microchip.com/downloads/en/devicedoc/61156g.pdf.

[Nor25a] Nordic Semiconductor Aksjeselskap. *nRF52805-Datenblatt*. Englisch. Dez. 2025. URL: https://docs.nordicsemi.com/bundle/nRF52805_PS_v1.4/resource/nRF52805_PS_v1.4.pdf.

[Nor25b] Nordic Semiconductor Aksjeselskap. *nRF52810-Datenblatt*. Englisch. Dez. 2025. URL: https://docs.nordicsemi.com/bundle/nRF52810_PS_v1.5/resource/nRF52810_PS_v1.5.pdf.

[Nor25c] Nordic Semiconductor Aksjeselskap. *nRF52811-Datenblatt*. Englisch. Dez. 2025. URL: https://docs.nordicsemi.com/bundle/nRF52811_PS_v1.2/resource/nRF52811_PS_v1.2.pdf.

[Nor25d] Nordic Semiconductor Aksjeselskap. *nRF52820-Datenblatt*. Englisch. Dez. 2025. URL: https://docs-be.nordicsemi.com/bundle/ps_nrf52820/attach/nRF52820_PS_v1.5.pdf?_LANG=enus.

[Nor25e] Nordic Semiconductor Aksjeselskap. *nRF52832-Datenblatt*. Englisch. Dez. 2025. URL: https://docs.nordicsemi.com/bundle/nRF52832_PS_v1.9/resource/nRF52832_PS_v1.9.pdf.

[Nor25f] Nordic Semiconductor Aksjeselskap. *nRF52840-Datenblatt*. Englisch. Dez. 2025. URL: https://docs-be.nordicsemi.com/bundle/ps_nrf52840/attach/nRF52840_PS_v1.11.pdf?_LANG=enus.

[Nor25g] Nordic Semiconductor Aksjeselskap. *nRF9160-Datenblatt*. Englisch. Dez. 2025. URL: https://docs-be.nordicsemi.com/bundle/ps_nrf9160/page/nRF9160_PS_v2.2.pdf?_LANG=enus.

[Pos25] Peter Posluschny. *Qualitätsmanagement für Führungskräfte: Instrumente, Methoden, qualitätssichernde Maßnahmen*. Deutsch. Wiesbaden: Springer Fachmedien Wiesbaden, 2025. ISBN: 978-3-658-46700-5 978-3-658-46701-2. DOI: 10.1007/978-3-658-46701-2.

[Ras25] Raspberry Pi Limited. *Raspberry Pi Pico W-Datenblatt*. Englisch. Dez. 2025. URL: https://pip-assets.raspberrypi.com/categories/686-raspberry-pi-pico-w/documents/RP-008312-DS-1-pico-w-datasheet.pdf?disposition=inline.

[SN25] Andy Stanford-Clark und Arlen Nipper. *MQTT*. Englisch. Softwareübersicht. Dez. 2025. URL: https://mqtt.org/ (besucht am 15. 12. 2025).

[Sou+22] João Sousa u. a. „Zero-defect manufacturing terminology standardization: Definition, improvement, and harmonization“. In: *Frontiers in Manufacturing Technology* 2 (Dez. 2022), S. 947474. ISSN: 2813-0359. DOI: 10.3389/fmtec.2022.947474.

[STM25a] STMicroelectronics Naamloze Vennootschap. *STM32205Fxx/STM32207Fxx-Datenblatt*. Englisch. Dez. 2025. URL: https://www.st.com/resource/en/datasheet/dm00037051.pdf.

[STM25b] STMicroelectronics Naamloze Vennootschap. *STM32405Fxx/STM32407Fxx-Datenblatt*. Englisch. Dez. 2025. URL: https://www.st.com/resource/en/datasheet/dm00037051.pdf.

[STM25c] STMicroelectronics Naamloze Vennootschap. *STM32F103x8/STM32F103xB-Datenblatt*. Englisch. Dez. 2025. URL: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf.

[STM25d] STMicroelectronics Naamloze Vennootschap. *STM32U535xx-Datenblatt*. Englisch. Dez. 2025. URL: https://www.st.com/resource/en/datasheet/stm32u535cb.pdf.

[STM25e] STMicroelectronics Naamloze Vennootschap. *STM32WL33xx-Datenblatt*. Englisch. Dez. 2025. URL: https://www.st.com/resource/en/datasheet/stm32wl33c8.pdf.

[Tex25] Texas Instruments Incorporated. *Texas Instruments CC3200-Datenblatt*. Englisch. Dez. 2025. URL: https://www.ti.com/lit/ds/symlink/cc3200.pdf?ts=1765779463706&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FCC3200.

[VF23] VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik und Fachbereich Optische Technologien. *VDI/VDE/VDMA 2632 Blatt 1*. Deutsch, Englisch. TD Anderes Dokument TR Andere technische Regel. Deutschland, Jan. 2023. URL: https://www.nautos.de/O7K/search/item-detail/DE88884204.

[VV25] VDI-Gesellschaft Produktion und Logistik und VDI-Fachbereich Fabrikplanung und -betrieb. *VDI 2889*. TD Anderes Dokument TR Andere technische Regel. Deutschland, Apr. 2025. URL: https://www.nautos.de/O7K/search/item-detail/DE45859322.

Übersicht verwendeter Hilfsmittel

Name	**Genutzter Funktionsumfang**	**Angabe der Textstelle**
ChatGPT	Rechtschreib- und Grammatikprüfung	Gesamte Bachelorarbeit
	Assistenz zur Erstellung und Verifizierung von Code	Listing 1, 2 und 3

8 Anhang

8.1 Listings

```
import network
import time
from umqtt.simple import MQTTClient
import ubinascii
import machine
import webrepl
from webrepl_client import webREPLClient
from machine import Pin, PWM, mem32

# WLAN verbinden
ssid = "WHZ-43536"
password = "7563Bp}2"

station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)

while not station.isconnected():
    time.sleep(1)

print("WLAN verbunden:", station.ifconfig())

PWM(18, freq=1000, duty_u16=65536 // 20)

pin22 = Pin(22, mode=Pin.OUT, pull=Pin.PULL_DOWN)
pin23 = Pin(23, mode=Pin.OUT, pull=Pin.PULL_DOWN)

def set_pins(pin_low, pin_high):
    # GPIO0-31 Output Register
    GPIO_OUT_REG = 0x3FF44004

    # Portwert aufbauen: nur *pin_high* HIGH, *pin_low* LOW
    # Alle anderen 0
    port_value = (0 << pin_low) | (1 << pin_high)

    # Ins Register schreiben
    mem32[GPIO_OUT_REG] = port_value
```

```
38
39  def an():
40      set_pins(22, 23)
41
42  def aus():
43      set_pins(23, 22)
44
45  def LED(delay):
46      an()
47      time.sleep(delay)
48      aus()
49
50  # WebREPL starten
51  webrepl.start()  # WebREPL aktivieren
52
53  # MQTT-Setup
54  client_id = ubinascii.hexlify(machine.unique_id())
55  mqtt_server = "141.32.208.185"
56  topic_sub = b'esp32/exec'
57
58  def sub_cb(topic, msg):
59      print(f"MQTT erhalten: {msg}")
60      try:
61          exec(msg.decode())  # Vorsicht!
62      except Exception as e:
63          print("Fehler bei MQTT-Ausfuehrung:", e)
64
65  client = MQTTClient(client_id, mqtt_server)
66  client.set_callback(sub_cb)
67  client.connect()
68  client.subscribe(topic_sub)
69
70  print(f"Warte auf MQTT ({topic_sub.decode()}) und WebREPL...")
71
72  webreplClient = webREPLClient('192.168.0.103', '123456789')
73
74  try:
75      while True:
76          client.check_msg()
77          time.sleep(0.1)
78  finally:
```

```
79      client.disconnect()
```

Listing 1: Boot-Datei, die beim Hochfahren des Mikrocontrollers sowie bei Zurücksetzung geladen und ausgeführt wird

```
1   import time
2   import websocket
3   import argparse
4
5   class InteractiveWebREPLClient:
6       def __init__(self, host: str, password: str, port: int = 8266):
7           self.host = host
8           self.port = port
9           self.password = password
10          self.ws = None
11
12      def connect(self):
13          print(f"Connecting to ws://{self.host}:{self.port}...")
14          self.ws = websocket.create_connection(f"ws://{self.host}:{self.
                port}")
15          self.ws.recv()
16          self.ws.send(self.password + "\n")
17          time.sleep(1)
18          self.ws.recv()
19          print("Connection established. You can now enter commands.")
20
21      def exec(self, code: str) -> str:
22          if not code.endswith('\n'):
23              code += '\n'
24          self.ws.send(b'\x05')
25          time.sleep(0.1)
26          self.ws.send(code.encode('utf-8') + b'\x04')
27          return self._read_response()
28
29      def _read_response(self) -> str:
30          output = b''
31
32          while True:
33              chunk = self.ws.recv()
34
35              if not chunk:
```

```
36                  break
37
38              chunk = chunk.encode('latin1') if isinstance(chunk, str) else
                    chunk
39              output += chunk
40
41              if chunk == b'>>> ':
42                  break
43
44          output = output.replace(b'>>> ', b'')
45
46          lines = output.decode('utf-8', errors='ignore').splitlines()
47
48          if lines and lines[-1].strip() == 'OK':
49              lines.pop()
50
51          return '\n'.join(lines)
52
53      def close(self):
54          if self.ws:
55              try:
56                  self.ws.send(b'\x02')
57                  time.sleep(0.1)
58              except Exception as e:
59                  print(f"Exception during close: {e}")
60              finally:
61                  self.ws.close()
62                  self.ws = None
63                  print("Connection closed.")
64
65  if __name__ == "__main__":
66      parser = argparse.ArgumentParser(
67          description="Funktion, die sich, mittels drahtloser Kommunikation
                , mit einem ESP32-Mikrocontroller verbindet und auf diesem
               Befehle ausfuehrt."
68      )
69      parser.add_argument("--esp_host", required=True, type=str)
70      parser.add_argument("--esp_password", required=True, type=str)
71      args = parser.parse_args()
72
73      esp_host = args.esp_host
```

```
74      esp_password = args.esp_password
75
76      client = InteractiveWebREPLClient(esp_host, esp_password)
77
78      try:
79          client.connect()
80
81          while True:
82              command = input("Enter Python command (or 'exit' to quit): ")
83              if command.strip().lower() == 'exit':
84                  break
85              result = client.exec(command)
86              print(">>> Output from ESP32:\n", result)
87
88
89      except Exception as e:
90          print("Error: ", e)
91      finally:
92          client.close()
```

Listing 2: Python-Programm, um eine Verbindung mit einem einzelnen ESP32-Mikrocontroller, mittels WebREPL, herzustellen

```
1   import websocket
2   import hashlib
3   import struct
4   import time
5
6   class webREPLClient:
7       def __init__(self, ip, password, port=8266):
8           self.ip = ip
9           self.port = port
10          self.password = password
11          self.ws = None
12
13      def connect(self):
14          url = f"ws://{self.ip}:{self.port}/"
15          self.ws = websocket.create_connection(url)
16          self._handshake()
17
18      def _handshake(self):
```

```
19          greeting = self.ws.recv()
20          if not greeting.startswith(b"Password:"):
21              raise Exception("Ungueltige Begruessung vom ESP32.")
22
23          pw_hash = hashlib.sha256(self.password.encode("utf-8")).digest()
24          self.ws.send(pw_hash[:9])
25          response = self.ws.recv()
26          if response != b'\x01':
27              raise Exception("Authentifizierung fehlgeschlagen.")
28
29          self._enter_raw_repl()
30
31      def _enter_raw_repl(self):
32          self.ws.send(b"\x02")
33          time.sleep(0.1)
34          self.ws.send(b"\x01")
35          time.sleep(0.1)
36          self._flush_output()
37
38      def _flush_output(self):
39          try:
40              while True:
41                  self.ws.settimeout(0.2)
42                  self.ws.recv()
43          except:
44              pass
45
46      def exec(self, command):
47          if not self.ws:
48              raise Exception("Nicht verbunden.")
49          if not command.endswith("\n"):
50              command += "\n"
51
52          data = command.encode("utf-8")
53          length = struct.pack(">I", len(data))
54          self.ws.send(b"\x05")
55          self.ws.send(length + data)
56          self.ws.send(b"\x04")
57
58          output = self._read_until(b'\x04')
59          return output.decode(errors="ignore")
```

```
60
61      def _read_until(self, end_marker):
62          buffer = b""
63          while True:
64              chunk = self.ws.recv()
65              buffer += chunk
66              if end_marker in buffer:
67                  break
68          return buffer
69
70      def close(self):
71          if self.ws:
72              self.ws.close()
73              self.ws = None
```

Listing 3: WebREPL-Client-Klasse, welche für die Kommunikation, mittels WebREPL, verwendet wird

```
1   try:
2       import usocket as socket
3   except:
4       import socket
5   import ustruct as struct
6   from ubinascii import hexlify
7
8   class MQTTException(Exception):
9       pass
10
11  class MQTTClient:
12
13      def __init__(self, client_id, server, port=0, user=None, password=
            None, keepalive=0,
14                   ssl=False, ssl_params={}):
15          if port == 0:
16              port = 8883 if ssl else 1883
17          self.client_id = client_id
18          self.sock = None
19          self.server = server
20          self.port = port
21          self.ssl = ssl
22          self.ssl_params = ssl_params
```

```
23          self.pid = 0
24          self.cb = None
25          self.user = user
26          self.pswd = password
27          self.keepalive = keepalive
28          self.lw_topic = None
29          self.lw_msg = None
30          self.lw_qos = 0
31          self.lw_retain = False
32
33      def _send_str(self, s):
34          self.sock.write(struct.pack("!H", len(s)))
35          self.sock.write(s)
36
37      def _recv_len(self):
38          n = 0
39          sh = 0
40          while 1:
41              b = self.sock.read(1)[0]
42              n |= (b & 0x7f) << sh
43              if not b & 0x80:
44                  return n
45              sh += 7
46
47      def set_callback(self, f):
48          self.cb = f
49
50      def set_last_will(self, topic, msg, retain=False, qos=0):
51          assert 0 <= qos <= 2
52          assert topic
53          self.lw_topic = topic
54          self.lw_msg = msg
55          self.lw_qos = qos
56          self.lw_retain = retain
57
58      def connect(self, clean_session=True):
59          self.sock = socket.socket()
60          addr = socket.getaddrinfo(self.server, self.port)[0][-1]
61          self.sock.connect(addr)
62          if self.ssl:
63              import ussl
```

```
64              self.sock = ussl.wrap_socket(self.sock, **self.ssl_params)
65          premsg = bytearray(b"\x10\0\0\0\0\0")
66          msg = bytearray(b"\x04MQTT\x04\x02\0\0")
67
68          sz = 10 + 2 + len(self.client_id)
69          msg[6] = clean_session << 1
70          if self.user is not None:
71              sz += 2 + len(self.user) + 2 + len(self.pswd)
72              msg[6] |= 0xC0
73          if self.keepalive:
74              assert self.keepalive < 65536
75              msg[7] |= self.keepalive >> 8
76              msg[8] |= self.keepalive & 0x00FF
77          if self.lw_topic:
78              sz += 2 + len(self.lw_topic) + 2 + len(self.lw_msg)
79              msg[6] |= 0x4 | (self.lw_qos & 0x1) << 3 | (self.lw_qos & 0x2
                    ) << 3
80              msg[6] |= self.lw_retain << 5
81
82          i = 1
83          while sz > 0x7f:
84              premsg[i] = (sz & 0x7f) | 0x80
85              sz >>= 7
86              i += 1
87          premsg[i] = sz
88
89          self.sock.write(premsg, i + 2)
90          self.sock.write(msg)
91          self._send_str(self.client_id)
92          if self.lw_topic:
93              self._send_str(self.lw_topic)
94              self._send_str(self.lw_msg)
95          if self.user is not None:
96              self._send_str(self.user)
97              self._send_str(self.pswd)
98          resp = self.sock.read(4)
99          assert resp[0] == 0x20 and resp[1] == 0x02
100         if resp[3] != 0:
101             raise MQTTException(resp[3])
102         return resp[2] & 1
103
```

```
104         def disconnect(self):
105             self.sock.write(b"\xe0\0")
106             self.sock.close()
107 
108         def ping(self):
109             self.sock.write(b"\xc0\0")
110 
111         def publish(self, topic, msg, retain=False, qos=0):
112             pkt = bytearray(b"\x30\0\0\0")
113             pkt[0] |= qos << 1 | retain
114             sz = 2 + len(topic) + len(msg)
115             if qos > 0:
116                 sz += 2
117             assert sz < 2097152
118             i = 1
119             while sz > 0x7f:
120                 pkt[i] = (sz & 0x7f) | 0x80
121                 sz >>= 7
122                 i += 1
123             pkt[i] = sz
124             self.sock.write(pkt, i + 1)
125             self._send_str(topic)
126             if qos > 0:
127                 self.pid += 1
128                 pid = self.pid
129                 struct.pack_into("!H", pkt, 0, pid)
130                 self.sock.write(pkt, 2)
131             self.sock.write(msg)
132             if qos == 1:
133                 while 1:
134                     op = self.wait_msg()
135                     if op == 0x40:
136                         sz = self.sock.read(1)
137                         assert sz == b"\x02"
138                         rcv_pid = self.sock.read(2)
139                         rcv_pid = rcv_pid[0] << 8 | rcv_pid[1]
140                         if pid == rcv_pid:
141                             return
142             elif qos == 2:
143                 assert 0
144 
```

```
145         def subscribe(self, topic, qos=0):
146             assert self.cb is not None, "Subscribe callback is not set"
147             pkt = bytearray(b"\x82\0\0\0")
148             self.pid += 1
149             struct.pack_into("!BH", pkt, 1, 2 + 2 + len(topic) + 1, self.pid)
150             self.sock.write(pkt)
151             self._send_str(topic)
152             self.sock.write(qos.to_bytes(1, "little"))
153             while 1:
154                 op = self.wait_msg()
155                 if op == 0x90:
156                     resp = self.sock.read(4)
157                     assert resp[1] == pkt[2] and resp[2] == pkt[3]
158                     if resp[3] == 0x80:
159                         raise MQTTException(resp[3])
160                     return
161 
162         # Wait for a single incoming MQTT message and process it.
163         # Subscribed messages are delivered to a callback previously
164         # set by .set_callback() method. Other (internal) MQTT
165         # messages processed internally.
166         def wait_msg(self):
167             res = self.sock.read(1)
168             self.sock.setblocking(True)
169             if res is None:
170                 return None
171             if res == b"":
172                 raise OSError(-1)
173             if res == b"\xd0":  # PINGRESP
174                 sz = self.sock.read(1)[0]
175                 assert sz == 0
176                 return None
177             op = res[0]
178             if op & 0xf0 != 0x30:
179                 return op
180             sz = self._recv_len()
181             topic_len = self.sock.read(2)
182             topic_len = (topic_len[0] << 8) | topic_len[1]
183             topic = self.sock.read(topic_len)
184             sz -= topic_len + 2
185             if op & 6:
```

```
186                 pid = self.sock.read(2)
187                 pid = pid[0] << 8 | pid[1]
188                 sz -= 2
189             msg = self.sock.read(sz)
190             self.cb(topic, msg)
191             if op & 6 == 2:
192                 pkt = bytearray(b"\x40\x02\0\0")
193                 struct.pack_into("!H", pkt, 2, pid)
194                 self.sock.write(pkt)
195             elif op & 6 == 4:
196                 assert 0
197 
198         # Checks whether a pending message from server is available.
199         # If not, returns immediately with None. Otherwise, does
200         # the same processing as wait_msg.
201         def check_msg(self):
202             self.sock.setblocking(False)
203             return self.wait_msg()
```

Listing 4: MQTT-Client-Klasse, welche für die Kommunikation, mittels MQTT, verwendet wird [lab17]

8.2 Weiterer Anhang

Mikrocontroller	ESP8266[1]	STM32-Serie[2]	Nordic Semiconductor nRF52-Serie[3]	Arduino Portenta H7[4]	ESP32[5]	Raspberry Pi Pico W[6]	nRF9160[7]	Texas Instruments CC3200[8]	PIC32-Serie[9]
Prozessorgeschwindigkeit	80 MHz (Übertaktung bis zu 160 MHz)	64 MHz bis 160 MHz (je nach Modell)	Bis zu 64 MHz	Dualcore: 480 MHz (Cortex-M7) & 240 MHz (Cortex-M4)	Dualcore, bis zu 240 MHz	Bis zu 133 MHz	64 MHz (Cortex-M33)	80 MHz (ARM Cortex-M4)	Bis zu 80 MHz
Ansteuerungsfähigkeit	Integriertes Wi-Fi, GPIOs, SPI, I2C, UART	Vielseitige GPIOs, ADCs, DACs, SPI, I2C, USB	BLE, viele GPIOs, SPI, I2C	Konfigurierbare IOs, unterstützt Arduino & Mbed OS	Integriertes Wi-Fi und Bluetooth, viele IOs	Breites Spektrum an IO-Schnittstellen, SPI, I2C, UART	LTE-M, NB-IoT, GPS, viele GPIOs	Einfache Internet-Anbindung über Wi-Fi, GPIOs, SPI, I2C	Breites Spektrum an IOs, USB, CAN
Einsatzfähigkeit in unterschiedlichen Umgebungen	Vorwiegend für IoT-Anwendungen im Innenbereich; begrenzt für rauere Umgebungen	Weitreichender Temperatureinsatzbereich; geeignet für industrielle Anwendungen	Entwickelt für Wearables; ideal für batteriebetriebene Anwendungen	Industrietauglich; bedient industrielle und kommerzielle Anwendungen	Vielseitig einsetzbar, auch in raueren Umgebungen mit geeignetem Gehäuse	Geeignet für Industrie- und Automobilanwendungen, robust und stabil	Ideal für IoT-Anwendungen, die Mobilfunk erfordern; niedriger Energieverbrauch	Vorwiegend für Bildungs- und Bastelprojekte	Einsatz in IoT-Geräten, die Wi-Fi-Konnektivität benötigen
Programmierumgebung	Arduino IDE, PlatformIO, ESP8266 SDK	STM32-CubeIDE, Keil, IAR Embedded Workbench	Segger Embedded Studio, Arm Keil	Arduino IDE, Visual Studio Code mit PlatformIO	Arduino IDE, Espressif IDF, PlatformIO, MicroPython	MicroPython, C/C++ SDK	Nordic SDK, Segger Embedded Studio	Code Composer Studio	MPLAB X IDE, Harmony Framework

Tabelle 8.1: Vergleich verschiedener Mikrocontroller-Varianten

[1] [Esp25b]
[2] [STM25e; STM25d; STM25c; STM25a; STM25b]
[3] [Nor25a; Nor25b; Nor25c; Nor25d; Nor25e; Nor25f]
[4] [Ard25]
[5] [Esp25a]
[6] [Ras25]
[7] [Nor25g]
[8] [Tex25]
[9] [Mic25b; Mic25a]

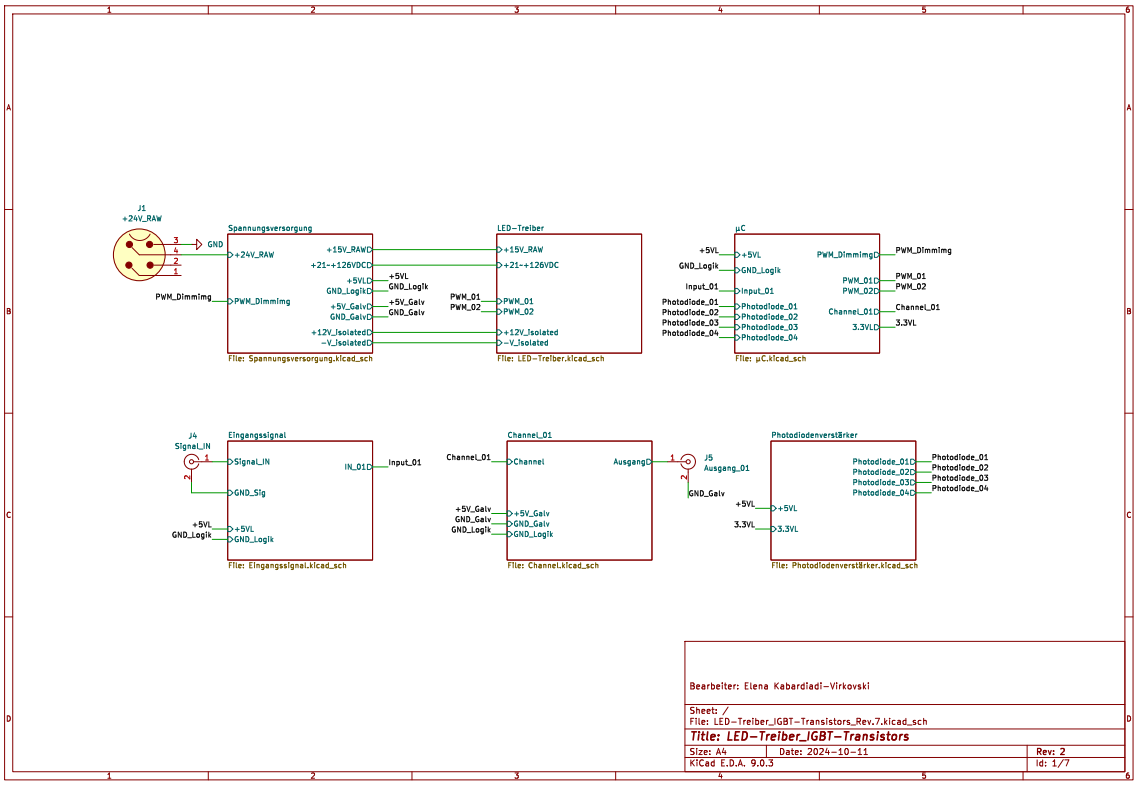


Abbildung 8.1: Schaltplan, der die Zusammensetzung der einzelnen elektrischen Komponenten des Systems darstellt, Seite 1

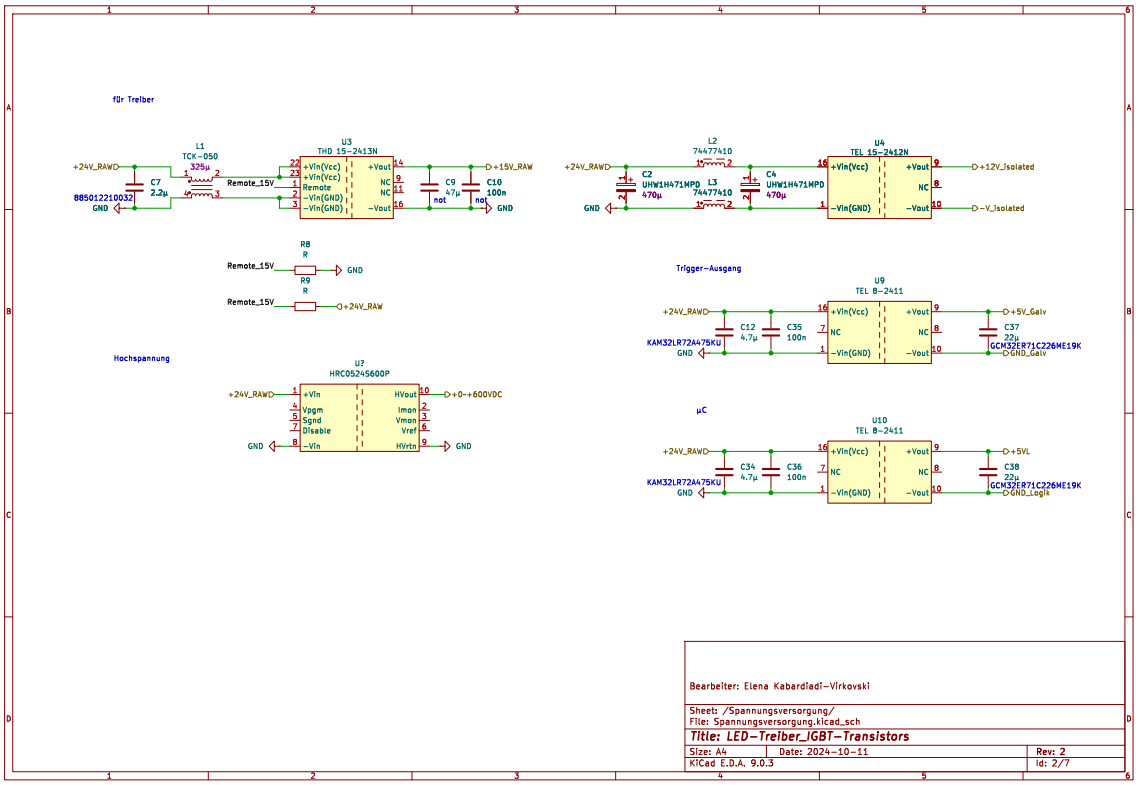


Abbildung 8.2: Schaltplan, der die Zusammensetzung der einzelnen elektrischen Komponenten des Systems darstellt, Seite 2

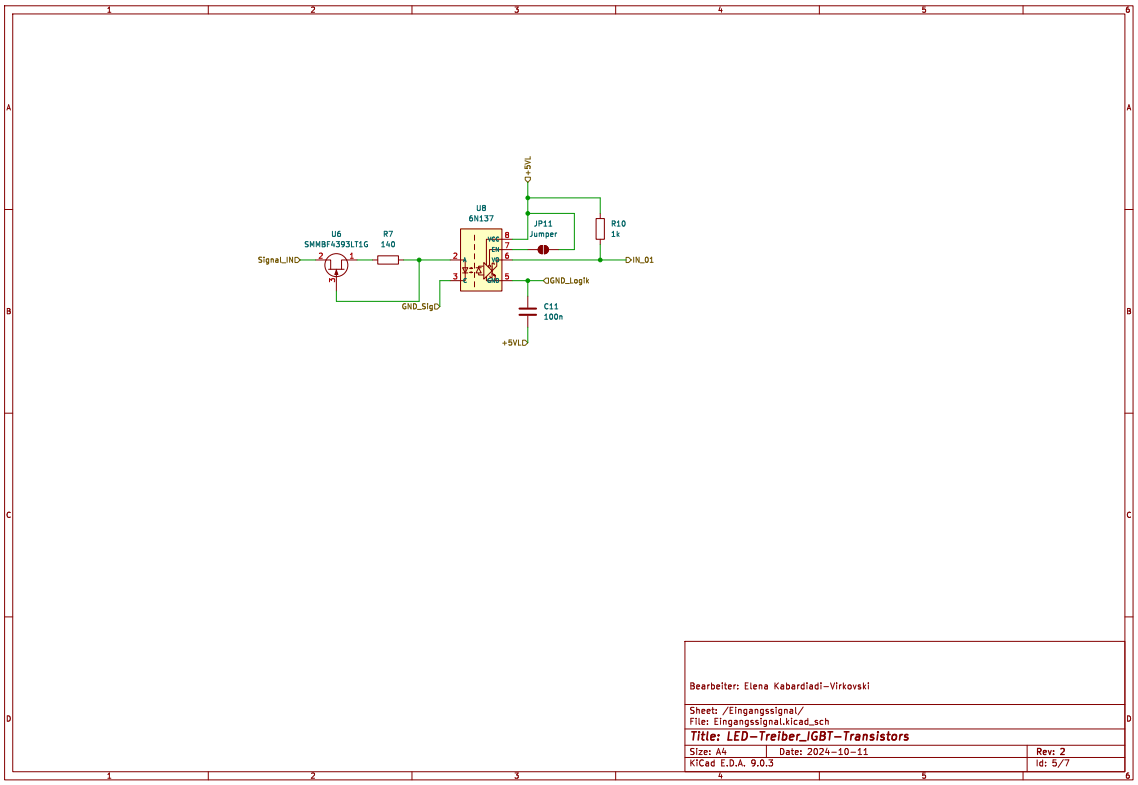


Abbildung 8.3: Schaltplan, der die Zusammensetzung der einzelnen elektrischen Komponenten des Systems darstellt, Seite 3

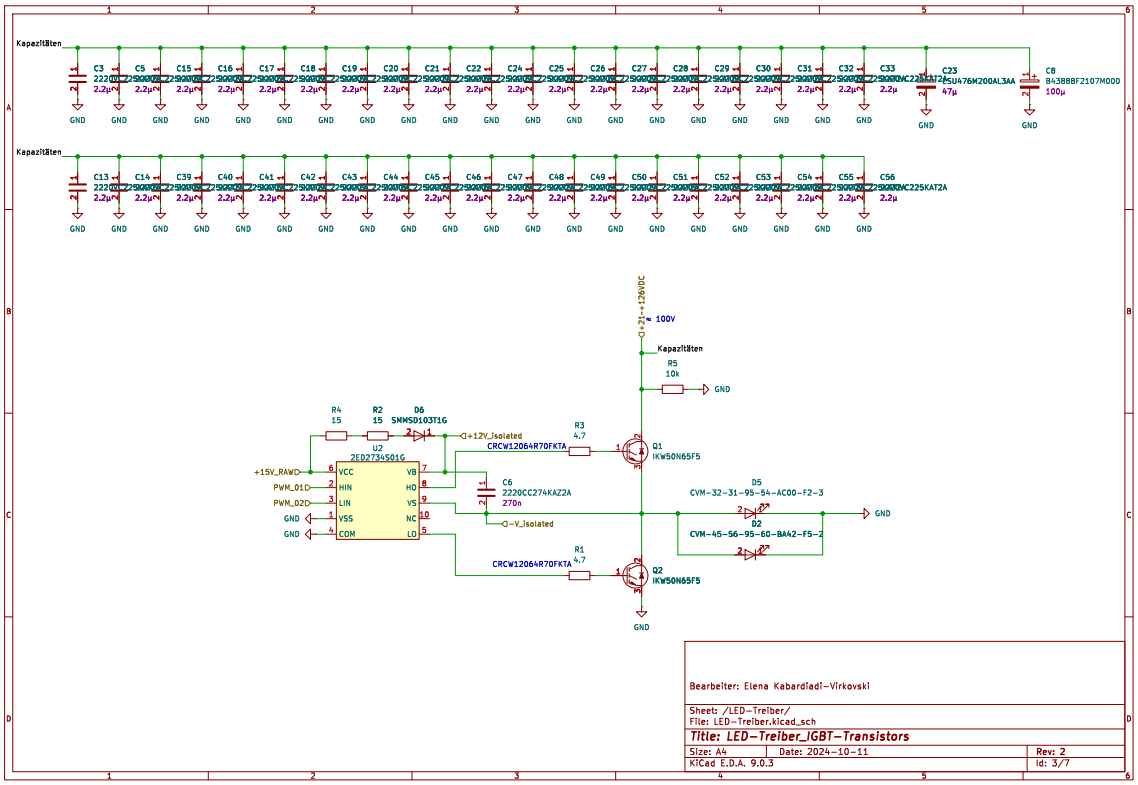


Abbildung 8.4: Schaltplan, der die Zusammensetzung der einzelnen elektrischen Komponenten des Systems darstellt, Seite 4

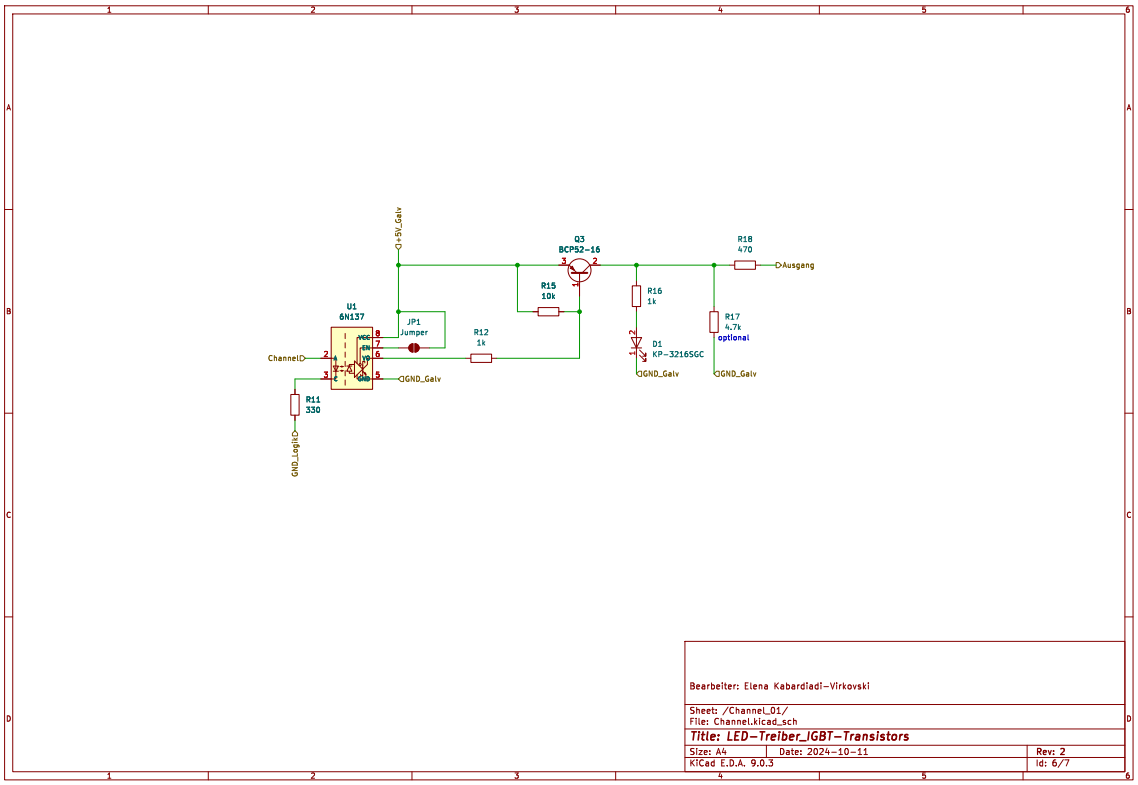


Abbildung 8.5: Schaltplan, der die Zusammensetzung der einzelnen elektrischen Komponenten des Systems darstellt, Seite 5

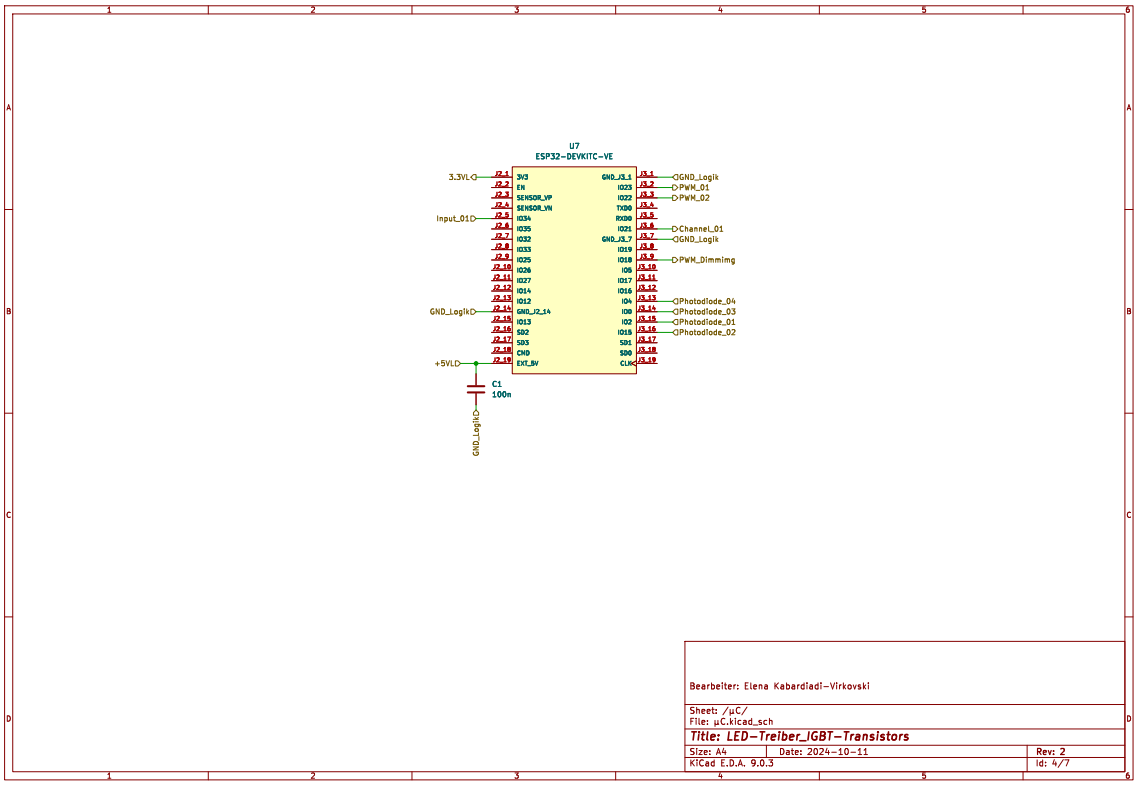


Abbildung 8.6: Schaltplan, der die Zusammensetzung der einzelnen elektrischen Komponenten des Systems darstellt, Seite 6

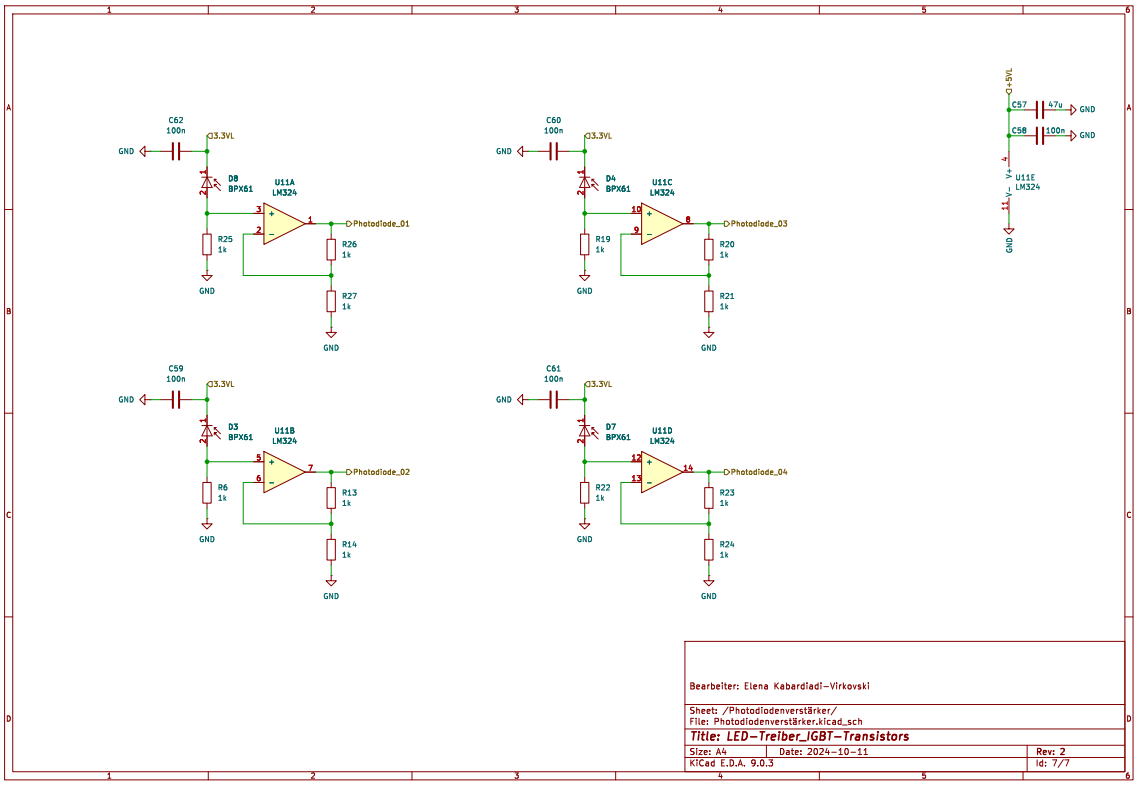


Abbildung 8.7: Schaltplan, der die Zusammensetzung der einzelnen elektrischen Komponenten des Systems darstellt, Seite 7

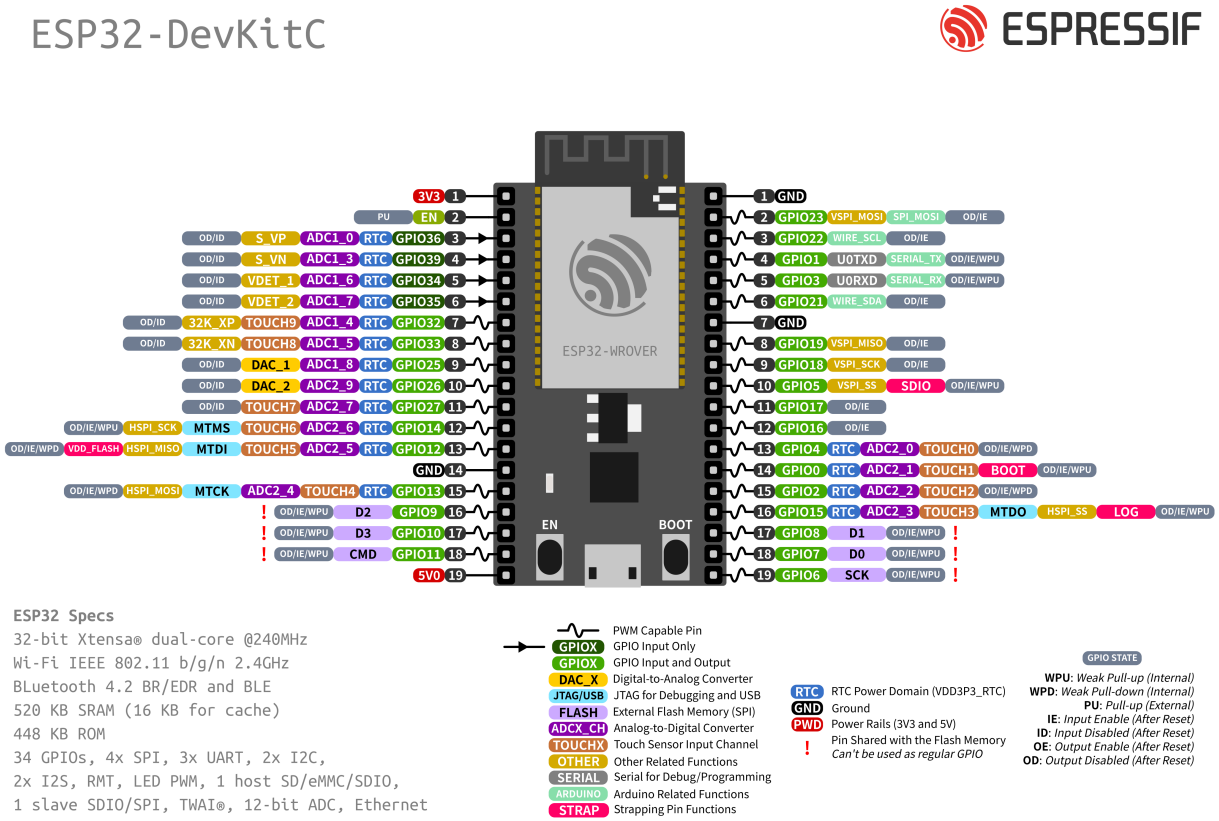


Abbildung 8.8: Übersicht aller Pins des ESP32-Mikrocontrollers und der dazugehörigen Funktionsgruppen, der einzelnen Pins, mit den jeweiligen Beschreibungen der einzelnen Funktionsgruppen [Esp22]

Selbstständigkeitserklärung

WHZ Westsächsische Hochschule Zwickau
Hochschule für Mobilität

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit in allen Teilen selbstständig angefertigt und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt habe, und dass die Arbeit in gleicher oder ähnlicher Form in noch keiner anderen Prüfung vorgelegen hat. Mir ist bewusst, dass ich Autor/in der vorliegenden Arbeit bin und volle Verantwortung für den Text trage.	☒
Ich erkläre, dass ich wörtlich oder sinngemäß aus anderen Werken – dazu gehören auch Internetquellen – übernommene Inhalte als solche kenntlich gemacht und die entsprechenden Quellen angegeben habe.	☒
Mir ist bewusst, dass meine Arbeit auf Plagiate überprüft werden kann. Mir ist bekannt, dass es sich bei der Abgabe eines Plagiats um ein schweres akademisches Fehlverhalten handelt und dass Täuschungen nach der für mich gültigen Prüfungsordnung geahndet werden.	☒
Zusätzlich versichere ich, dass ich auf künstlicher Intelligenz (KI) basierende Werkzeuge nur in Absprache mit den Prüfern verwendet habe. Dabei stand meine eigene geistige Leistung im Vordergrund, und ich habe jederzeit den Prozess steuernd bearbeitet.	☒
Diese Werkzeuge habe ich im Quellenverzeichnis in der Rubrik „Übersicht verwendeter Hilfsmittel“ mit ihrem Produktnamen und einer Übersicht des im Rahmen dieser Prüfungs-/Studienarbeit genutzten Funktionsumfangs unter Angabe der Textstelle in der Arbeit vollständig aufgeführt.	☒
Ich versichere, dass ich keine KI-basierten Tools verwendet habe, deren Nutzung die Prüfer explizit schriftlich ausgeschlossen haben. Ich bin mir bewusst, dass die Verwendung von Texten oder anderen Inhalten und Produkten, die durch KI-basierte Tools generiert wurden, keine Garantie für deren Qualität darstellt.	☒
Ich verantworte die Übernahme jeglicher von mir verwendeter maschinell generierter Passagen vollumfänglich selbst und trage die Verantwortung für eventuell durch die KI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.	☒

Zwickau, 02.01.2026 — A. Sicheneder

Ort, Datum — Unterschrift