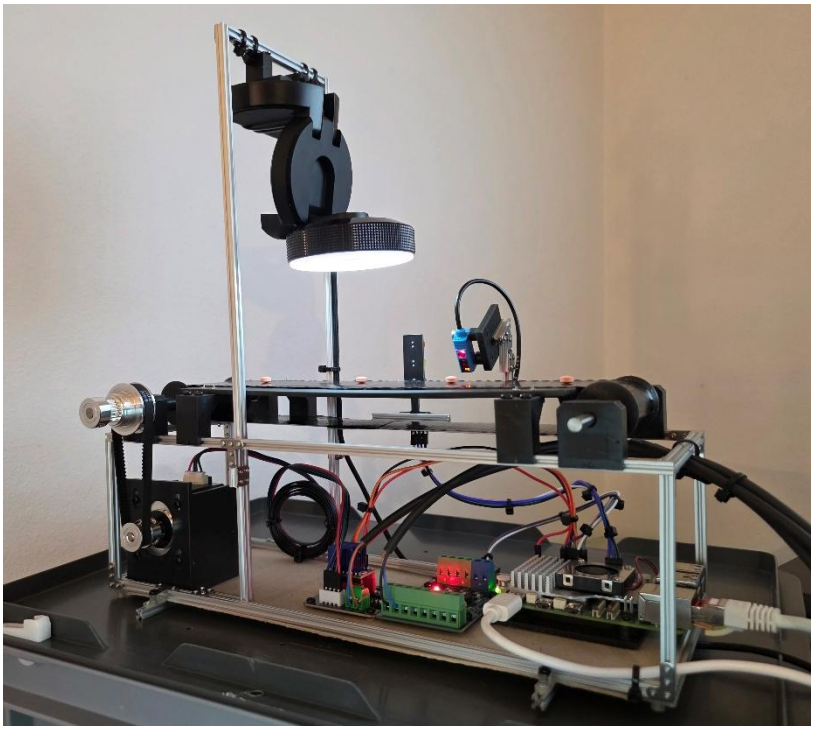
Schweizerische Fachschule TEKO



Edge-AI Prüfband

Entwicklung eines Edge-AI-basierten Computer-Vision-Demonstrators zur Qualitätskontrolle pharmazeutischer Produkte mit IoT-/Cloud-Anbindung

Diplomarbeit 2026

Autor & Diplomand:	Christian Kuhn
Diplomexperte:	Dominic Merz
Diplomcoach:	Stephan Kessler
Klassifizierung:	Öffentlich
Schule, Klasse:	TEKO Zürich, Z-TIP-23-T-a
Ort, Datum:	Zürich, 28. Oktober 2026

Inhaltsverzeichnis

1. Aufgabenstellung 10
1.1. Hintergrund 10
1.2. Motivation 10
1.3. Problemstellung 10
1.4. Lösungsidee 11
1.5. Aufgabe 11
1.6. Stakeholder (Interessengruppen) 11
1.7. Vorkenntnisse 12
1.8. Aufbau der Arbeit 12
2. Pflichtenheft und Zieldefinition 13
2.1. Ziele und Erfolgskriterien 13
2.2. Anforderungen 14
2.2.1. Funktionale Anforderungen 14
2.2.2. Nicht-funktionale Anforderungen 14
2.3. Rahmenbedingungen 14
2.4. Abgrenzung 15
2.5. Annahmen 15
2.6. Akzeptanzkriterien 15
2.7. Zu erbringende Leistungen 15
3. Terminplan SOLL/IST und Projektrisiken 16
3.1. Vorgehensmodell 16
3.2. Projektorganisation 17
3.3. Projektstrukturplan 18
3.4. Terminplan SOLL und Meilensteine 19
3.5. Projektsteuerung 20
3.6. Projektrisiken 20
3.7. Terminplan IST und Soll-Ist-Vergleich 21
3.8. Ereignisse 22
3.9. Werkzeuge 23
4. Analyse und Grundlagen 24
4.1. Sichtprüfung in der pharmazeutischen Industrie 24

4.2. Computer Vision: Klassifizierung und Objekterkennung 24
4.3. YOLO und Transfer Learning 26
4.4. Edge-Computing 27
4.5. Warum ein Desktop-PC mit einer RTX 3060 Ti für KI geeignet ist 27
4.6. Der Hardware-Markt unter dem Einfluss der KI 28
4.6.1. Grafikkarten 28
4.6.2. Arbeitsspeicher und Speichermedien 28
4.6.3. Die Auswirkungen auf Verbraucher und Unternehmen 28
4.6.4. Auswirkungen auf dieses Projekt 29
4.7. Schrittmotoren und Motortreiber 29
4.8. IoT, MQTT und Cloud 30
4.9. Systemkontext und Anwendungsfälle 31
5. Evaluation 32
5.1. Methode 32
5.2. Edge-Computer für die KI-Auswertung 32
5.2.1. Erste Evaluation 32
5.2.2. Neubewertung nach einer Preiserhöhung 33
5.3. Kamera 34
5.4. Auslösesensor 34
5.5. Antrieb und Mechanik 35
5.6. Steuerung und Elektronik 35
5.7. KI-Framework und Annotation 36
5.8. Cloud-Plattform 37
5.8.1. Erste Evaluation 37
5.8.2. Neubewertung nach der Einstellung 37
5.8.3. Visualisierung der Prüfdaten 38
5.9. Entscheidungsübersicht 39
6. Systemkonzept 40
6.1. Systemübersicht und Architekturprinzipien 40
6.2. Netzwerk- und IP-Konzept 41
6.3. Komponentenmodell Hardware 43
6.4. Komponentenmodell Software 43

6.5. Prüfablauf 44
6.6. Prüflogik 45
6.7. Datenmodell 46
6.8. Sicherheitskonzept 47
6.9. Cloud-Konzept 48
6.9.1. IoT Hub 48
6.9.2. Namenskonvention 48
6.9.3. Bewusste Abweichungen 48
6.9.4. Fabric 48
6.9.5. Kostenüberwachung 48
6.10. Elektrisches Konzept 49
6.11. Mechanisches Konzept 49
7. Machbarkeitsstudie 51
7.1. Ziel und Vorgehensweise 51
7.2. Machbarkeit der Bildverarbeitung 51
7.2.1. Testaufbau 51
7.2.2. Datensatz und Annotation 52
7.2.3. Training 53
7.2.4. Prüflogik 54
7.2.5. Testdurchführung und Ergebnisse 55
7.2.6. Auswertung 57
7.3. Machbarkeit der Cloud 57
7.3.1. Testaufbau 57
7.3.2. Testdurchführung und Ergebnisse 57
7.3.3. Auswertung 58
7.4. Ergebnisse und Entscheidungen 58
7.4.1. Vergleich der beiden Ansätze 58
7.5. Fazit 59
8. Umsetzung 59
8.1. Herstellung der gedruckten Teile 59
8.2. Testaufbau der Elektronik 60
8.3. Inbetriebnahme des Raspberry Pi 61

8.4. Inbetriebnahme des Edge-PCs und des MQTT-Brokers 62
8.5. Mechanische Montage und Justierung 62
8.6. Software des Raspberry Pi 5 63
8.7. Prüfsoftware auf dem Edge-PC 66
8.8. Best Practices im Code 68
8.9. Bilddatensatz und Training des KI-Modells 69
8.9.1. Klassen 69
8.9.2. Erfassung im realen Prüfzyklus 69
8.9.3. Probleme mit den Bildern 69
8.9.4. Vorverarbeitung 70
8.9.5. Training 71
8.10. Cloud-Anbindung und Visualisierung 75
8.10.1. IoT Hub 75
8.10.2. Microsoft Fabric 76
8.11. Inbetriebnahme und Betrieb 78
8.11.1. Start 78
8.11.2. Stopp 78
8.11.3. Beobachtung 78
9. Tests 78
9.1. Testkonzept 78
9.1.1. Testziele 78
9.1.2. Teststrategie und Testebenen 78
9.1.3. Testumgebung 79
9.1.4. Fehler und Kriterien 79
9.2. Testfälle 80
9.3. Testdurchführung und Ergebnisse 81
9.3.1. Finale Komponenten- und Integrationstests 81
9.3.2. Systemtest 81
9.3.3. Vollständigkeit in der Cloud (T09) 81
9.3.4. Betriebsdaten (EK4) 82
9.3.5. Dauerbetrieb (T10) 82
9.4. Gesamtbewertung 82

10. Wirtschaftlichkeitsanalyse 83
10.1. Annahmen 83
10.2. Kosten des Prototyps 83
10.3. Bewertung als Einzelanfertigung 85
10.4. Serienprodukt 85
10.4.1. Nicht für die pharmazeutische Industrie 85
10.4.2. Ein Schulungskit für Vision-AI im pharmazeutischen Bereich 85
10.4.3. Was sich bei einem Serienprodukt ändern würde 86
10.4.4. Cloud für eine Markteinführung 86
10.4.5. Break-even 87
11. Quellen- und Literaturverzeichnis 88
12. Verzeichnisse 89
12.1. Abbildungsverzeichnis 89
12.2. Glossar 90
12.3. Abkürzungsverzeichnis 91
13. Reflexion 92
13.1. Weg zum Ziel 92
13.1.1. Massnahmen, die funktioniert haben 92
13.2. Gespräche mit dem Diplomcoach 93
13.2.1. Gespräch 1 (08.10.2026) 93
13.2.2. Gespräch 2 (22.10.2026) 93
13.3. Zielerreichung gemäss Pflichtenheft 94
13.3.1. Erfolgskriterien 94
13.3.2. Anforderungen 95
13.4. Lessons learnt 96
13.4.1. Technisch 96
13.4.2. Projekt 96
13.4.3. Künstliche Intelligenz (Claude von Anthropic) 96
13.5. Ausblick 97
13.5.1. Mögliche Erweiterungen 97
13.6. Eigenständigkeitserklärung 98
13.7. KI-Deklaration 98

13.8. KI-Promptverzeichnis 99

14. Fazit 101

Management Summary

Das «Prüfband» ist ein funktionierender Tischdemonstrator, der einzelne Tabletten mithilfe von KI prüft, innerhalb von Millisekunden lokal entscheidet und jedes Ergebnis an die Microsoft-Cloud meldet. Im abschliessenden Systemtest hat es jede defekte Tablette aussortiert (0 Schlupf) und jede gute akzeptiert.

Ausgangspunkt

Prüfsysteme in der pharmazeutischen Industrie sind zuverlässig, aber teuer und geschlossen. Drei Entwicklungen senken die Einstiegshürde: vortrainierte KI-Modelle, die sich mit wenigen eigenen Bildern an eine neue Aufgabe anpassen, Edge-Computing direkt an der Maschine und standardisierte IoT-Cloud-Dienste. Wer jedoch Vision-AI-Software entwickeln und testen möchte, findet auf dem breiten Markt keinen kostengünstigen Demonstrator für die Praxis: Förderband, Sensor und Signalsäule müssen erst aufgebaut werden. Ein solches Prüfband kann sich auch allein für Schulungszwecke lohnen, für Personen, die sich mit Vision-AI vertraut machen möchten. Aus diesem Grund befasst sich Kapitel 10 damit, wie das Prüfband marktreif gemacht werden könnte. Marktreife kann zwar in Betracht gezogen werden, war jedoch nicht das Ziel dieser Arbeit: Das Prüfband ist vor allem das persönliche Projekt des Diplomanden, um Wissen aufzubauen und die Möglichkeiten der Vision-AI zu erleben.

Lösung

Ein kurzes Förderband, angetrieben von einem Schrittmotor, befördert jeweils eine Tablette. Ein Sensor erkennt die Tablette, ein Raspberry Pi 5 hält sie unter einer Webcam an und fordert über MQTT eine Prüfung an. Ein Edge-PC mit einer NVIDIA RTX 3060 Ti klassifiziert das Bild mit einem YOLO11-Modell in drei Klassen: IO (in Ordnung), NIO - broken (gebrochen) und NIO - wrong (falsche Tablette). Eine Signalsäule zeigt das Ergebnis an. Ergebnisse und Betriebsdaten werden an Azure IoT Hub gesendet und in einem Real-Time Dashboard von Microsoft Fabric visualisiert.

Vorgehensweise

Das Projekt folgte dem Vorgehensmodell IPERKA. Eine Machbarkeitsstudie mit Tablettenblistern bestätigte die Bildverarbeitung und die Cloud-Anbindung, bevor Hardware gebaut wurde. Alle Komponenten wurden anschliessend vor der Endmontage lose auf dem Tisch getestet.

Ergebnisse

Messgrösse	Ziel	Ergebnis
Schlupf (defekte Tablette als gut bewertet)	0	0%
Gute Tabletten als gut bewertet	mindestens 95 %	100%
Korrekte IO/NIO-Entscheidung	-	100%
KI-Prüfzeit pro Bild	höchstens 100 ms	5,7 ms
Ergebnisse in der Cloud	100 %	100%
Betrieb ohne Internetverbindung	Prüfung läuft weiter	Ja, die Prüfung läuft weiter

Alle fünf Erfolgskriterien der genehmigten Themeneingabe sind erfüllt, EK5 mit einer Einschränkung beim Dauerbetrieb. Der Systemtest mit 27 Tabletten bestätigte die Funktion, sodass er ohne weitere Durchläufe abgeschlossen wurde. Die grössten Einschränkungen sind die handelsübliche Webcam und wechselnde Lichtverhältnisse.

Erkenntnisse und Ausblick

Bilddaten sind das wertvollste Kapital eines Vision-AI-Systems. Bessere Beleuchtung, eine bessere Kamera und ein besseres Förderband würden das System am stärksten verbessern. Als Einzelanfertigung ist der Demonstrator nicht wirtschaftlich: Etwa 201 bis 236 Stunden Entwicklungsaufwand zu einem internen Stundensatz von CHF 100 entsprechen mit dem Material rund CHF 20’400 bis 23’900. Als kostengünstiges Schulungskit könnte eine vereinfachte Version jedoch einen Markt bei Personen finden, die Vision-AI anhand von Anwendungsfällen aus der Pharmaindustrie erlernen möchten.

1. Aufgabenstellung

1.1. Hintergrund

Jedes pharmazeutische Produkt, das eine Produktionslinie verlässt, sollte einer Qualitätsprüfung unterzogen werden. Gebrochene, abgesplitterte oder fehlende Tabletten dürfen niemals in die Verpackung gelangen. Prüfsysteme prüfen daher jede Einheit mit Industriekameras, gesteuerter Beleuchtung und regelbasierter Bildverarbeitung. Diese Systeme arbeiten zuverlässig, sind jedoch teuer, geschlossen und stehen selten für Schulungen oder Experimente zur Verfügung.

Drei Entwicklungen senken nun die Einstiegshürde:

- Ein vortrainiertes Modell lässt sich mit nur wenigen hundert eigenen Bildern an eine neue Prüfaufgabe anpassen. Somit wird KI-basierte Bilderkennung deutlich zugänglicher.
- Ergebnisse können mit Standardprotokollen in die Cloud übertragen werden, wo sie sicher gespeichert und analysiert werden.
- Mittels Edge-Computing wird das Bild direkt neben der Maschine ausgewertet. Entscheidungen erfolgen schnell und sind nicht von einer Internetverbindung abhängig.

1.2. Motivation

Der Diplomand absolvierte seine Lehre bei Bosch Packaging Technology (heute Syntegon). Dort baute und konfigurierte er unter anderem Verpackungs- und Inspektionsmaschinen für die Pharmaindustrie. Heute arbeitet er nicht mehr in der Industrie, sondern in der Unternehmens-IT und verfügt über fundierte Kenntnisse im Bereich Cloud-Dienste. Diese Diplomarbeit verbindet beide Bereiche. Sie ermöglicht ihm zudem den Einstieg in das maschinelle Lernen, ein Gebiet, in dem er bisher keine Erfahrung hatte, und vertieft seine IoT-Grundkenntnisse.

Das Ziel ist nicht nur die Maschine mit ihrer Hardware und Software an sich, sondern auch der Weg dorthin: Projektmanagement, Methoden und Werkzeuge wie ProjectLibre, draw.io sowie KI-Werkzeuge wie Claude und DeepL Write. Als Studierender einer Höheren Fachschule (HF) legt der Diplomand bewusst Wert auf eine strukturierte Vorgehensweise, Planung und Risikokontrolle.

1.3. Problemstellung

Es gibt keinen kostengünstigen, transparenten Demonstrator, der anhand eines konkreten pharmazeutischen Beispiels zeigt, wie KI-Bilderkennung, Edge-Computing und Cloud-Anbindung zusammenwirken, sodass Menschen experimentieren und Know-how aufbauen können. Ein solcher Demonstrator sollte zudem aufzeigen, wozu heutige Standardkomponenten fähig sind, wo ihre Grenzen liegen und welche Designentscheidungen eine sichere Prüfung erfordert.

1.4. Lösungsidee

Eine kleine Prüflinie, in dieser Arbeit «Prüfband» genannt, befördert Tabletten auf einem kurzen Förderband und hält jede einzelne unter einer Kamera an. Ein trainiertes KI-Modell auf einem Edge-PC klassifiziert die Tablette, eine Signalsäule zeigt das Ergebnis an, und jedes Ergebnis wird an eine Cloud-Plattform gemeldet. Ein Raspberry Pi 5 steuert das Förderband, liest den Sensor aus und schaltet die Signalsäule.

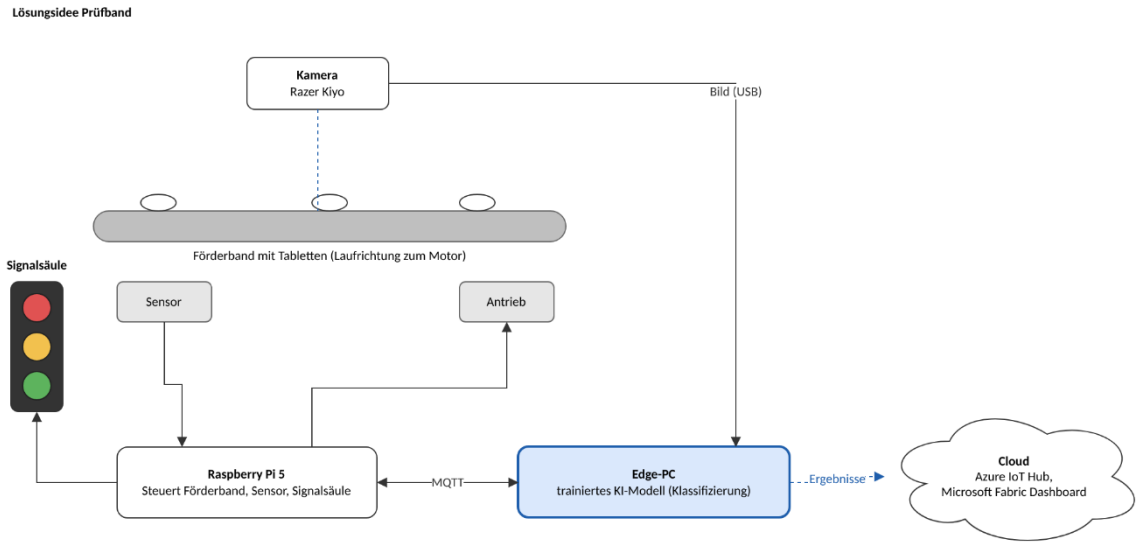


Abbildung 1: Lösungsidee (mit KI generierte Darstellung, draw.io)

1.5. Aufgabe

Ziel ist ein funktionsfähiger Demonstrator, das «Prüfband», zur KI-basierten Prüfung von Tabletten. Es besteht aus fünf Teilsystemen:

- Ein etwa 26 cm langes Förderband (mit Rahmen etwa 30 cm), angetrieben durch einen Schrittmotor, das die Tabletten einzeln unter die Kamera transportiert.
- Ein Auslösesensor, der eine ankommende Tablette erkennt und die Prüfung startet.
- Bildaufnahme und KI-Auswertung in Echtzeit auf einem Edge-Computer.
- Ein Raspberry Pi 5, der den Motor, den Sensor und die Signalsäule steuert.
- Eine Cloud-Anbindung, die die Ergebnisse sicher überträgt und auswertet.

1.6. Stakeholder (Interessengruppen)

Stakeholder	Interesse	Einfluss
Diplomand	Bestehen des Diploms, Aufbau von Kompetenzen in KI und IoT	Hoch
Diplomexperte	Eine nachvollziehbare Diplomarbeit auf HF-Niveau	Hoch
Diplomcoach	Begleitung, Qualität, Einhaltung von Fristen	Mittel
Arbeitgeber	Vereinbarkeit mit der täglichen Arbeit, Druck von 3D-Teilen	Mittel
Lieferanten	Lieferung von Komponenten	Gering

1.7. Vorkenntnisse

Fachgebiet	Niveau vor dem Projekt
Cloud (Azure)	Sehr gut
Projektmanagement	Gut, aus Studium und Beruf
Python	Grundkenntnisse
Maschinelles Lernen, Bilderkennung (Vision-AI)	Keine
IoT, MQTT	Grundkenntnisse
Elektronik, Mechanik	Grundkenntnisse

1.8. Aufbau der Arbeit

Die Kapitel folgen der Struktur der TEKO-Vorgabe und dem Verlauf des Projekts:

- Aufgabenstellung (Kapitel 1)
- Pflichtenheft und Zieldefinition (2)
- Terminplan SOLL/IST und Projektrisiken (3)
- Die Lösung der Aufgabe umfasst die Grundlagen (4)
- die Evaluation und das Systemkonzept (5 und 6)
- die Machbarkeitsstudie (7)
- die Umsetzung und die Tests (8 und 9)
- sowie die Wirtschaftlichkeitsanalyse (10)
- Danach folgen das Quellen- und Literaturverzeichnis (11)
- weitere Verzeichnisse (12)
- die Reflexion mit Lessons learnt und Kommentar zu KI (13)
- das persönliche Schlusswort (14)
- sowie die Eigenständigkeitserklärung (15)

2. Pflichtenheft und Zieldefinition

2.1. Ziele und Erfolgskriterien

Die Erfolgskriterien stammen aus der genehmigten Themeneingabe. Sie entscheiden darüber, ob die Diplomarbeit ihr Ziel erreicht hat.

Nr.	Erfolgskriterium	Nachweis
EK1	Evaluation der Komponenten und der Plattform: Die Auswahl des Edge-Geräts, der Kamera, der Förderlösung und der Cloud-Plattform ist anhand definierter Kriterien dokumentiert.	Kapitel 5
EK2	Funktionierender Demonstrator: Am Ende der Diplomarbeit ist ein Computer-Vision-Demonstrator mit Förderband, Kamera und lokalem Edge-Gerät aufgebaut und betriebsbereit.	Kapitel 8, Systemtest
EK3	Lokale Bilderkennung: Das System erkennt mindestens zwei definierte Zustände von pharmazeutischen Produkten, zum Beispiel «gut» und «Fehler erkannt», lokal auf dem Edge-Gerät.	Systemtest
EK4	Cloud-Anbindung: Betriebsdaten und Erkennungsereignisse werden erfolgreich an die ausgewählte IoT-/Cloud-Plattform übertragen.	Systemtest
EK5	Test und Auswertung: Erkennungsqualität, Latenz und Systemstabilität werden in einem dokumentierten Testlauf gemessen und ausgewertet.	Kapitel 9

Um EK3 bis EK5 messbar zu machen, werden im Testlauf die folgenden Zielwerte verwendet:

Nr.	Ziel	Metrik	Wert	Referenz
Z1	Keine defekte Tablette wird als gut bewertet	Schlupf	0	EK3, EK5
Z2	Zuverlässige Erkennung im Normalbetrieb	Anteil korrekter Entscheidungen	Mindestens 95 %	EK3, EK5
Z3	Echtzeitprüfung	KI-Prüfzeit pro Bild	Höchstens 100 ms	EK5
Z4	Vollständige Übertragung in die Cloud	Anteil der bei bestehender Verbindung übertragenen Ergebnisse	100 %	EK4
Z5	Stabiler Betrieb	Dauerbetrieb ohne Eingriff	Kein Ausfall	EK2, EK5

2.2. Anforderungen

2.2.1. Funktionale Anforderungen

ID	Anforderung	Priorität
F01	Das Förderband transportiert Tabletten mit einer einstellbaren Geschwindigkeit.	Muss
F02	Ein Sensor erkennt jede ankommende Tablette und löst die Prüfung aus.	Muss
F03	Nach dem Sensorsignal hält das Förderband die Tablette an einer wiederholbaren Position unter der Kamera an.	Muss
F04	Das KI-Modell klassifiziert jede Tablette als gut (IO) oder nicht gut (NIO).	Muss
F05	Eine unsichere Erkennung führt zu NIO.	Muss
F06	Eine Signalsäule zeigt den Status an.	Muss
F07	Jedes Ergebnis wird mit einem Zeitstempel an die Cloud gemeldet.	Muss
F08	Bei Ausfall der Internetverbindung setzt das Prüfband die Prüfung fort und signalisiert weiterhin lokal.	Muss
F09	Die Cloud zeigt die Ergebnisse in einer übersichtlichen Darstellung an.	Wunsch
F10	Prüfbilder werden lokal gespeichert.	Wunsch

2.2.2. Nicht-funktionale Anforderungen

ID	Anforderung	Priorität
N01	Alle Verbindungen zur Cloud sind verschlüsselt.	Muss
N02	Anmeldedaten werden getrennt vom Programmcode gespeichert.	Muss
N03	Das Prüfband verwendet Standardkomponenten und offene Protokolle.	Muss
N04	Cloud-Dienste laufen im kostenlosen Tarif und werden überwacht.	Muss
N05	Das Prüfband ist portabel und läuft auf einem Tisch.	Muss
N06	Einrichtung und Software sind so detailliert dokumentiert, dass eine Fachperson das Prüfband ohne weitere Unterstützung durch den Diplomanden nachbauen kann.	Wunsch

2.3. Rahmenbedingungen

Bereich	Rahmenbedingung
Technisch	Einige Geräte sind bereits zu Hause vorhanden und werden bei der Evaluation als Optionen berücksichtigt (Kapitel 5). Alles andere wird nach der Evaluation beschafft. Die Cloud-Umgebung wird im Laufe des Projekts eingerichtet.
Organisatorisch	Die Diplomarbeit ist eine Einzelarbeit, die neben einer 80-%-Stelle und Teilzeit-IT-Mandaten erledigt wird. Die Themeneingabe war ab dem 25.05.2026 möglich und musste bis zum 08.07.2026 erfolgen. Die Termine der Schule sind festgelegt: Einreichung am 28.10.2026 und Präsentation am 19.11.2026.
Umgebung	Die Entwicklung erfolgt zu Hause und am Arbeitsplatz. Am Präsentationsort muss das Prüfband ohne ein Fremdnetzwerk funktionieren.

2.4. Abgrenzung

Die folgenden Themen sind nicht Teil dieser Arbeit:

- GMP-Validierung und die regulatorischen Anforderungen der pharmazeutischen Industrie. Das Prüfband ist ein Demonstrator, keine Produktionsmaschine.
- Industriekamera und Industriebeleuchtung.
- Mehrere Tablettentypen oder Prüflinien.
- Automatische Geräteregistrierung und ähnliche Funktionen für viele Geräte.
- Ein sauberer, geregelter Betrieb des Tenants und des Abonnements ist nicht Teil dieser Arbeit. Berechtigungen wurden nicht systematisch nach dem Prinzip der geringsten Berechtigung (Least Privilege) gestaltet, und es gibt keine Just-in-Time-Verwaltung (z. B. Privileged Identity Management), keine festgelegten Richtlinien und keine zentrale Protokollierung. Wo es einfach war, wurde das Prinzip dennoch angewendet, zum Beispiel mit einem separaten Fabric-Konto, das nur eine beschränkte Rolle besitzt.
- Ein Langzeittest (Haltbarkeit) des Motors und des Förderbands.

2.5. Annahmen

- Beim Transfer Learning reichen einige hundert eigene Bilder für eine zuverlässige Erkennung aus.
- Die Komponenten sind leicht erhältlich (Lieferung innerhalb weniger Wochen und bei verschiedenen Lieferanten verfügbar).
- Die 3D-gedruckten Teile können beim Arbeitgeber des Diplomanden hergestellt werden.
- Am Präsentationsort ist Mobilfunkempfang vorhanden.

2.6. Akzeptanzkriterien

In Abschnitt 13.2 wird beurteilt, ob die Erfolgskriterien EK1 bis EK5 (2.1), die Zielwerte Z1 bis Z5 und die Anforderungen (2.2) erfüllt wurden.

2.7. Zu erbringende Leistungen

Leistung	Form	Fälligkeit
Diplomarbeit einschliesslich Anhang	PDF (DA_2026_Kuhn_Christian_Pruefband.pdf)	28.10.2026
Qualifikationsprofil einschliesslich Lebenslauf	PDF (QP_2026_Kuhn_Christian_Pruefband.pdf)	28.10.2026
Microsite	TEKO Webapp	28.10.2026
Prüfband und Präsentation	15 Minuten Vorstellung und 15 Minuten Fragen beantworten	19.11.2026

3. Terminplan SOLL/IST und Projektrisiken

3.1. Vorgehensmodell

Das Projekt nutzt das Phasenmodell IPERKA (Informieren, Planen, Entscheiden, Realisieren, Kontrollieren, Auswerten). Jede Phase ist Kapiteln dieser Arbeit zugeordnet. Die Realisierung selbst ist in Inkremente unterteilt, und jedes Inkrement endet mit einem getesteten, funktionierenden Ergebnis.



Abbildung 2: IPERKA erklärt (https://www.ict-berufsbildung-bern.ch/resources/Iperka_OdA_200617.pdf)

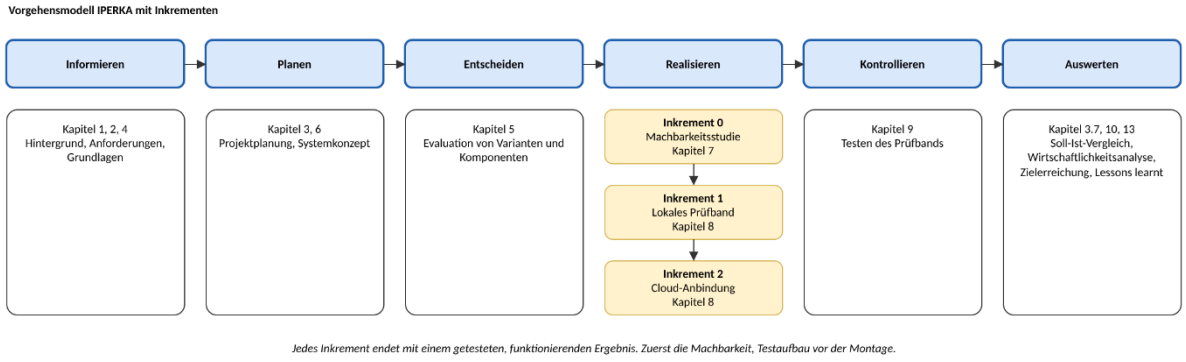


Abbildung 3: IPERKA-Phasen mit Kapiteln (mit KI generierte Darstellung, draw.io)

Phase	Inhalt	Kapitel
Informieren	Hintergrund, Anforderungen, Grundlagen	1, 2, 4
Planen	Projektplanung, Systemkonzept	3, 6
Entscheiden	Evaluation von Varianten und Komponenten	5

Realisieren	Inkrement 0: Machbarkeitsstudie; Inkrement 1: lokales Prüfband; Inkrement 2: Cloud-Anbindung	7, 8
Kontrollieren	Testen des Prüfbands	9
Auswerten	Wirtschaftlichkeitsanalyse, Soll-Ist-Vergleich, Zielerreichung, gewonnene Erkenntnisse	3.7, 10, 13

Zwei Grundsätze leiten die Realisierung:

1. **Zuerst die Machbarkeit:** Bilderkennung und Cloud-Anbindung, die beiden grössten technischen Unbekannten, werden in einer Machbarkeitsstudie getestet, bevor Hardware montiert wird.
2. **Testaufbau vor der Montage:** Alle elektronischen Komponenten werden zunächst lose angeschlossen und getestet, da sich Fehler in der Hardware und der Verkabelung auf einer offenen Arbeitsfläche viel leichter finden und korrigieren lassen.

3.2. Projektorganisation

Rolle	Verantwortung
Diplomand, Projektleiter	Planung, Umsetzung, Dokumentation, Präsentation
Diplomcoach	Begleitung, zwei Review-Gespräche, Bewertung
Diplomexperte	Bewertung der Diplomarbeit und der Präsentation

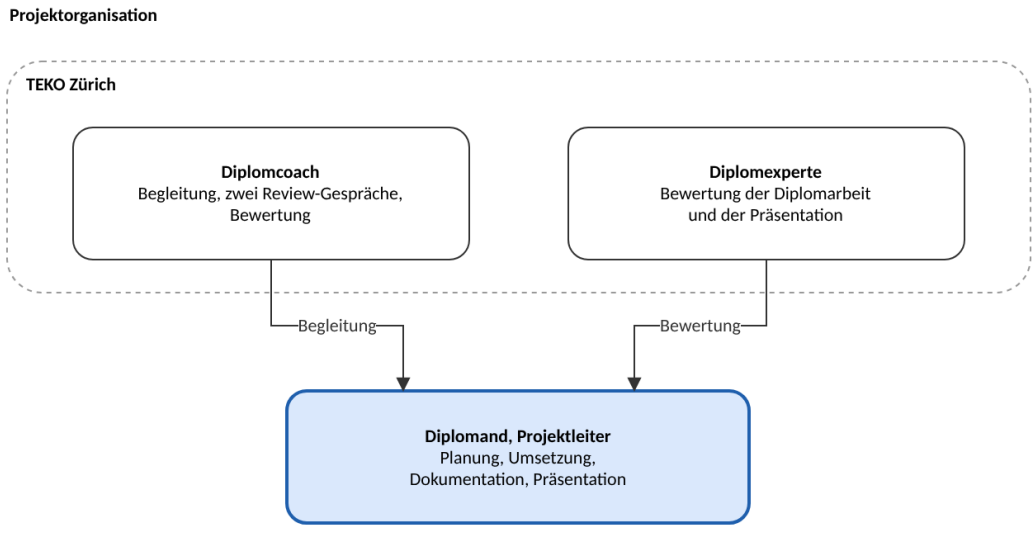


Abbildung 4: Organigramm (mit KI generierte Darstellung, draw.io)

3.3. Projektstrukturplan

Der Projektstrukturplan gruppiert die zwölf Arbeitspakete nach IPERKA-Phasen.

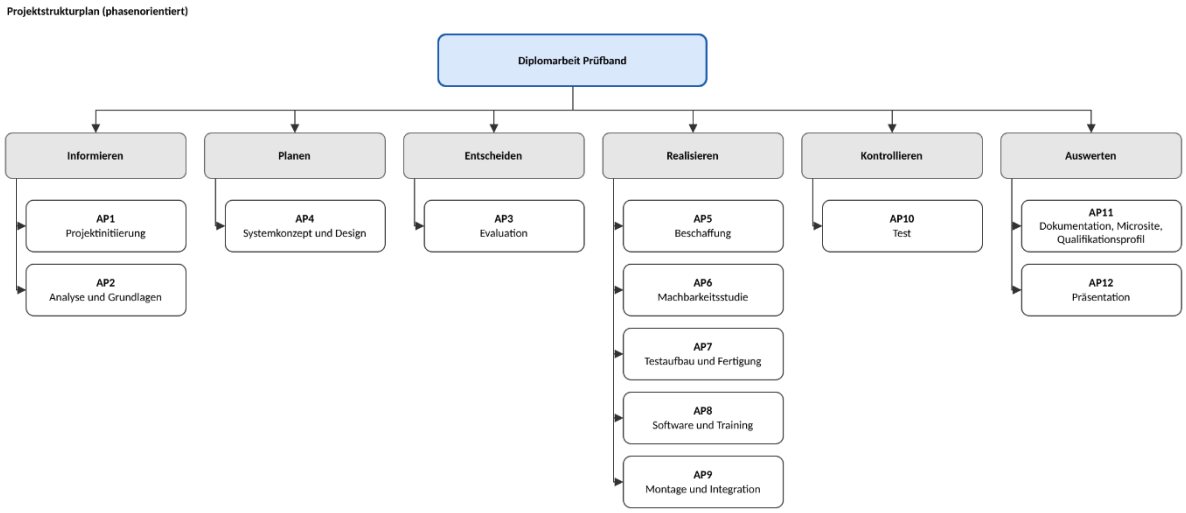


Abbildung 5: Projektstrukturplan (mit KI generierte Darstellung, draw.io)

Nr.	Arbeitspaket	Ergebnis
AP1	Projektinitiierung	Genehmigte Themeneingabe
AP2	Analyse und Grundlagen	Anforderungen, Kapitel «Grundlagen»
AP3	Evaluation	Entscheidungen zu Hardware, Software und Cloud
AP4	Systemkonzept und Design	Architektur, Netzwerk, elektrisches und mechanisches Konzept, Druckdateien
AP5	Beschaffung	Komponenten verfügbar
AP6	Machbarkeitsstudie	Machbarkeit von Bildverarbeitung und Cloud nachgewiesen
AP7	Testaufbau und Fertigung	Getestete Komponenten, gedruckte Teile
AP8	Software und Training	Software des Raspberry Pi 5, Prüfsoftware, trainiertes KI-Modell
AP9	Montage und Integration	Funktionierendes Prüfband
AP10	Test	Testprotokolle
AP11	Dokumentation, Microsite, Qualifikationsprofil	Einreichung
AP12	Präsentation	Präsentation und Demonstration

3.4. Terminplan SOLL und Meilensteine

Der Terminplan ist in ProjectLibre geplant und in Kalenderwochen gegliedert. Die Ferien vom 15. bis 26. Oktober 2026 (Kalenderwochen 42 bis 44) wurden von Anfang an als Puffer reserviert. Läuft das Projekt nach Plan, bleiben sie frei und bei Verzögerungen bieten sie zusätzliche Arbeitszeit.

Der Terminplan zeigt Termine, keinen Aufwand. Die Dauer einer Aufgabe ist der Zeitraum in Arbeitstagen, in dem ein Ergebnis zu erwarten ist. Sie gibt nicht an, wie viele Stunden benötigt werden. An einem Arbeits- oder Schultag bleiben etwa 1,5 bis 2 Stunden für das Projekt übrig. Während des Puffers, der in schul- und arbeitsfreier Zeit liegt, standen mehr als 8 Stunden pro Tag zur Verfügung. Eine Aufgabe von 10 Tagen erstreckt sich daher über zwei Kalenderwochen, und ein Puffertag enthält etwa das Vier- bis Fünffache des Arbeitsaufwands eines normalen Tages.

Die Themeneingabe wurde am 08.06.2026 eingereicht und am 16.06.2026 genehmigt.

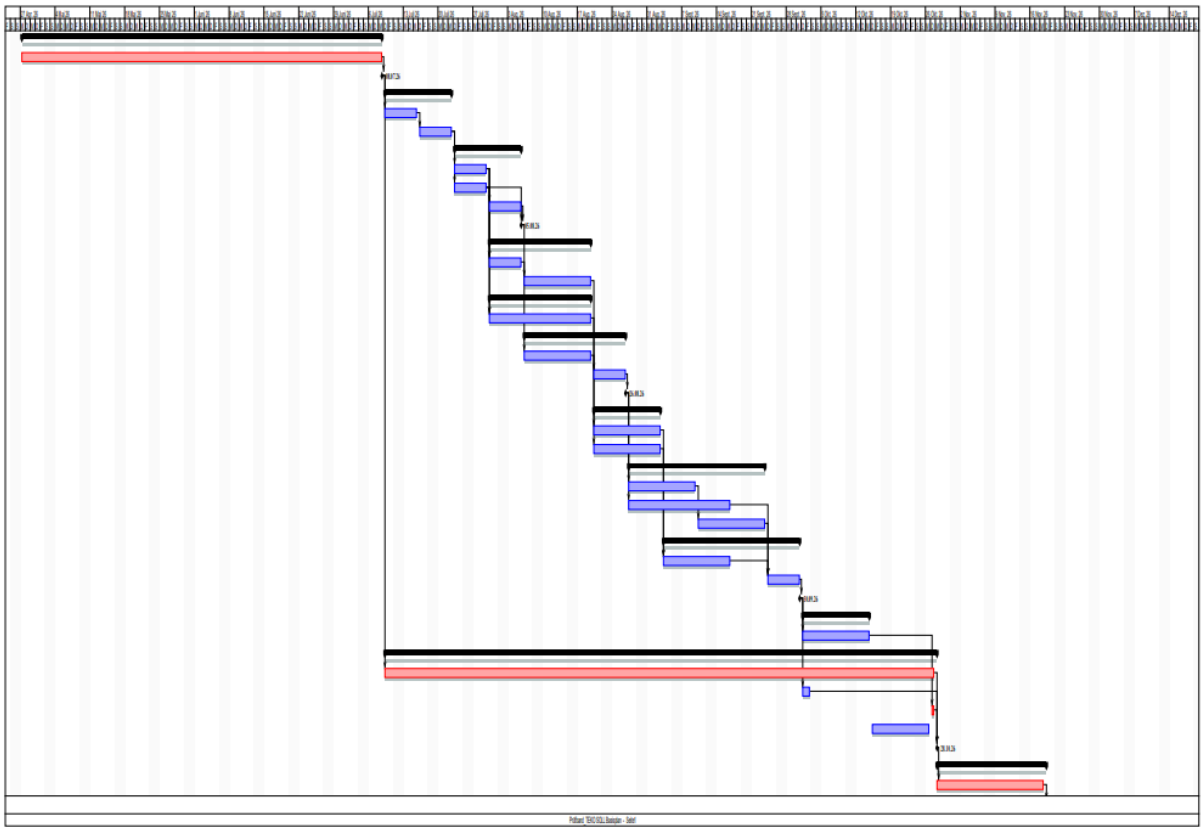


Abbildung 6: Terminplan SOLL (eigene Aufnahme)

Meilenstein	Solldatum
M1 Themeneingabe genehmigt	08.07.2026
M2 Evaluation abgeschlossen	05.08.2026
M3 Machbarkeit nachgewiesen	26.08.2026
M4 Prüfband funktionsfähig	30.09.2026
M5 Einreichung der Diplomarbeit, des Qualifikationsprofils und der Microsite	28.10.2026
M6 Präsentation	19.11.2026

3.5. Projektsteuerung

Der Terminplan wurde zu mehreren Zeitpunkten überprüft: Der verbleibende Arbeitsaufwand wurde mit der verbleibenden Zeit verglichen, um zu prüfen, ob der Plan noch aufging. Der Terminplan wurde zudem mit dem Diplomcoach angeschaut, und es wurden gemeinsam Massnahmen besprochen, beispielsweise eine Prognose eine Woche vor der Einreichung mit einem Plan A und einem Plan B (Kapitel «13.8. Gespräche mit dem Diplomcoach»). Die eingetretenen Ereignisse werden in Abschnitt 3.7 beschrieben.

3.6. Projektrisiken

Jedes Risiko wird hinsichtlich Wahrscheinlichkeit (P) und Auswirkung (I) auf einer Skala von 1 (niedrig) bis 3 (hoch) bewertet. Das Produkt P × I bestimmt die Priorität. Die Einstufung der einzelnen Risiken zeigt die Risikomatrix.

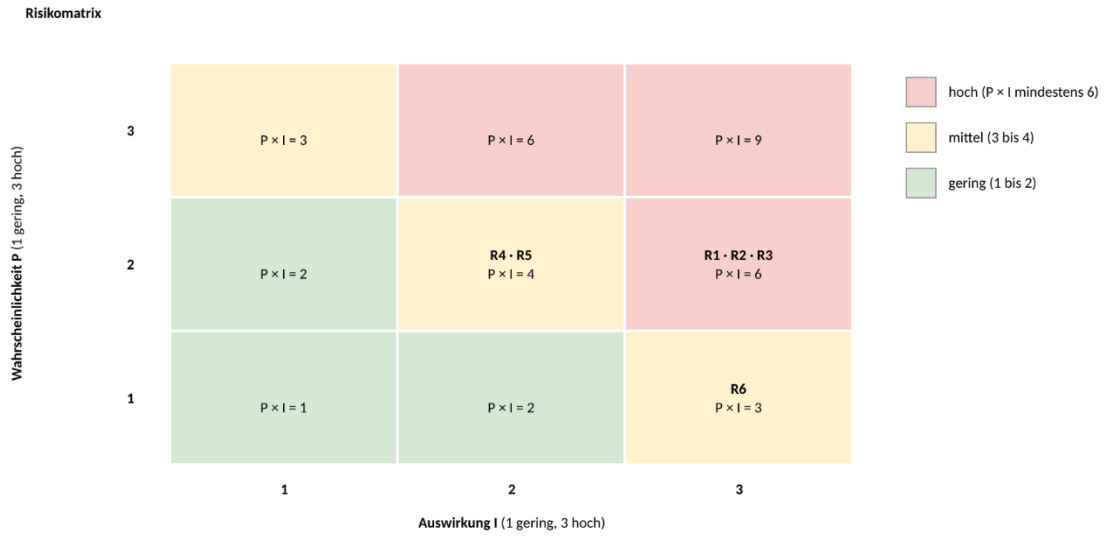


Abbildung 7: Risikomatrix (mit KI generierte Darstellung, draw.io)

Nr.	Risiko	Massnahme
R1	Kapazitätsengpass durch Arbeitsbelastung oder Abwesenheit	Ferien als Puffer eingeplant (Kalenderwochen 42 bis 44), priorisieren.
R2	Technische Machbarkeit nicht gegeben: Kamera und Modell erkennen nicht zuverlässig	Machbarkeitsstudie vor der Realisierung, ausfallsichere Prüflogik.
R3	Abhängigkeit von Dritten	Frühzeitig Komponenten bestellen, Ersatzteile einplanen.
R4	Das Erlernen neuer Technologien (KI, IoT) dauert länger als geplant	Einfachere und vortrainierte Modelle verwenden, Machbarkeitsstudie als Lernschritt.
R5	Integrationsprobleme oder Beschädigung der Hardware	Klare Schnittstellen über MQTT, Testaufbau vor der Endmontage, Hardware auf Reserve bestellen.
R6	Datenverlust (Code, Bilder, Dokumentation)	Regelmässige Backups auf externe Speichermedien und in die Cloud

Im Verlauf des Projekts traten die Risiken R1, R3 und R5 ein und die Massnahmen haben funktioniert.

3.7. Terminplan IST und Soll-Ist-Vergleich

Die Projektinitiierung wurde etwa einen Monat früher als geplant abgeschlossen. Bis zum Nachweis der Machbarkeit führten ungeplante Neubewertungen und drei Wochen ohne jegliche Arbeit an der Diplomarbeit dazu, dass aus diesem Vorsprung eine Verzögerung resultierte. Parallele Arbeiten und vor allem der geplante Puffer ermöglichten es, den Einreichungstermin am 28.10.2026 einzuhalten.

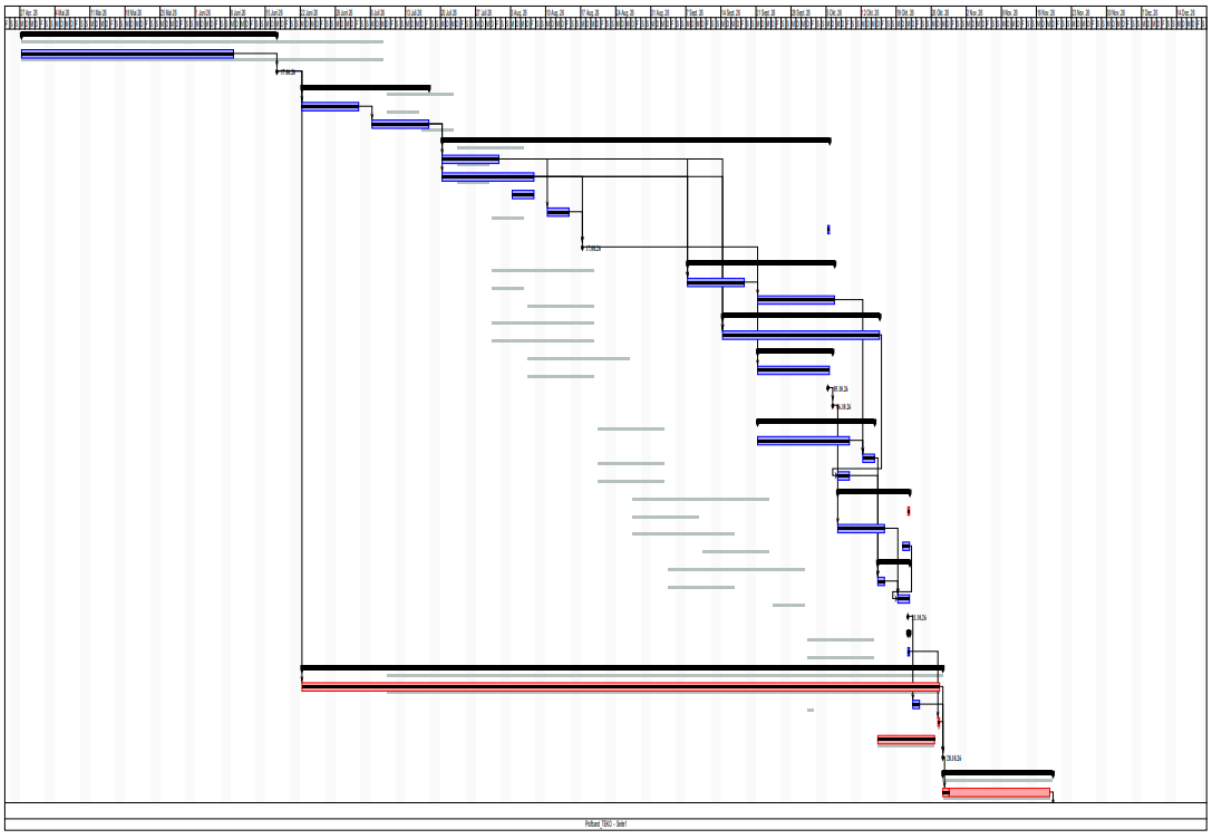


Abbildung 8: Terminplan SOLL- / IST-Vergleich (eigene Aufnahme aus ProjectLibre)

Meilenstein	Soll	Ist	Abweichung
M1 Themeneingabe genehmigt	08.07.2026	16.06.2026	3 Wochen früher
M2 Evaluation abgeschlossen	05.08.2026	14.08.2026	1,5 Wochen später
M3 Machbarkeit nachgewiesen	26.08.2026	05.10.2026	6 Wochen später
M4 Prüfband funktionsfähig	30.09.2026	21.10.2026	3 Wochen später
M5 Einreichung	28.10.2026	28.10.2026	Zeitgerecht
M6 Präsentation	19.11.2026	19.11.2026	-

3.8. Ereignisse

Kalenderwoche	Ereignis	Auswirkung	Massnahme
32	Der Preis des Jetson Orin Nano steigt von ca. CHF 339 auf ca. CHF 500.	Die gewählte Edge-Hardware ist nicht mehr sinnvoll.	Neue Nutzwertanalyse (5.2.2).
34 bis 35	Unerwartet hohe Arbeitsbelastung im Beruf	Zwei Wochen ohne Arbeit an der Diplomarbeit	Terminplan verschiebt sich in den Puffer.
36	Der Diplomand ist eine Woche lang krank.	Eine weitere Woche ohne Arbeit	Terminplan verschiebt sich in den Puffer.
39	Microsoft stellt Azure IoT Central für neue Anwendungen ein.	Die gewählte Cloud-Plattform ist nicht verfügbar.	Neubewertung, Wechsel zu Azure IoT Hub (5.8.2).
39	Die Klärung, ob die Teile beim Arbeitgeber gedruckt werden können, dauert länger als geplant.	Der mechanische Aufbau verzögert sich.	Ausweichlösungen werden angefragt: externer Druckdienst oder ein Drucker von Bekannten.
40	Die Lieferung des SICK-Sensors verzögert sich weiter.	Der ausgewählte Sensor fehlt.	Überbrückung mit dem TCRT5000; Austausch ohne Codeänderung.
43	Fabric erfordert ein Arbeitskonto; die Testversion kann zunächst nicht gestartet werden.	Durch das Troubleshooting um ½ Tag verzögert.	Separater Fabric-Benutzer, vorübergehende F2-Kapazität, dann Testversion (8.10).

3.9. Werkzeuge

Zweck	Werkzeug
Terminplan (Gantt)	ProjectLibre
Diagramme (Architektur, Netzwerk, Abläufe, Projektstrukturplan, Risikomatrix, Schaltplan)	draw.io
Entwicklung	Visual Studio Code, Python
Annotation von Trainingsbildern (Machbarkeitsstudie)	Label Studio
Generieren der CAD-Dateien der 3D Druckteile	Claude
Unterstützung bei der Umsetzung (Code-Entwürfe, Troubleshooting, Grafiken)	Claude (KI, siehe KI-Promptverzeichnis)
Umformulierung und Korrektur der Dokumentation	DeepL Write
Cloud	Microsoft Azure, Microsoft Fabric

Lösung der Aufgabe

Die Kapitel 4 bis 10 beschreiben die Lösung der Aufgabe: Grundlagen, Evaluation der Komponenten, Systemkonzept, Machbarkeitsstudie, Umsetzung, Tests und Wirtschaftlichkeitsanalyse.

4. Analyse und Grundlagen

4.1. Sichtprüfung in der pharmazeutischen Industrie

In der pharmazeutischen Industrie finden Computergestütze Sichtprüfungen verschiedene Einsatzgebiete: Ist eine Tablette vorhanden, ist sie unversehrt, hat sie die richtige Form? Fehlerhafte Produkte werden automatisch aussortiert. Es muss zwischen zwei Arten von Fehlern unterschieden werden:

Begriff	Bedeutung	Folge
Schlupf	Eine fehlerhafte Einheit wird als gut bewertet.	Der Fehler gelangt zum Patienten. Kritische Auswirkung möglich.
Pseudoausschuss	Eine gute Einheit wird als fehlerhaft bewertet.	Materialverlust, kein Risiko für den Patienten.

Die Industrie akzeptiert daher bewusst mehr Pseudoausschuss, um Schlupf auszuschliessen. Diese Gewichtung bestimmt die Prüflogik des Prüfbands.

4.2. Computer Vision: Klassifizierung und Objekterkennung

Computer Vision umfasst die Methoden, mit denen Computer Bilder interpretieren können. Die klassische Bildverarbeitung wendet feste Regeln wie Helligkeitsschwellen oder Kantenfilter an. Sie ist schnell und nachvollziehbar, reagiert jedoch empfindlich auf Änderungen und ist aufwendig in der Feinabstimmung.

Maschinelles Lernen hingegen lernt die relevanten Merkmale aus Beispielbildern. Ein Convolutional Neural Network (CNN) erkennt in seinen ersten Schichten einfache Muster wie Kanten und kombiniert diese in tieferen Schichten zu Formen und Objekten. Zwei Aufgabentypen sind üblich und wurden in dieser Arbeit verwendet:

Aufgabe	Frage	Ausgabe	Verwendung in dieser Arbeit
Klassifizierung	Was zeigt das gesamte Bild?	Eine Klasse und ein Konfidenzwert[1]	Prüfband: eine Tablette pro Bild
Objekterkennung	Welche Objekte befinden sich wo?	Begrenzungsrahmen (Boxen), jeweils mit Klasse und Konfidenz	Machbarkeitsstudie: zehn Mulden pro Blister

[1] Ultralytics: Image Classification with Ultralytics YOLO, https://docs.ultralytics.com/tasks/classify



Abbildung 9: Klassifizierung vs Objekterkennung (https://labelyourdata.com/articles/object-detection-vs-image-classification)

Die Wahl hängt von der Fragestellung ab: Bei einem Blister ist die Objekterkennung erforderlich, da das System wissen muss, welche der zehn Mulden leer ist. Auf dem Prüfband befindet sich genau eine Tablette im Bild, sodass eine Klassifizierung ausreicht (7.4).

4.3. YOLO und Transfer Learning

YOLO (You Only Look Once) ist eine Modellfamilie, die das gesamte Bild in einem einzigen Durchlauf verarbeitet, wodurch sie schnell genug für den Echtzeiteinsatz ist [2]. Die Ultralytics-Implementierung bietet dieselbe Modellfamilie für Erkennung, Segmentierung und Klassifizierung an.

Beim Transfer Learning wird von einem Modell ausgegangen, das bereits Kanten, Formen und Objekte aus einem grossen allgemeinen Datensatz gelernt hat. Einige Dutzend bis einige hundert eigene Bilder passen es dann an die neue Aufgabe an. Nur die letzten Schichten müssen neu gelernt werden; der Rest wird wiederverwendet.

Begriff	**Bedeutung**
Precision	Anteil der gemeldeten Objekte, die korrekt sind (Erkennung)
Recall	Anteil der vorhandenen Objekte, die gefunden werden (Erkennung)
mAP50	Mean Average Precision, wenn sich die erkannte Box zu mindestens 50 % mit der tatsächlichen Box überschneidet (Erkennung)
mAP50-95	Wie mAP50, gemittelt über Überschneidungen von 50 bis 95 %; misst, wie genau Objekte lokalisiert werden (Erkennung)
Top-1 Accuracy	Anteil der Bilder, deren wahrscheinlichste Klasse korrekt ist (Klassifizierung)[3]
Konfidenz	Sicherheit des Modells für ein einzelnes Ergebnis (0 bis 1)
Augmentierung	Künstliche Veränderungen von Trainingsbildern (Spiegelung, Helligkeit, Kontrast), um das Modell robuster zu machen
Batch Size	Anzahl der Bilder pro Lernschritt
Epoche	Ein Durchlauf über alle Trainingsbilder

Moderne GPUs trainieren mit gemischter Genauigkeit (Mixed Precision): Die meisten Berechnungen verwenden 16-Bit-Zahlen, was den Speicherbedarf halbiert und das Training beschleunigt[4]. Ultralytics nutzt diesen Modus automatisch («AMP» in der Trainingsausgabe).

[2] Ultralytics: YOLO11-Dokumentation (Training, Validierung, Lizenz), https://docs.ultralytics.com

[3] Ultralytics: Image Classification with Ultralytics YOLO, https://docs.ultralytics.com/tasks/classify

[4] NVIDIA: Train with mixed precision, https://docs.nvidia.com/deeplearning/performance/mixed-precision-training

4.4. Edge-Computing

Edge-Computing verarbeitet Daten dort, wo sie entstehen. Gründe dafür sind kurze Reaktionszeiten, weniger über das Internet gesendete Daten und der fortgesetzte Betrieb ohne Internetverbindung.

Geräteklasse	Beispiel	Stärke	Einschränkung
Mikrocontroller	ESP32, STM32	Sehr günstig, Echtzeit, geringer Stromverbrauch	Kaum KI-Leistung
Einplatinencomputer	Raspberry Pi 5	Günstig, GPIO, Linux, Netzwerk	Geringe KI-Leistung ohne Beschleuniger
Edge-KI-Modul	NVIDIA Jetson Orin Nano	KI in kompakter, industrieller Bauweise	Höherer Preis, eingeschränkte Aufrüstbarkeit
Workstation oder Edge-Server	Desktop-PC mit GPU	Hohe KI-Leistung, aufrüstbar	Grösse, Stromverbrauch

Welche Geräteklassen das Prüfband verwendet, wird in Kapitel 5 evaluiert.

4.5. Warum ein Desktop-PC mit einer RTX 3060 Ti für KI geeignet ist

Dieser Abschnitt nimmt ein Ergebnis der Evaluation vorweg: Dort setzte sich der vorhandene Desktop-PC als Edge-PC gegenüber einem dedizierten KI-Modul durch (Details in Abschnitt 5.2). Die folgenden Absätze erläutern, warum ein solcher PC überhaupt für KI geeignet ist.

Der Edge-PC ist ein Desktop-Computer, der für CAD-Arbeiten angeschafft wurde: ein Dell XPS 8940 mit einem Intel i7-11700, 16 GB RAM und einer NVIDIA GeForce RTX 3060 Ti. Solche Rechner sind für hohe Grafiklasten ausgelegt und bieten bereits das, was KI benötigt: eine leistungsstarke Grafikkarte, eine ausreichende Stromversorgung und eine gute Kühlung.

Ein neuronales Netz besteht hauptsächlich aus Matrixmultiplikationen. Eine CPU berechnet diese mit wenigen leistungsstarken Kernen, eine GPU parallel mit Tausenden kleiner Kerne. Die RTX 3060 Ti verfügt über 4'864 CUDA-Kerne, auf Matrixberechnungen spezialisierte Tensor-Kerne und 8 GB Grafikspeicher[5] [6]. PyTorch und Ultralytics nutzen die CUDA-Schnittstelle von NVIDIA direkt, sodass eine Standardinstallation ohne zusätzlichen Aufwand auf der GPU läuft: In diesem Projekt dauerte das Training des Tablettenmodells 3,5 Minuten und die Klassifizierung eines Bildes 5,7 ms.

Die 8 GB Grafikspeicher sind die grösste Einschränkung: Sie reichen für YOLO-Modelle aus, nicht jedoch für grosse Sprachmodelle. Im Gegensatz zu einem Raspberry Pi lässt sich ein Desktop-PC aufrüsten, falls zukünftige Modelle mehr Speicher benötigen.

[5] NVIDIA: GeForce RTX 3060 family, specifications, https://www.nvidia.com/en-ph/geforce/graphics-cards/30-series/rtx-3060-3060ti/

[6] NVIDIA: Tensor Cores, https://www.nvidia.com/en-eu/data-center/tensorcore/

4.6. Der Hardware-Markt unter dem Einfluss der KI

Der KI-Boom verändert die Preise für Computerhardware, und dieser Effekt hat sich von Grafikkarten auf Arbeitsspeicher und Speichermedien ausgeweitet. Dies ist für die Evaluation (Kapitel 5) und die Wirtschaftlichkeitsanalyse (Kapitel 10) von Bedeutung.

4.6.1. Grafikkarten

KI-Rechenzentren benötigen denselben Chip-Typ wie Gaming-Grafikkarten und zahlen dafür mehr[7]. Im Februar 2026 warnte der CEO von NVIDIA, dass das Angebot an Gaming-GPUs für einige Quartale «sehr knapp» sein werde[8]. Im August 2026 wurden aktuelle Grafikkarten immer noch deutlich über ihren Einführungspreisen verkauft, das Topmodell sogar zum mehr als doppelten Einführungspreis[9].

4.6.2. Arbeitsspeicher und Speichermedien

Ein Bericht geht davon aus, dass im Jahr 2026 bis zu 70 % der gesamten Speicherproduktion in Rechenzentren fliessen werden[10]. Die DRAM-Preise stiegen in einigen Fällen innerhalb eines Jahres um fast 700 %, und auch SSDs wurden teurer [11]. Ein 32-GB-DDR5-Kit, das Mitte 2025 etwa USD 90 bis 119 kostete, kostete ein Jahr später etwa USD 379 bis 459[12].

4.6.3. Die Auswirkungen auf Verbraucher und Unternehmen

In den USA stiegen die Preise für Computer zum ersten Mal seit den frühen 1980er-Jahren[13]. In der Schweiz rechneten Einzelhändler für 2026 mit einem Anstieg der Notebook-Preise um 10 bis 20 %[14], und ein IT-Unternehmen in Lausanne berichtete, dass ein Computer, der ein Jahr zuvor noch CHF 660 gekostet hatte, nun CHF 1'500 kostete[15].

---

[7] Stokel-Walker, C.: How the AI revolution is triggering a hardware arms race and pushing up prices. Tom's Hardware, 22.08.2025, https://www.tomshardware.com/tech-industry/semiconductors/how-the-ai-revolution-is-triggering-a-hardware-arms-race-and-pushing-up-prices

[8] Shilov, A.: Nvidia warns of constrained supply of gaming GPUs. Tom's Hardware, 26.02.2026, https://www.tomshardware.com/pc-components/gpus/nvidia-warns-of-constrained-supply-of-gaming-gpus-we-do-believe-for-a-couple-of-quarters-it-is-going-to-be-very-tight

[9] Bendle, S.: GPU prices keep surging in 2026. Tom's Hardware, 03.08.2026, https://www.tomshardware.com/pc-components/gpus/gpu-prices-for-current-gen-nvidia-and-amd-price-increases-why-have-the-prices-not-dropped-and-can-you-still-buy-a-cheap-gpu

[10] Ferreira, B.: Data centers will consume 70 percent of memory chips made in 2026. Tom's Hardware, 18.01.2026, https://www.tomshardware.com/pc-components/ram/data-centers-will-consume-70-percent-of-memory-chips-made-in-2026-supply-shortfall-will-cause-the-chip-shortage-to-spread-to-other-segments

[11] Bloomberg: Why the AI Boom is Making Everything More Expensive (interaktive Grafik, Datenstand 01.03.2026), https://www.bloomberg.com/graphics/2026-ai-boom-memory-chip-shortage/

[12] Warwick, S.: One year into the AI-induced RAM apocalypse. Tom's Hardware, 28.07.2026, https://www.tomshardware.com/pc-components/ram/one-year-into-the-ai-induced-ram-apocalypse-how-much-does-memory-actually-cost-and-is-there-hope-for-a-more-affordable-future

[13] Cerullo, M.: Buying a computer? AI is making it more expensive. CBS News, 30.04.2026, https://www.cbsnews.com/news/computer-price-memory-chip-shortage-artificial-intelligence/

[14] Lago, P.: Wegen KI: Handys und Laptops werden bis zu 20 Prozent teurer. SRF News, 04.04.2026, https://www.srf.ch/news/wirtschaft/chip-krise-wegen-ki-handys-und-laptops-werden-bis-zu-20-prozent-teurer

[15] Miquel, T. (RTS): KI-Boom lässt Preise für Computer in der Schweiz steigen. SRF News, 07.10.2026, https://www.srf.ch/news/dialog/ki-boom-it-kosten-von-staedten-und-kantonen-steigen-stark

4.6.4. Auswirkungen auf dieses Projekt

Der Preisanstieg des Jetson Orin Nano (Abschnitte 3.7 und 5.2.2) passt in dieses Muster. Die Verwendung eines vorhandenen Desktop-PCs anstelle des Kaufs neuer KI-Hardware war daher auch wirtschaftlich die richtige Entscheidung. Bei einem Serienprodukt stellen Speicherpreise ein Kostenrisiko dar (Kapitel 10).

4.7. Schrittmotoren und Motortreiber

Ein Schrittmotor dreht sich in kleinen, festgelegten Schritten. Sein Treiber regelt den Strom im Motor und kann jeden Schritt in kleinere Schritte unterteilen (Mikroschrittbetrieb), wodurch die Bewegung gleichmässiger wird.

Klassische Treiber wie der A4988 regeln den Strom, indem sie die Spulenspannung mit hoher Frequenz ein- und ausschalten[16]; die Strombegrenzung wird über ein Potentiometer eingestellt[17]. Der Strom weist dann kleine Welligkeiten auf, und bei niedrigen Drehzahlen neigt der Motor zum Brummen und Vibrieren. Der TMC2209 arbeitet bei niedrigen Drehzahlen anders: Sein StealthChop2-Modus erzeugt einen nahezu sinusförmigen Strom und reduziert das Motorgeräusch bei niedrigen Drehzahlen fast bis zur Geräuschlosigkeit. Zudem interpoliert er die 1/8-Schritte des Prüfbands intern auf 256 Mikroschritte pro Vollschritt[18].

Ein Förderband läuft langsam und gleichmässig, also genau dort, wo der TMC2209 seine Vorteile hat. Ein A4988 wurde lediglich als Ersatzteil bestellt.

[16] Allegro MicroSystems: Datenblatt A4988

[17] Pololu: A4988 Stepper Motor Driver Carrier, Produktdokumentation

[18] Analog Devices: Datenblatt TMC2209, Rev. 1.08

4.8. IoT, MQTT und Cloud

Das Internet der Dinge (IoT) verbindet Geräte mit Diensten in einem Netzwerk. Das am weitesten verbreitete Protokoll ist MQTT, das dem Publish/Subscribe-Prinzip folgt: Ein Gerät veröffentlicht eine Nachricht zu einem Thema (Topic) bei einem Broker, und Empfänger abonnieren dieses Thema[19]. Sender und Empfänger kommunizieren niemals direkt miteinander, sodass die Teilsysteme entkoppelt bleiben. MQTT bietet drei Zustellungsgarantien (Quality of Service 0, 1 und 2) und läuft verschlüsselt über TLS auf Port 8883.

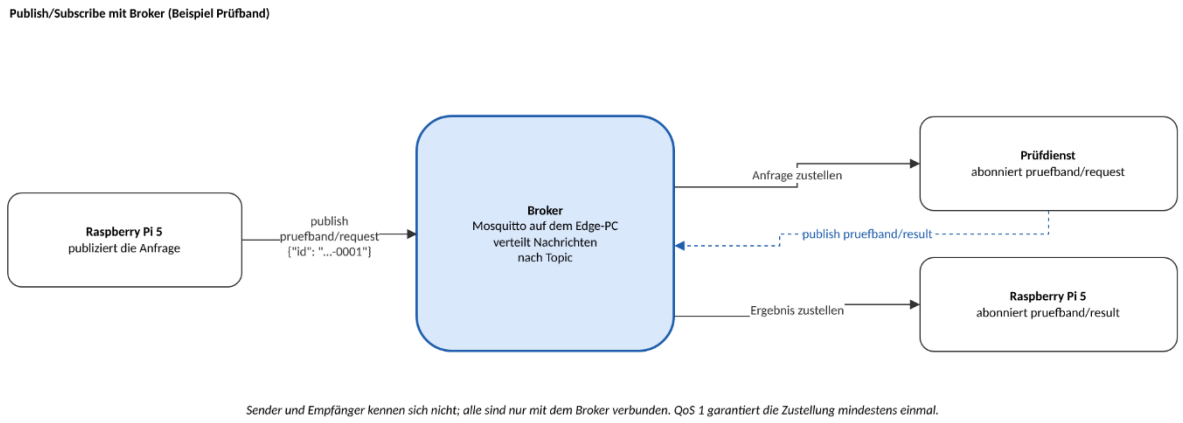


Abbildung 10: Publish-Subscribe (mit KI generierte Darstellung, draw.io)

Cloud-Anbieter bieten IoT-Plattformen an, die Geräte mit eigener Identität verwalten, deren Nachrichten (Telemetrie) empfangen und diese zur Speicherung und Analyse an andere Dienste weiterleiten. Microsoft empfiehlt Azure IoT Hub für die Geräteanbindung und Microsoft Fabric Real-Time Intelligence für Dashboards und Analysen[20]. In Fabric empfängt ein Eventstream die Nachrichten, ein Eventhouse speichert sie in einer Datenbank, die mit der Kusto Query Language (KQL) abgefragt wird, und ein Real-Time Dashboard zeigt sie an[21].

[19] OASIS: MQTT Version 5.0, OASIS Standard, https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[20] Microsoft Learn: Azure IoT Central is evolving, https://learn.microsoft.com/en-us/azure/iot-central/core/iot-central-evolution

[21] Microsoft Learn: Add Azure IoT Hub source to an eventstream, https://learn.microsoft.com/en-in/fabric/real-time-intelligence/event-streams/add-source-azure-iot-hub

4.9. Systemkontext und Anwendungsfälle

Das Prüfband befindet sich zwischen dem Bediener, dem zu prüfenden Objekt und der Cloud-Plattform.

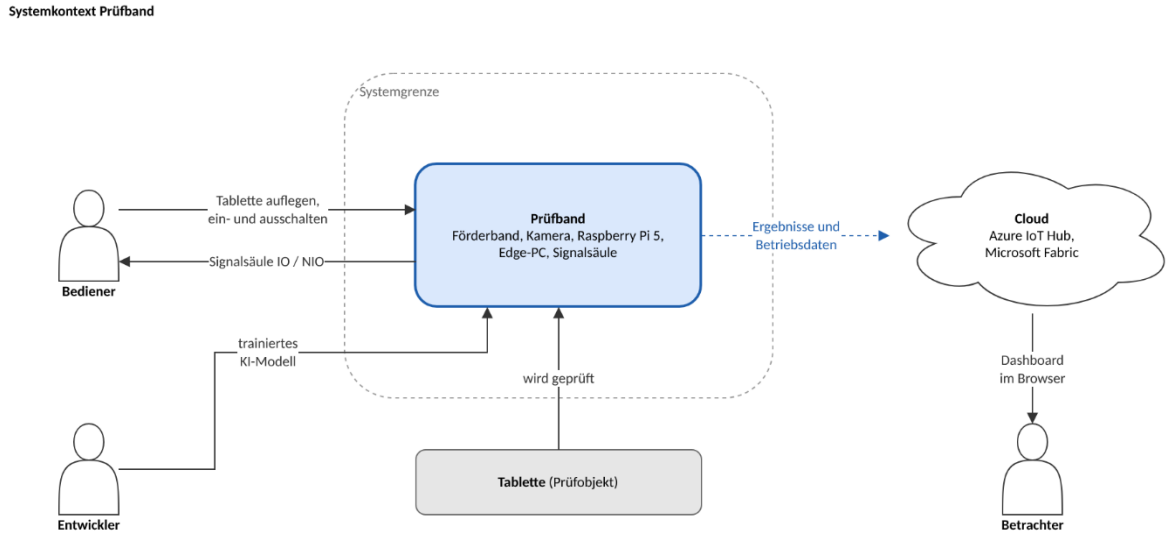


Abbildung 11: Systemkontext (mit KI generierte Darstellung, draw.io)

Anwendungsfall	Akteur	Beschreibung
UC1 Tablette prüfen	Bediener	Eine Tablette auf das Förderband legen; das Prüfband transportiert und prüft sie und zeigt das Ergebnis an.
UC2 Ergebnisse auswerten	Betrachter	Die Ergebnisse im Cloud-Dashboard einsehen (Anzahl pro Klasse, Ausschuss, Liste der letzten Prüfungen).
UC3 Prüfband in Betrieb nehmen	Bediener	Einschalten; den Raspberry Pi 5, die Prüfsoftware und die Cloud-Anbindung starten.
UC4 KI-Modell trainieren	Entwickler	Bilder pro Klasse auf dem Prüfband sammeln, das Modell trainieren und in Betrieb nehmen.

5. Evaluation

5.1. Methode

Die Hauptkomponenten werden mittels einer Nutzwertanalyse ausgewählt. Jede Variante erhält pro Kriterium eine Punktzahl von 1 (schlecht) bis 5 (sehr gut); die Gewichte summieren sich auf 100 %. Der Nutzwert ist die Summe aus Punktzahl mal Gewicht. Die Kriterien leiten sich aus den Zielen und Anforderungen in Kapitel 2 ab. Bei Komponenten mit nur wenigen sinnvollen Varianten basiert die Entscheidung auf einem begründeten Vergleich.

5.2. Edge-Computer für die KI-Auswertung

5.2.1. Erste Evaluation

Im Mittelpunkt der ersten Runde standen kompakte Edge-Geräte, die in das Prüfband passen. Der NVIDIA Jetson Orin Nano war für etwa CHF 339 erhältlich. Da das Hardware-Budget eines privaten Projekts begrenzt ist, wird der Anschaffungspreis mit 25 % gewichtet.

Kriterium	Gewichtung	Jetson Orin Nano	Raspberry Pi 5 mit AI HAT	Raspberry Pi AI-Kamera
KI-Leistung, Echtzeit	30 %	5	3	2
Anschaffungspreis	25 %	3	4	5
Industrielle Relevanz, Integration	20 %	5	4	4
Lernkurve, Tools, Modellauswahl	10 %	3	3	3
Verfügbarkeit, Lieferzeit	15 %	4	4	4
Nutzwert	100 %	4,15	3,60	3,55

Entscheidung: NVIDIA Jetson Orin Nano.

5.2.2. Neubewertung nach einer Preiserhöhung

In Kalenderwoche 32 stieg der Preis des Jetson Orin Nano auf etwa CHF 500, was das gesamte Hardware-Budget aufgebraucht hätte. Die Evaluation wurde wiederholt und um den vorhandenen PC mit einer NVIDIA RTX 3060 Ti als Edge-Server erweitert.

Kriterium	Gewichtung	Jetson Orin Nano	PC mit der RTX 3060 Ti	Raspberry Pi 5 mit AI HAT	Raspberry Pi AI-Kamera
KI-Leistung, Echtzeit	30 %	5	5	3	2
Anschaffungspreis	25 %	1	5	4	5
Industrielle Relevanz, Integration	20 %	5	2	4	4
Lernkurve, Tools, Modellauswahl	10 %	3	5	3	3
Verfügbarkeit, Lieferzeit	15 %	4	5	4	4
Nutzwert	100 %	3,65	4,40	3,60	3,55

Entscheidung: Der **PC mit der RTX 3060 Ti** wird zum Edge-Server, im Folgenden als Edge-PC bezeichnet. Er ist sofort verfügbar, leistungsstark und mit Standard-Tools (Python, CUDA) nutzbar. Eine geringere Eignung für den industriellen Einsatz wird in Kauf genommen: Die Prüfsoftware ist so konzipiert, dass sie ohne Änderung des Ablaufs auf einen Jetson oder einen Industrie-PC verlagert werden kann.

5.3. Kamera

Variante	Beurteilung
Razer Kiyo (vorhanden)	Full HD, integriertes Ringlicht, manuelle Einstellung von Helligkeit, Kontrast, Sättigung, Weissabgleich und Fokus. Sofort verfügbar. Bildverarbeitung auf Consumer-Niveau.
Industriekamera mit Objektiv	Konstante Bildqualität, externer Trigger; gemäss Abgrenzung nicht Teil dieser Arbeit (2.4).
Raspberry Pi AI-Kamera	KI in der Kamera, jedoch begrenzte Modellgrösse; nach der Entscheidung in 5.2 nicht erforderlich.
HUSKYLENS 2	Einfache KI-Kamera für den Bildungsbereich, wenig Einfluss auf das Modell.

Entscheidung: Razer Kiyo.

5.4. Auslösesensor

Variante	Beurteilung
SICK WT2S-P231, Lichttaster	Industrieller Sensor mit Hintergrundausblendung, 12 bis 24 V, robust, als Gebrauchtteil kostengünstig[22].
TCRT5000, reflektierender optischer Sensor	Sehr kostengünstig, läuft direkt mit 3,3 V, weniger robust gegenüber Umgebungslicht. Ausweichlösung bei verspäteter Lieferung.
Farbsensor	Verworfen: für die Auslösung nicht erforderlich.
Auslösung über das Kamerabild	Kein zusätzlicher Sensor, aber ständige Bildauswertung und eine weniger wiederholbare Position. Verworfen.

Entscheidung: SICK WT2S-P231, Ausweichlösung TCRT5000. Die Software des Raspberry Pi 5 behandelt den Sensor als einfachen digitalen Eingang, sodass beide Sensoren ohne Änderung des Codes ausgetauscht werden können.

[22] SICK AG: Datenblatt WT2S-P231

5.5. Antrieb und Mechanik

Komponente	Auswahl	Begründung
Motor	NEMA-17-Schrittmotor (StepperOnline 17HE15-1504S)	Wiederholgenaue Positionierung ohne Encoder, Standardgrösse
Treiber	TMC2209 auf einer StepStick-Platine	Läuft leise bei der niedrigen Geschwindigkeit des Förderbands (4.7)
Kraftübertragung	GT2-Zahnriemen, Untersetzung 1:2	Bietet mehr Drehmoment und eine feinere Auflösung
Rahmen und Lager	MakerBeamXS-Profile, abgedichtete MR105-Lager	Verfügbar, leicht und modular; die Lager laufen reibungsarm
Förderband und gedruckte Teile	Gummiband, 40 mm breit (später durch Gewebeband ersetzt, 8.5), 3D-gedruckte Teile	Guter Halt; die Teile passen exakt zum Profilsystem

5.6. Steuerung und Elektronik

Thema	Auswahl	Begründung
Steuerung	Raspberry Pi 5 (vorhanden)	Trennt die zeitkritische Steuerung von der KI
Verkabelung	Freie Verkabelung mit Hebelklemmen	Flexibel, jedes Signal kann gemessen werden
Signalpegel und Versorgung	Optokoppler von 12 V auf 3,3 V; 12 V für Motor und Sensor, separate Versorgung für den Pi	Die Pi-Eingänge vertragen nur 3,3 V; alle Teile teilen sich eine gemeinsame Masse
Signalisierung	Mini-Signalsäule (rot, gelb, grün)	Bei der Demonstration gut sichtbar

5.7. KI-Framework und Annotation

Variante	Beurteilung
Ultralytics YOLO11	Aktuell, gut dokumentiert, Training mit wenigen Zeilen Python, echtzeitfähig. Lizenz AGPL-3.0: kein Problem für eine öffentliche Diplomarbeit; ein kommerzielles Produkt würde eine Lizenz erfordern [23].
TensorFlow Object Detection	Leistungsstark, erfordert jedoch einen deutlich höheren Lernaufwand.
Klassische Bildverarbeitung (OpenCV)	Kein Training, jedoch sind Regeln für jeden Fehlertyp und jede Beleuchtungssituation erforderlich.

Entscheidung: Ultralytics YOLO11. In der Machbarkeitsstudie wurden die Trainingsbilder in Label Studio (Open Source, Export im YOLO-Format)[24] annotiert, da die Objekterkennung für jede Mulde eine Box benötigt. Ob das Prüfband auch eine Objekterkennung benötigt, wird nach der Machbarkeitsstudie entschieden (7.4).

[23] Ultralytics: YOLO11-Dokumentation (Training, Validierung, Lizenz), https://docs.ultralytics.com

[24] Label Studio: Label Studio Documentation, https://labelstud.io/guide

5.8. Cloud-Plattform

5.8.1. Erste Evaluation

Variante	Beurteilung
Azure IoT Central	Fertige IoT-Anwendung mit Geräteverwaltung und Dashboards, keine Programmierung erforderlich; vorhandene Azure-Kenntnisse. Ausgewählt.
Azure IoT Hub	Kerndienst für Geräte und Nachrichten; Dashboard muss separat erstellt werden.
AWS IoT Core	Gleichwertiger Kerndienst, jedoch keine Vorkenntnisse in AWS vorhanden.
Google Cloud IoT Core	2023 eingestellt. Verworfen.

5.8.2. Neubewertung nach der Einstellung

Seit dem 23.09.2026 können in Azure IoT Central keine neuen Anwendungen mehr erstellt werden; bestehende laufen bis zum 20.09.2029 weiter[25]. Microsoft empfiehlt Azure IoT Hub in Verbindung mit Microsoft Fabric. Es zeichnet sich ein Muster ab: Google hat Cloud IoT Core eingestellt, AWS hält IoT SiteWise Monitor im Wartungsmodus, und Microsoft stellt IoT Central ein. Komplette IoT-Lösungen bergen ein hohes Risiko, eingestellt zu werden.

Entscheidung: Azure IoT Hub im kostenlosen Tarif F1 (8'000 Nachrichten pro Tag)[26]. Ein Kerndienst mit dem Standardprotokoll MQTT hält die Herstellerabhängigkeit gering: Ein Plattformwechsel betrifft nur die Verbindung, nicht die Prüfung.

[25] Microsoft Learn: Azure IoT Central is evolving, https://learn.microsoft.com/en-us/azure/iot-central/core/iot-central-evolution

[26] Microsoft Learn: Azure IoT Hub pricing, https://azure.microsoft.com/en-us/pricing/details/iot-hub/

5.8.3. Visualisierung der Prüfdaten

Bei IoT Central war das Dashboard Teil der Plattform. Bei IoT Hub muss es separat erstellt werden. Das Ziel ist eine Seite, die mindestens die Anzahl der IO- und NIO-Ergebnisse anzeigt (F09, UC2).

Variante	Beurteilung
Microsoft Fabric Real-Time Intelligence	Von Microsoft empfohlener Weg[27]. IoT Hub kann direkt als Quelle eines Eventstreams angebunden werden[28]. Kostenlose Testversion (Trial) für 60 Tage[29]. Erfordert ein Arbeits- oder Schulkonto. Ausgewählt.
Event Hub mit einem kostenlosen Azure-Data-Explorer-Cluster	Funktioniert ohne Arbeits- oder Schulkonto, aber ein Event Hub kostet etwa CHF 10 pro Monat und erfordert mehr Einrichtungsschritte.
Metriken im Azure-Portal	Kostenlos und schnell, zeigen jedoch nur die Anzahl der Nachrichten an, nicht IO und NIO. Verworfen.
Zwei IoT Hubs, einer für IO und einer für NIO	Ergebnisse pro Hub zählen. Verworfen, da pro Abonnement nur ein kostenloser IoT Hub möglich ist (6.9).

Entscheidung: Microsoft Fabric mit Eventstream, Eventhouse und Real-Time Dashboard. Die Testversion läuft 60 Tage und deckt somit auch gleich die Präsentation am 19.11.2026 ab. Der Weg zu einer funktionierenden Fabric-Umgebung war nicht ganz einfach; er wird in Abschnitt 8.10 beschrieben.

[27] Microsoft Learn: Azure IoT Central is evolving, https://learn.microsoft.com/en-us/azure/iot-central/core/iot-central-evolution

[28] Microsoft Learn: Add Azure IoT Hub source to an eventstream, https://learn.microsoft.com/en-in/fabric/real-time-intelligence/event-streams/add-source-azure-iot-hub

[29] Microsoft Learn: Try Microsoft Fabric for free, https://learn.microsoft.com/en-us/fabric/fundamentals/fabric-trial

5.9. Entscheidungsübersicht

Bereich	Entscheidung
Edge-Computer	PC mit RTX 3060 Ti
Steuerung	Raspberry Pi 5
Kamera	Razer Kiyo
Auslösesensor	SICK WT2S-P231, Ausweichlösung TCRT5000
Antrieb	NEMA 17 mit TMC2209, GT2 1:2
KI-Framework	Ultralytics YOLO11; die Methode für das Prüfband ergibt sich aus der Machbarkeitsstudie (7.4)
Cloud	Azure IoT Hub F1
Visualisierung	Microsoft Fabric Real-Time Dashboard (Testversion)

6. Systemkonzept

Das Systemkonzept beschreibt das Prüfband, wie es schliesslich aufgebaut wurde. Es enthält daher bereits die Entscheidungen aus der Machbarkeitsstudie in Kapitel 7, vor allem Klassifizierung anstelle von Objekterkennung.

6.1. Systemübersicht und Architekturprinzipien

Das Prüfband ist in vier Ebenen unterteilt: Feldebene, Steuerungsebene mit dem Raspberry Pi 5, Edge-PC und Cloud. MQTT verbindet die Ebenen lose miteinander.

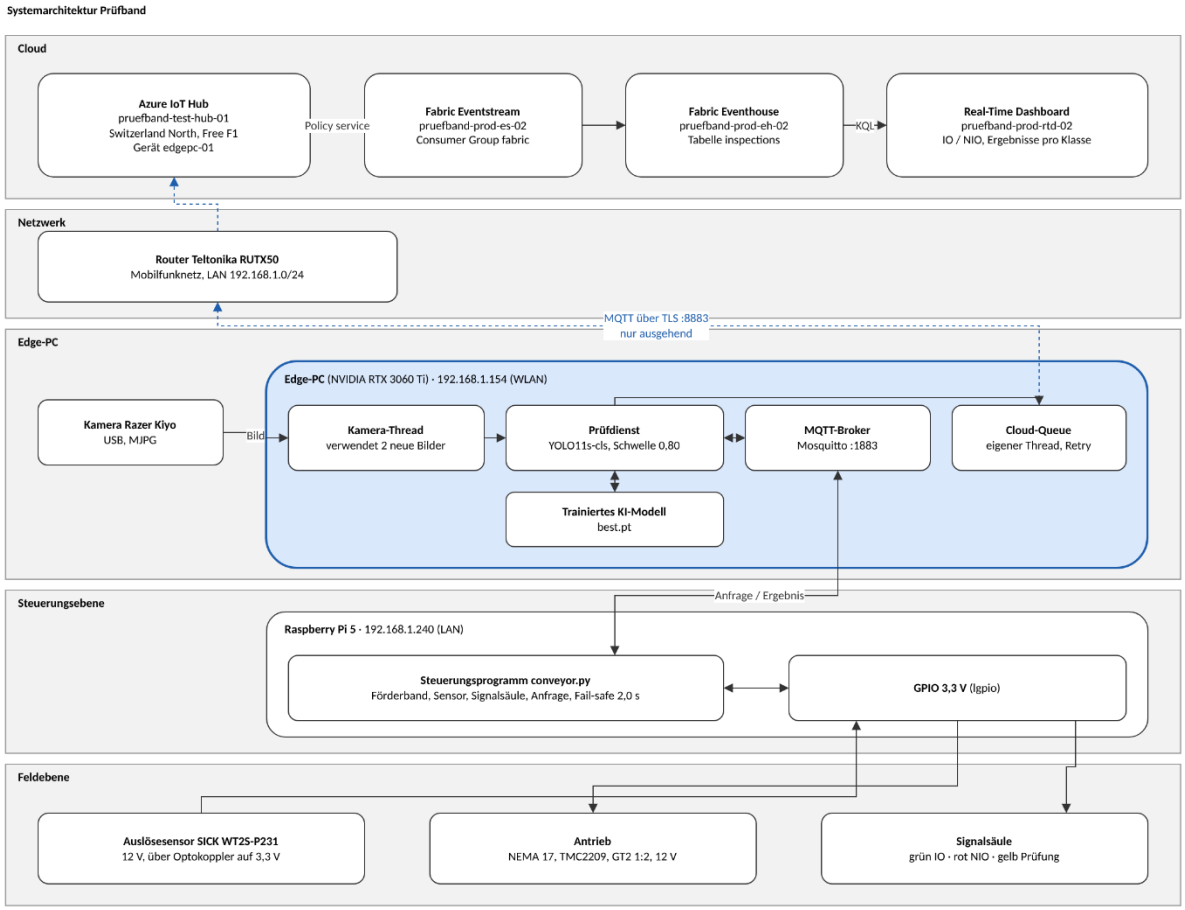


Abbildung 12: Systemarchitektur (mit KI generierte Darstellung, draw.io)

Ebene	Gerät oder Dienst	Aufgaben
Feldebene	Sensor, Schrittmotor mit Treiber, Signalsäule	Fördern, Auslösen, Anzeigen
Steuerungsebene	Raspberry Pi 5	Motor ansteuern, Sensor auslesen, Signalsäule schalten, Prüfung anfordern
Edge-PC	PC mit RTX 3060 Ti, Kamera, lokalem MQTT-Broker	Bildaufnahme, KI-Prüfung, Entscheidung, Senden an die Cloud

Cloud	Azure IoT Hub, Microsoft Fabric	Geräteidentität, Empfang der Ergebnisse, Speicherung, Dashboard

Architekturprinzipien

- Lokal entscheiden. Die Prüfentscheidung und die Signalisierung funktionieren ohne die Cloud. Bei einem Internetausfall arbeitet das Prüfband weiter.
- Aufgaben trennen. Die zeitkritische Steuerung läuft auf dem Raspberry Pi, der sich gut für die Steuerung einer Maschine eignet. Die rechenintensive KI läuft auf dem Edge-PC, der in Zukunft für komplexere Modelle aufgerüstet werden kann.
- Lose koppeln. Die Teilsysteme kommunizieren über MQTT; ein Teilsystem kann ersetzt werden, ohne die anderen zu verändern.
- Im Zweifel ablehnen. Unsichere Ergebnisse gelten als nicht gut.
- Standards vor Sonderlösungen. Offene Protokolle und Cloud-Kerndienste.

6.2. Netzwerk- und IP-Konzept

Alle Geräte befinden sich hinter einem eigenen Router mit Mobilfunkverbindung, sodass das Prüfband nicht vom Netzwerk am Präsentationsort abhängig ist. Der Router vergibt Adressen im Netzwerk 192.168.REDACTED. Der Raspberry Pi ist per LAN-Kabel angeschlossen, der Edge-PC über WLAN.

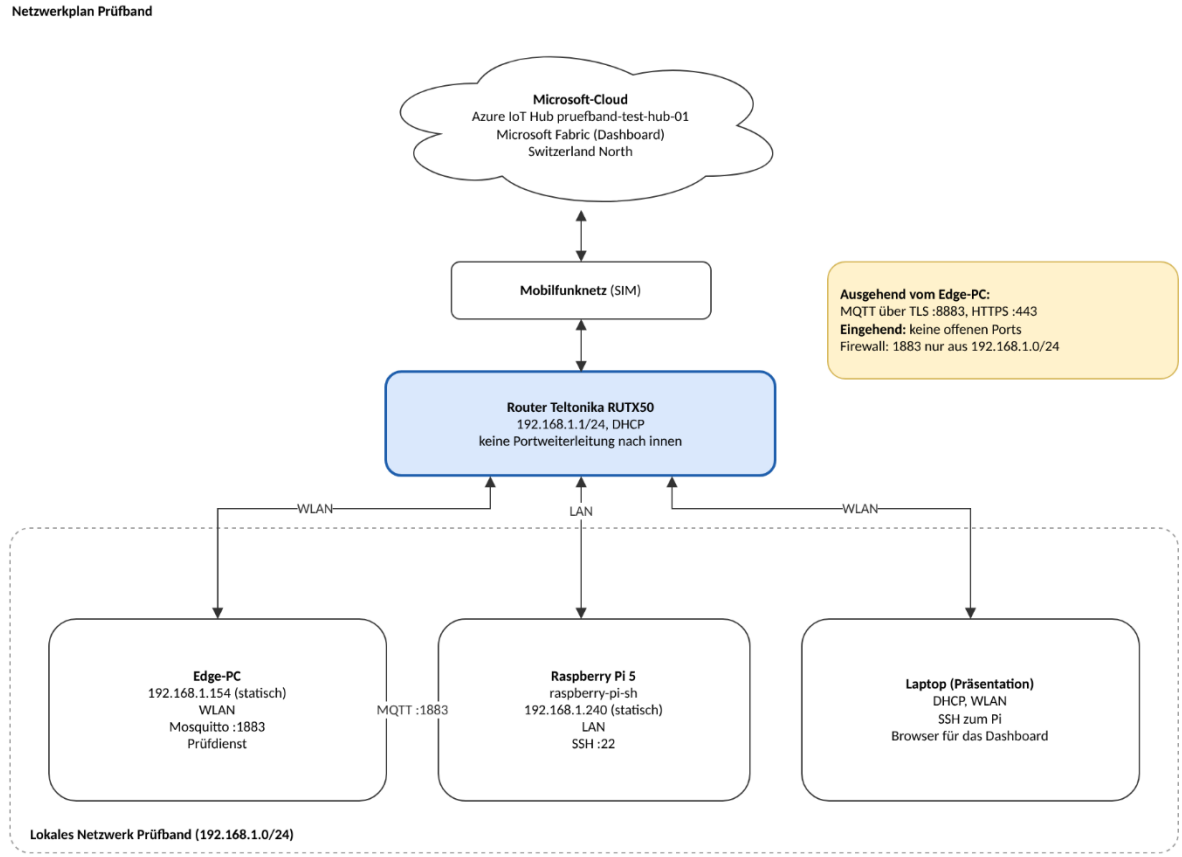


Abbildung 13: Netzwerkplan (mit KI generierte Darstellung, draw.io)

REDACTED

Von aussen ist kein Port offen. Eine Firewall-Regel auf dem Edge-PC lässt Port 1883 nur aus dem lokalen Netzwerk 192.168.REDACTED zu.

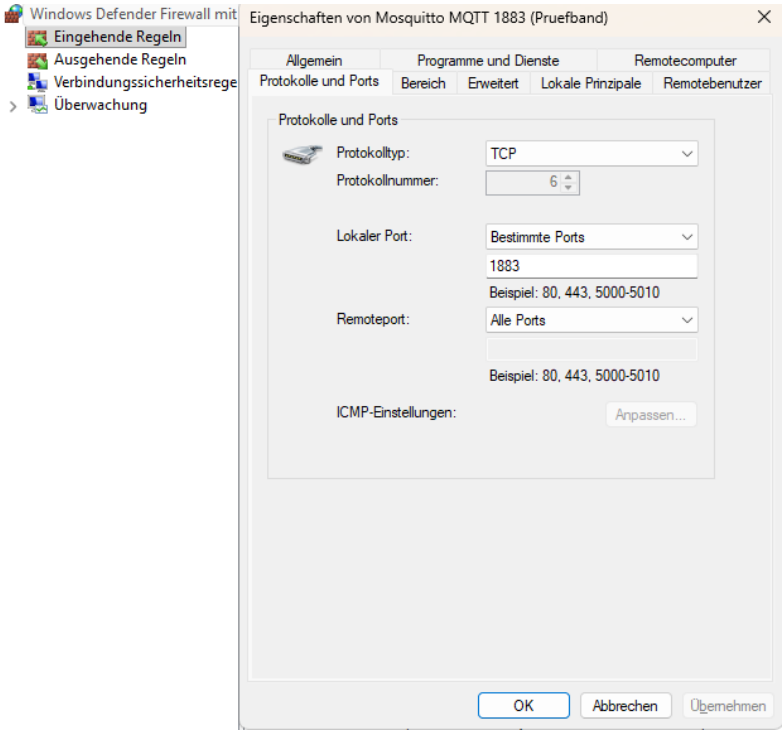


Abbildung 14: Firewall-Regel 1883 (eigene Aufnahme)

6.3. Komponentenmodell Hardware

Die Hardware besteht aus dem Edge-PC (Dell XPS 8940 mit NVIDIA RTX 3060 Ti), der Razer-Kiyo-Webcam mit Ringlicht, dem Raspberry Pi 5, dem Antrieb mit NEMA-17-Motor und TMC2209-Treiber, dem SICK-Auslösesensor mit Optokoppler, der Signalsäule, zwei Netzteilen und dem Router Teltonika RUTX50.

6.4. Komponentenmodell Software

Gerät	Software	Aufgabe
Edge-PC	Windows, Python 3.12.10, PyTorch 2.6.0 mit CUDA, Ultralytics 8.4.172	Führt das trainierte KI-Modell aus
Edge-PC	Prüfdienst (inspection_service.py)	Empfängt die Anfrage, nimmt das Bild auf, trifft die Entscheidung, sendet das Ergebnis
Edge-PC	Mosquitto 2.1.2	Lokaler MQTT-Broker
Edge-PC	paho-mqtt 1.6.1, azure-iot-device 2.14.0	MQTT-Client, Übertragung an Azure IoT Hub
Raspberry Pi	Raspberry Pi OS Lite 64-Bit, Python 3, lgpio, paho-mqtt 2.1	Steuerungsprogramm (conveyor.py): Motor, Sensor, Signalsäule, Prüfanfrage
Cloud	Azure IoT Hub, Microsoft Fabric	Empfang, Speicherung, Dashboard
Entwicklung	Visual Studio Code, Label Studio (nur Machbarkeitsstudie)	Entwicklung, Annotation

Der Edge-PC verwendet paho-mqtt 1.6.1, da azure-iot-device 2.14 eine Version unter 2.0 erfordert. Der Prüfdienst ist so programmiert, dass er mit beiden Versionen funktioniert.

6.5. Prüfablauf

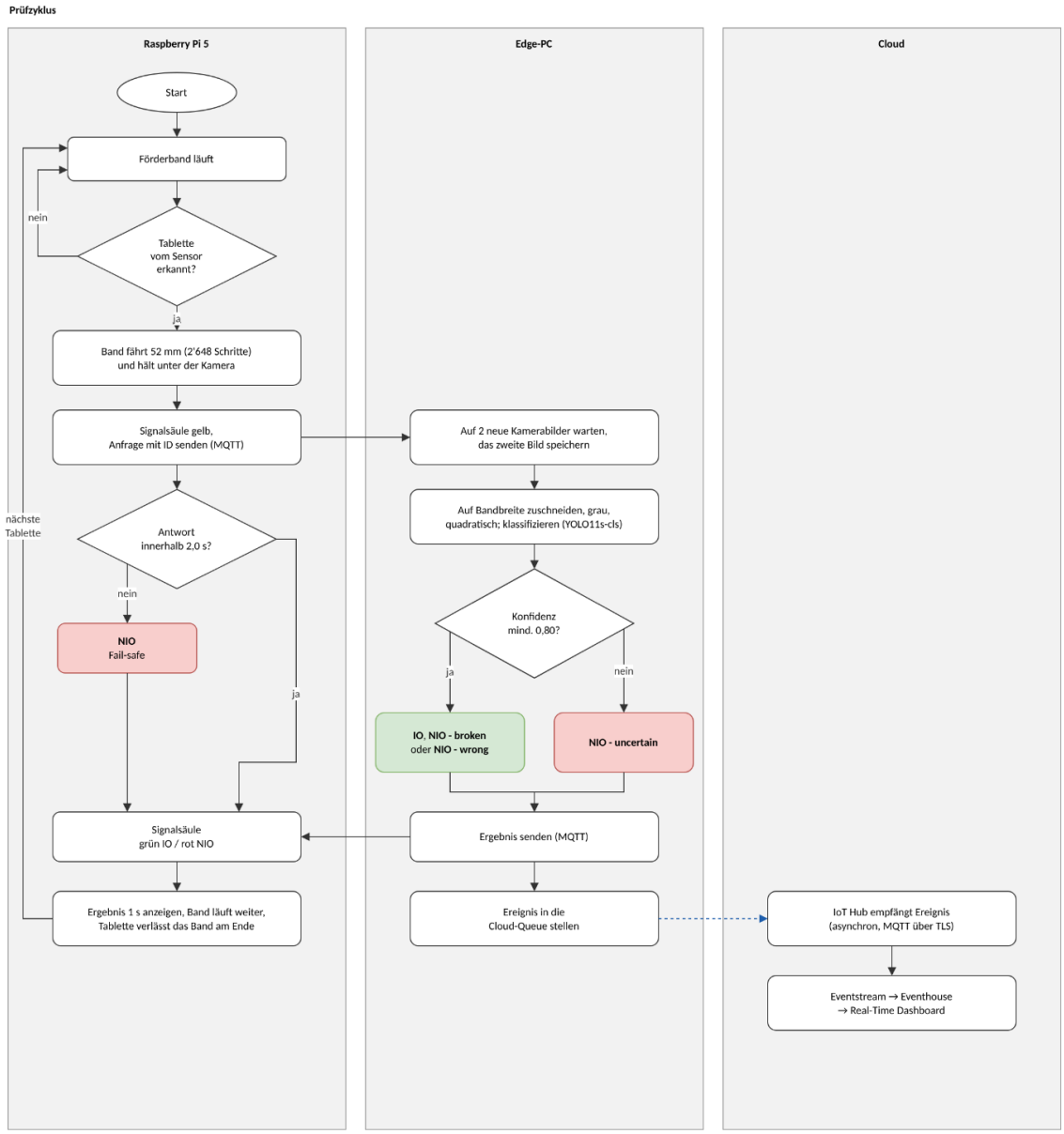


Abbildung 15: Prüfzyklus (mit KI generierte Darstellung, draw.io)

1. Das Förderband läuft; der Sensor erkennt eine Tablette.
2. Der Raspberry Pi bewegt das Band um eine feste Strecke von 52 mm, bis sich die Tablette unter der Kamera befindet, und hält dann an.
3. Der Pi sendet über den lokalen Broker eine Prüfanfrage mit einer eindeutigen ID an den Edge-PC und schaltet die Signalsäule auf Gelb.
4. Der Edge-PC wartet auf zwei neue Kamerabilder, speichert das zweite, klassifiziert es und wendet die Prüflogik an.
5. Der Edge-PC sendet das Ergebnis zurück; der Pi schaltet die Signalsäule auf Grün (IO) oder Rot (NIO). Trifft innerhalb von 2,0 s keine Antwort ein, zeigt der Pi NIO an (ausfallsicher).
6. Unabhängig davon meldet der Edge-PC das Ergebnis an die Cloud.
7. Nach 1 s läuft das Förderband wieder; die Tablette verlässt das Band an dessen Ende.

6.6. Prüflogik

Die Logik folgt der Regel «im Zweifel nicht gut». Das Modell gibt eine Klasse und einen Konfidenzwert zurück. Ein Ergebnis gilt nur dann als sicher, wenn die Konfidenz mindestens 0,80 beträgt. Die Ergebnisbezeichnungen entsprechen den Texten, die die Software ausgibt und an die Cloud sendet.

Ergebnis	Bedingung	Signalsäule
IO	Klasse «io» mit einer Konfidenz von mindestens 0,80	grün
NIO - broken	Klasse «nio_broken» (gebrochen) mit einer Konfidenz von mindestens 0,80	rot
NIO - wrong	Klasse «nio_wrong» (falsche Tablette) mit einer Konfidenz von mindestens 0,80	rot
NIO - uncertain	Jede Klasse mit einer Konfidenz unter 0,80 (unsicher)	rot
NIO - error	Kamera- oder Modellfehler auf dem Edge-PC	rot
NIO (fail-safe)	Keine Antwort vom Edge-PC innerhalb von 2,0 s	rot

Nur das Ergebnis IO gilt als gut. Alle anderen Fälle werden als NIO angezeigt.

6.7. Datenmodell

Alle Nachrichten sind im JSON-Format. Die lokalen Nachrichten sind auf ein Minimum beschränkt. Die Cloud-Nachrichten enthalten ein versioniertes Schema, damit spätere Erweiterungen bestehende Auswertungen nicht beeinträchtigen.

Nachricht	Pfad	Felder
Prüfanfrage	Pi an Edge-PC, Topic pruefband/request, QoS 1	id
Prüfergebnis	Edge-PC an Pi, Topic pruefband/result, QoS 1	id, result
Prüfereignis (Schema inspection.v2)	Edge-PC an IoT Hub	schema, id, timestamp (UTC), result, confidence, inspection_ms
Betriebsdaten (Schema status.v1)	Edge-PC an IoT Hub	schema, id (Gerät), timestamp (UTC), result (started oder stopped), confidence, inspection_ms

Die ID hat das Format «Unixzeit-Zähler», zum Beispiel 1792612353-0001. Sie verknüpft die Anfrage, das Ergebnis, die Bilddatei und das Cloud-Ereignis. Das Schema inspection.v2 löst pruefergebnis.v1 aus der Machbarkeitsstudie ab. Die Statusmeldung verwendet dieselben Felder wie ein Prüfereignis, passt somit in dieselbe Tabelle und wird durch das Feld «schema» unterschieden.

6.8. Sicherheitskonzept

Thema	Umsetzung
Verschlüsselung	MQTT über TLS in die Cloud, mindestens TLS-Version 1.2
Geräteidentität	Eigenes Gerät edgepc-01 im IoT Hub mit symmetrischem Schlüssel
Anmeldedaten	In separaten Konfigurationsdateien (mqtt.json, iothub.json), niemals im Code; Schlüssel werden bei jedem Verdacht neu generiert
Lokaler Broker	Kein anonymer Zugriff, Benutzername und Passwort; nur im lokalen Netzwerk erreichbar
Cloud-Zugriff für Fabric	Shared Access Policy «service» mit Leserechten anstelle von «iothubowner»; eigene Consumer Group «fabric»
Konten	Separater Fabric-Benutzer ausschliesslich mit der Rolle «Contributor»; das Eigentümerkonto wird nicht für Fabric verwendet
Netzwerk	Nur ausgehende Verbindungen
Elektrische Sicherheit	Kleinspannung 12 V, Pegelumsetzung auf 3,3 V, gemeinsame Masse

6.9. Cloud-Konzept

6.9.1. IoT Hub

Region Switzerland North (Daten verbleiben in der Schweiz; Partnerregion für Disaster Recovery: Switzerland West), Tarif Free F1, öffentlicher Zugriff mit mindestens TLS-Version 1.2. Ein privater Endpunkt wird nicht verwendet: Die Verbindung läuft über ein Mobilfunknetz mit wechselnder IP-Adresse, und ein virtuelles Netzwerk mit VPN wäre für einen Demonstrator überdimensioniert. Defender for IoT, Device Provisioning Service und Device Update sind deaktiviert, da nur ein Gerät manuell registriert wird.

6.9.2. Namenskonvention

Muster <Workload>-<Umgebung>-<Ressourcentyp>-<Nr.>, zum Beispiel pruefband-test-rg-01 (Ressourcengruppe der Machbarkeitsstudie) und pruefband-test-hub-01 (IoT Hub). Die Organisation wird weggelassen, da Organisationen durch eigene Abonnements voneinander getrennt sind. Die Region wird weggelassen, da eine Ressourcengruppe Ressourcen aus mehreren Regionen enthalten kann und das Projekt nur eine nutzt. Im Gegensatz zum Microsoft Cloud Adoption Framework steht die Typabkürzung an letzter Stelle und wird konsistent verwendet[30].

6.9.3. Bewusste Abweichungen

- Der IoT Hub der Machbarkeitsstudie, pruefband-test-hub-01, wird auch für das Prüfband verwendet. Pro Abonnement ist nur ein kostenloser IoT Hub möglich. Deshalb wurde pruefband-**test**-hub-01 weiterverwendet und kein neuer Hub mit einem Produktionsnamen angelegt.
- Die Fabric-Elemente tragen durchgehend die Nummer 02 (pruefband-prod-ws-02, -eh-02, -es-02, -rtd-02), da der erste Workspace ersetzt werden musste (8.10).
- Ein Fabric-Kapazitätsname darf nur Kleinbuchstaben und Ziffern enthalten, daher kann das Muster dort nicht angewendet werden.

6.9.4. Fabric

Workspace pruefband-prod-ws-02 auf der Testkapazität (Trial), Eventstream pruefband-prod-es-02 mit IoT Hub als Quelle, Eventhouse pruefband-prod-eh-02 mit der Tabelle inspections und das Real-Time Dashboard pruefband-prod-rtd-02.

6.9.5. Kostenüberwachung

Ein Budget auf dem Abrechnungskonto löst bei 50, 80 und 100 % der Ist-kosten sowie bei 100 % der prognostizierten Kosten eine E-Mail-Benachrichtigung aus[31]. Ein zweites Budget von CHF 20 pro Monat deckt die Produktions-Ressourcengruppe ab. Ein Budget warnt lediglich; es stoppt die Kosten nicht. Tags (Workload, Umgebung, Eigentümer, Projekt) ermöglichen eine Kostenauswertung pro Projekt und Umgebung.

[30] Microsoft Learn: Cloud Adoption Framework, naming convention, https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming

[31] Microsoft Learn: Cost Management, budgets and alerts

6.10. Elektrisches Konzept

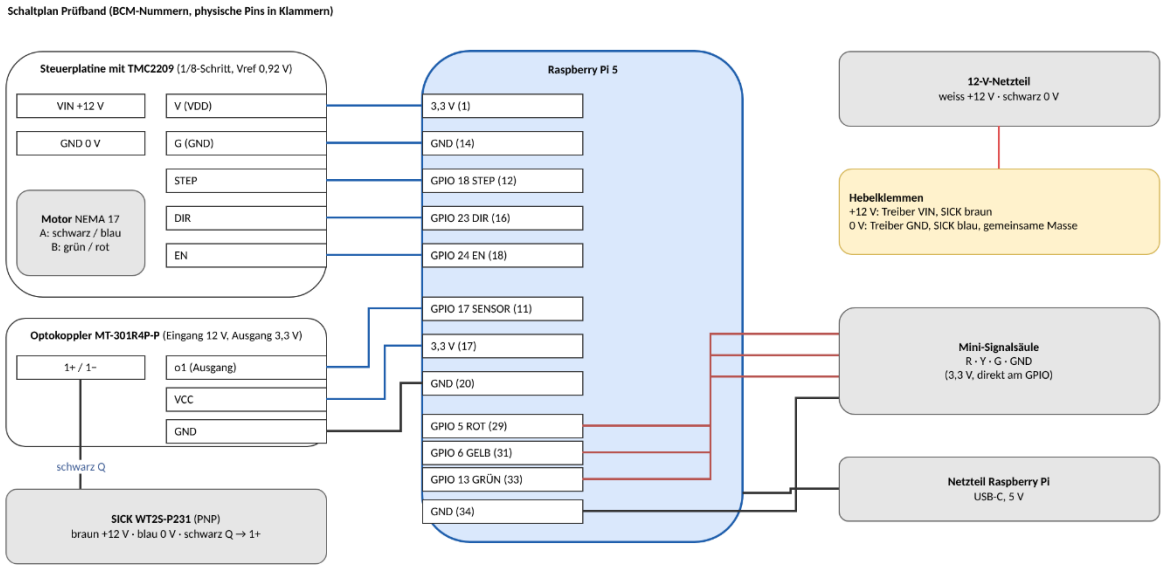


Abbildung 16: Schaltplan (mit KI generierte Darstellung, draw.io)

Alle Signale am Raspberry Pi arbeiten mit 3,3 V. Der Motortreiber und der Industriesensor werden über Hebelklemmen mit 12 V versorgt; das Sensorsignal gelangt über einen Optokoppler, der 12 V in 3,3 V umwandelt, zum Pi. Der Treiber arbeitet mit 1/8-Mikroschritten, und alle Baugruppen teilen sich eine gemeinsame Masse.

6.11. Mechanisches Konzept

Ein Rahmen aus MakerBeamXS-Profilen trägt zwei Rollen mit dem Förderband, ein Gleitbett unter dem oberen Trum, ein Sensorhalterung, einen Kameramast und ein Signalhalterung. Eine Rutsche und ein Auffangbecher am Ende des Bandes waren geplant, werden für den Demonstrator jedoch nicht benötigt (8.5). Der Motor sitzt auf der Grundplatte und treibt die Antriebsrolle über einen Zahnriemen an.

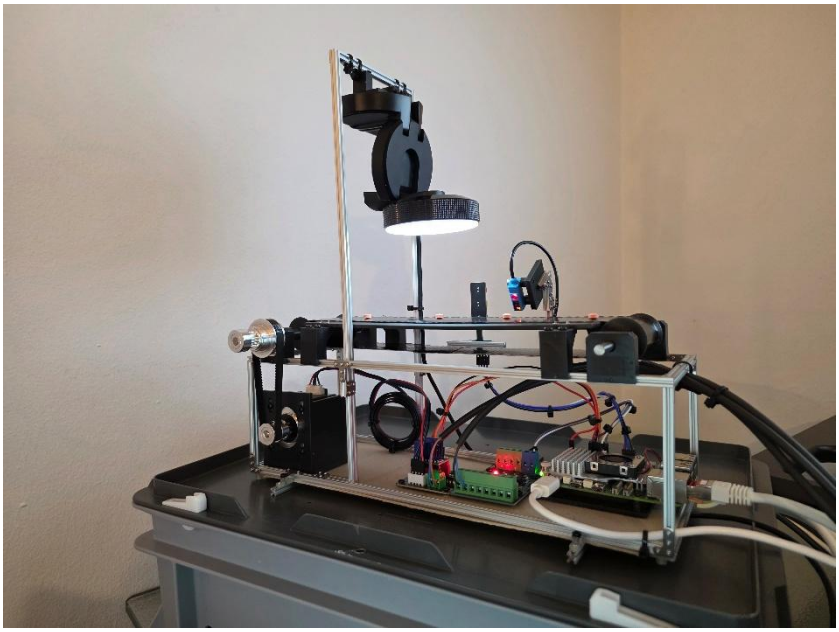


Abbildung 17: Prüfband Aufbau Ansicht 1 (eigene Aufnahme)

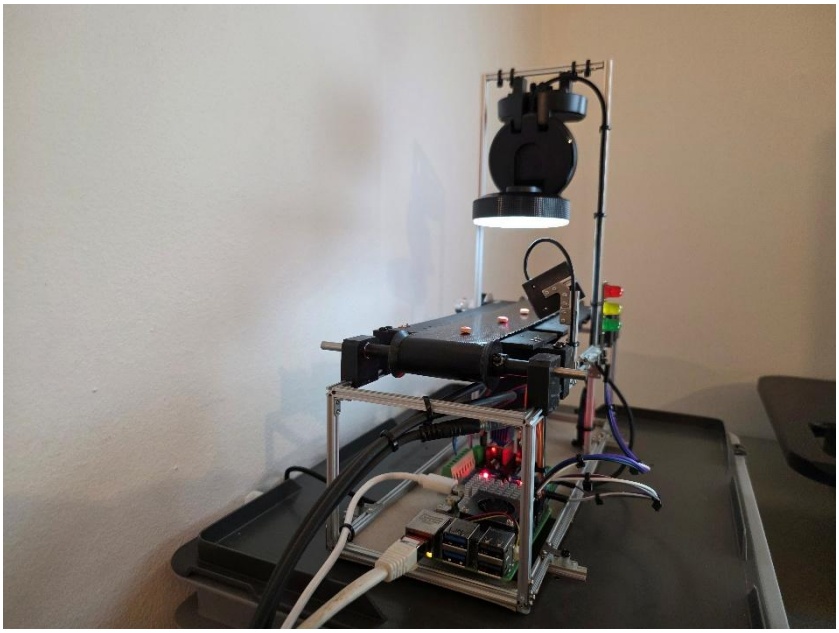


Abbildung 18: Prüfband Aufbau Ansicht 2 (eigene Aufnahme)

Die gedruckten Teile (Lagerblöcke, Rollen, Motorhalterung, Gleitbett, Halterungen für Sensor, Kamera und Signalsäule) wurden mit Claude als parametrische OpenSCAD-Modelle erstellt; zwei Testplatten dienten zur Überprüfung der kritischen Passungen, bevor alle Teile gedruckt wurden. Da die Konstruktion nicht im Mittelpunkt einer Informatikarbeit steht, wird sie nicht weiter beschrieben.

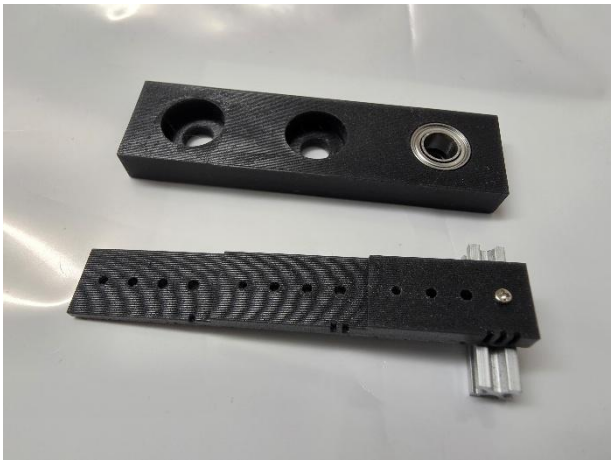


Abbildung 19: Testplatten (eigene Aufnahme)

7. Machbarkeitsstudie

7.1. Ziel und Vorgehensweise

Die Machbarkeitsstudie (Inkrement 0) befasst sich vor dem Bau des Prüfbands mit den beiden grössten technischen Unbekannten:

- Bildverarbeitung: Erkennt ein mit wenigen eigenen Bildern trainiertes Modell Fehler zuverlässig, schnell genug und ohne Schlupf?
- Cloud: Kann der Edge-PC Ergebnisse sicher und zuverlässig an die Cloud-Plattform senden?

Kamera, Bilderkennung und Cloud-Zugriff waren bereits in früheren eigenen Experimenten ausprobiert worden. Die Studie überprüfte diese vom 03. bis 05.10.2026 anhand von Tablettenblistern mit zehn Mulden: Sie kamen ohne jegliche Mechanik aus und beantworten dieselbe Kernfrage wie das Prüfband, nämlich ob eine Consumer-Kamera und eine Handvoll Trainingsbilder für eine sichere Prüfung ausreichen.

7.2. Machbarkeit der Bildverarbeitung

7.2.1. Testaufbau

Die Machbarkeitsstudie lief auf dem Edge-PC mit Python 3.12.10, PyTorch 2.6.0 und Ultralytics 8.4.172. Die Razer Kiyo wurde 10 cm über dem Blister auf einer dunklen, matten Oberfläche in einem abgedunkelten Raum angebracht, mit deaktiviertem Autofokus, festem Weissabgleich und dem Ringlicht auf Minimalstufe. Diese festen Einstellungen verhindern, dass die Kamera die Bilder automatisch anpasst.

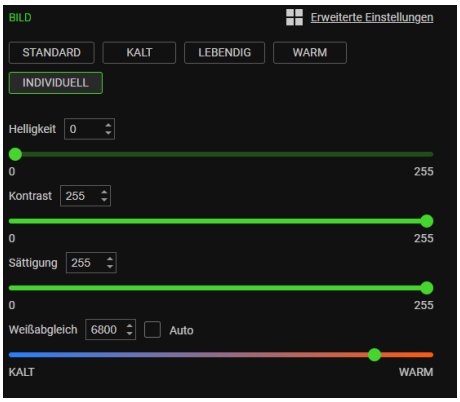


Abbildung 20: Einstellungen Kamera (eigene Aufnahme)

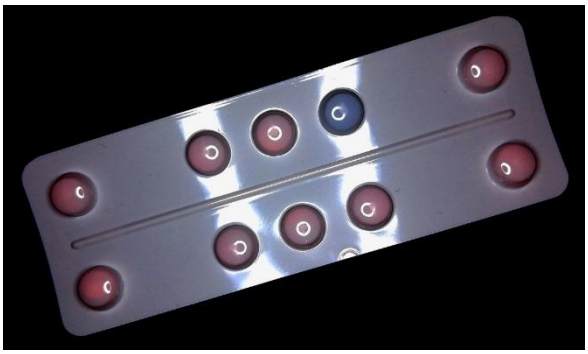


Abbildung 21: Blister mit einer leeren Mulde (eigene Aufnahme)

7.2.2. Datensatz und Annotation

Für das Training wurden 65 Bilder aufgenommen (32 vollständige und 33 defekte Blister). 34 davon wurden in Label Studio mit den Klassen «tablette» (Mulde gefüllt) und «leer» (Mulde leer) annotiert und stratifiziert im Verhältnis 80:20 in Trainings- und Validierungsdaten aufgeteilt.

Jede sichtbare Mulde erhält eine Box, auch am Bildrand, sofern ihre Klasse eindeutig ist. Mulden, die vollständig ausserhalb des Bildes liegen, werden nicht annotiert; die Prüflogik behandelt sie als nicht erkennbar.

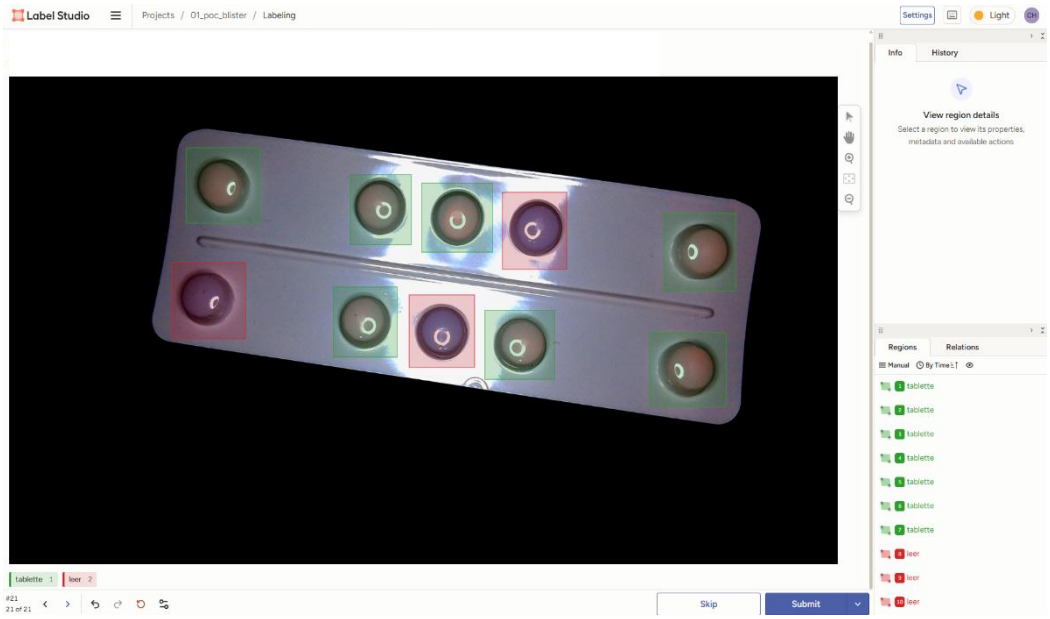


Abbildung 22: Annotation eines Blisters in Label Studio (eigene Aufnahme)

Datensatz	Bilder	Boxen «tablette»	Boxen «leer»
Training	27	213	57
Validierung	7	57	12

7.2.3. Training

YOLO11s wurde mittels Transfer Learning trainiert. Die Farbaugmentierung wurde bewusst gering gehalten (keine Farbtonverschiebung, Sättigung und Helligkeit ±20 %, Spiegelung, Drehung bis zu 5°), damit die Farbe als Merkmal nutzbar blieb.

Version	Batch	Patience (Epochen)	mAP50	Beurteilung
v1	16	30	0,06	Fehlgeschlagen: Bei 27 Bildern gibt es nur 2 Lernschritte pro Epoche, und das Training brach während der Aufwärmphase ab.
v2	8	100 (max. 300 Epochen)	0,995	Erfolgreich: mehr Lernschritte pro Epoche und eine längere Patience.

Validierung von v2: Precision 0,996, Recall 1,0, mAP50 0,995, mAP50-95 0,681, Rechenzeit 5,1 ms pro Bild auf der Grafikkarte.

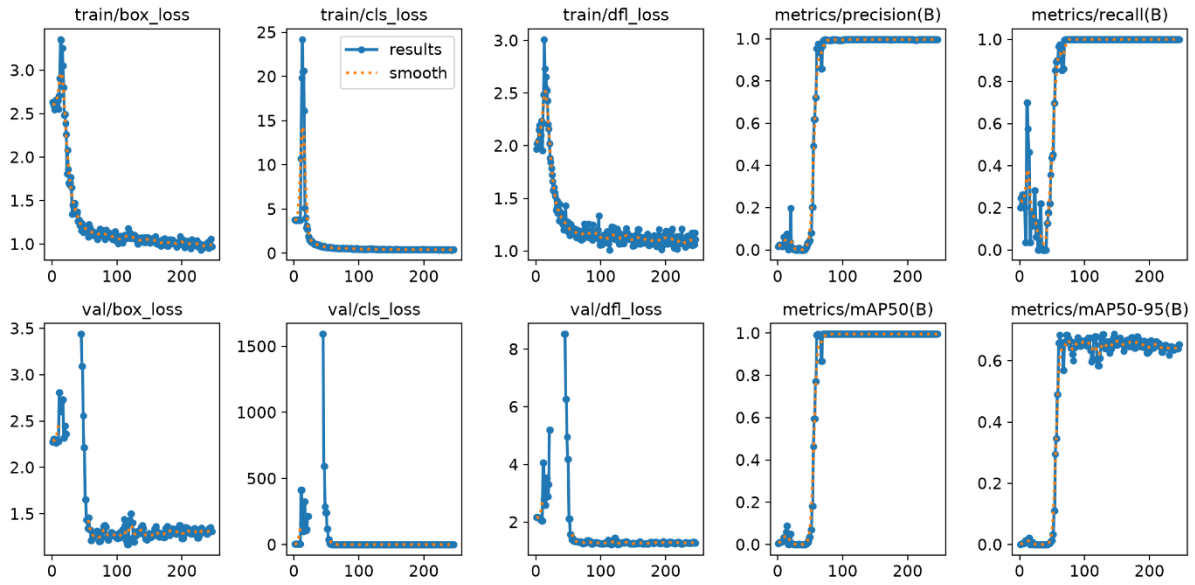


Abbildung 23: Resultat Lernkurve v2 (eigene Aufnahme aus Ultralytics)

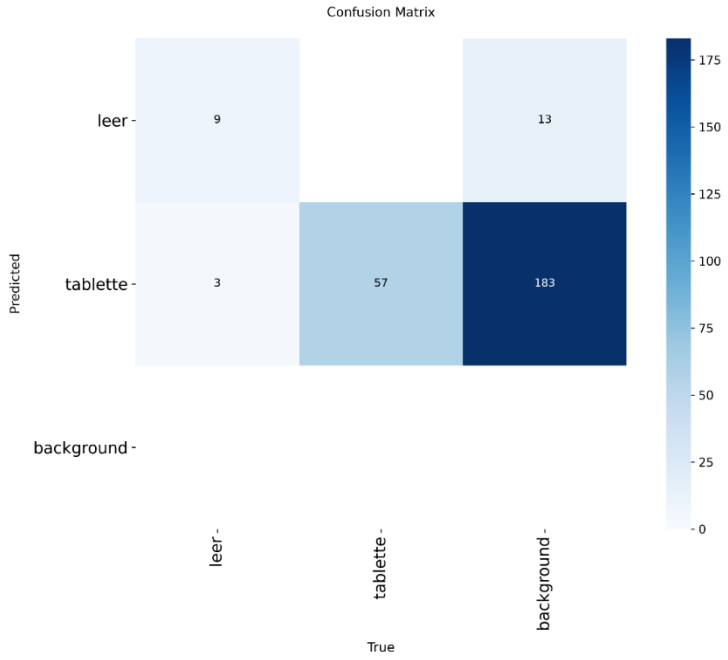


Abbildung 24: Konfusionsmatrix v2 (eigene Aufnahme aus Ultralytics)

7.2.4. Prüflogik

Parameter	Wert	Bedeutung
Mindestkonfidenz	0,25	Erkennungen unterhalb dieses Wertes werden verworfen.
Sichere Konfidenz	0,60	Erkennungen unterhalb dieses Wertes gelten als unsicher.
Randabstand	10 Pixel	Objekte, die den Bildrand berühren, gelten als unsicher.
Erwartete Anzahl	10 Mulden	Jede Mulde muss sicher erkannt werden.

Ergebnis	Bedingung
IO	10 sicher erkannte Mulden, alle mit Tablette
NIO_LEER (leer)	Mindestens eine Mulde ohne Tablette
NIO_UNVOLLSTAENDIG (unvollständig)	Weniger als 10 sicher erkannte Mulden; Meldung «nur X von 10 Mulden erkennbar»

7.2.5. Testdurchführung und Ergebnisse

Für den Test wurden 76 neue Bilder verwendet, die zuvor nie im Training eingesetzt worden waren, aufgeteilt in vier Gruppen.

Testgruppe	Bilder	Erwartet	Ergebnis	Schlupf
Vollständige Blister	31	IO	30 korrekt, 1 Pseudoausschuss	0
Blister mit leeren Mulden	15	NIO_LEER	15 korrekt	0
Blister, die nicht vollständig im Bild zu sehen sind	15	NIO_UNVOLLSTAENDIG	alle NIO_LEER, da diese Blister ebenfalls leere Mulden aufwiesen	0
Vollständige Blister mit zusätzlichem Licht	15	IO	5 korrekt, 10 Pseudoausschüsse	0

Prüfzeit von der Aufnahme bis zur Entscheidung: Mittelwert 24,0 ms, 95. Perzentil 31,2 ms, Maximum 41,3 ms.

Beispiel - IO

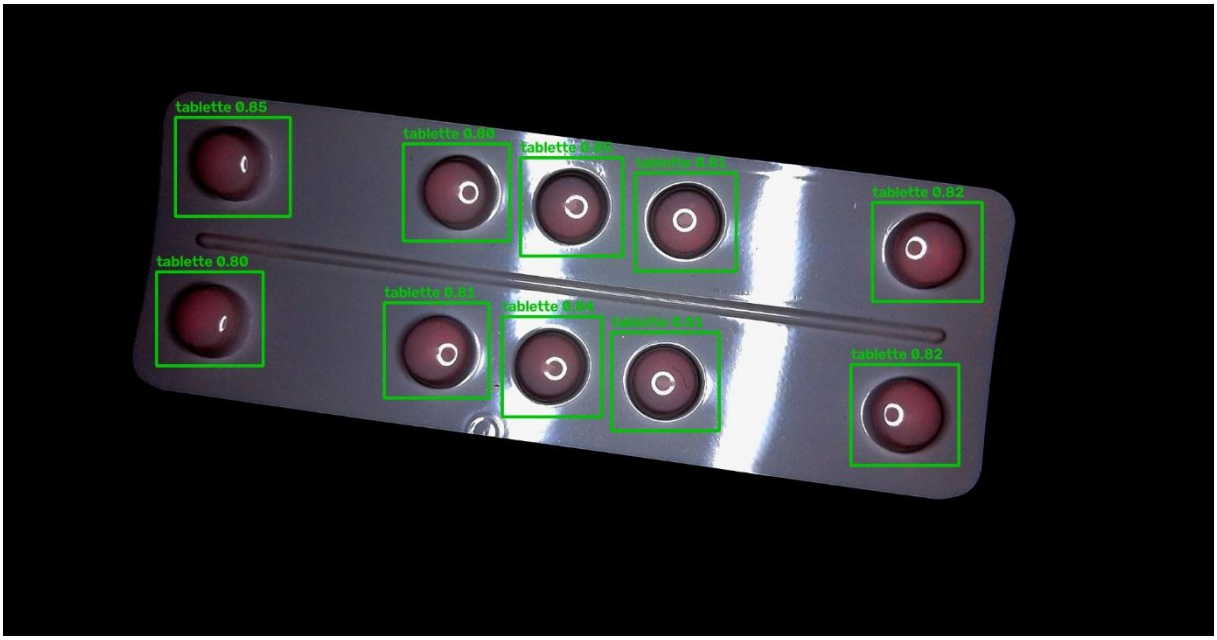


Abbildung 25: Machbarkeitsstudie Blister IO (eigene Aufnahme aus Ultralytics)

Beispiel - NIO_LEER

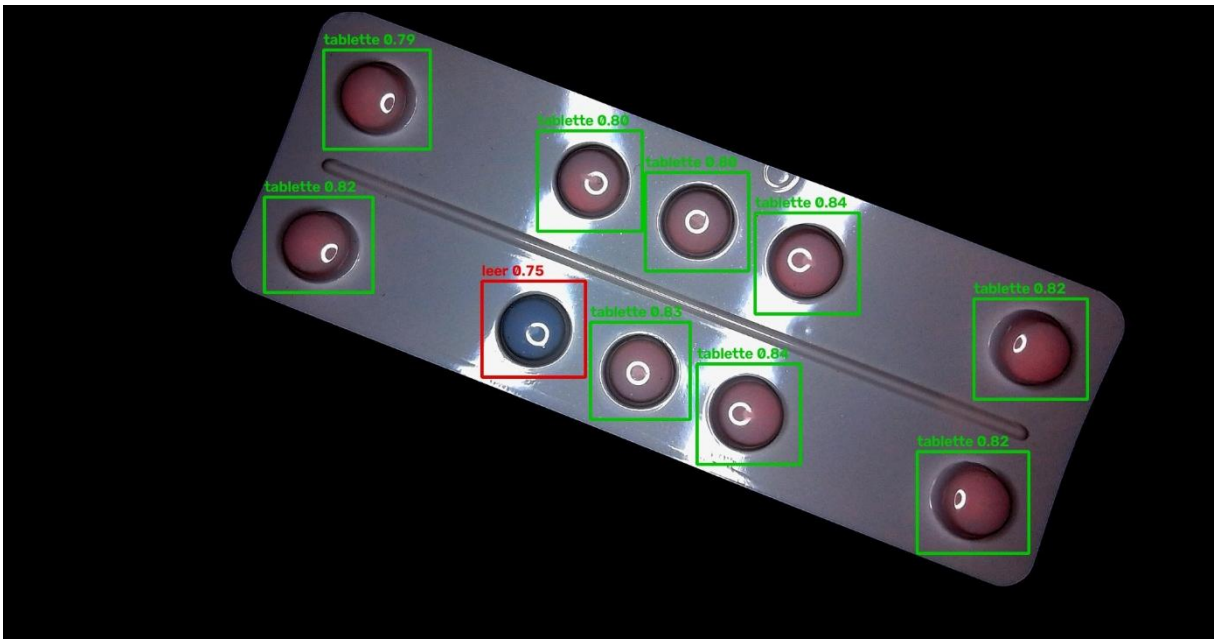


Abbildung 26: Machbarkeitsstudie Blister NIO_L (eigene Aufnahme aus Ultralytics)

Beispiel - NIO_UNVOLLSTAENDIG

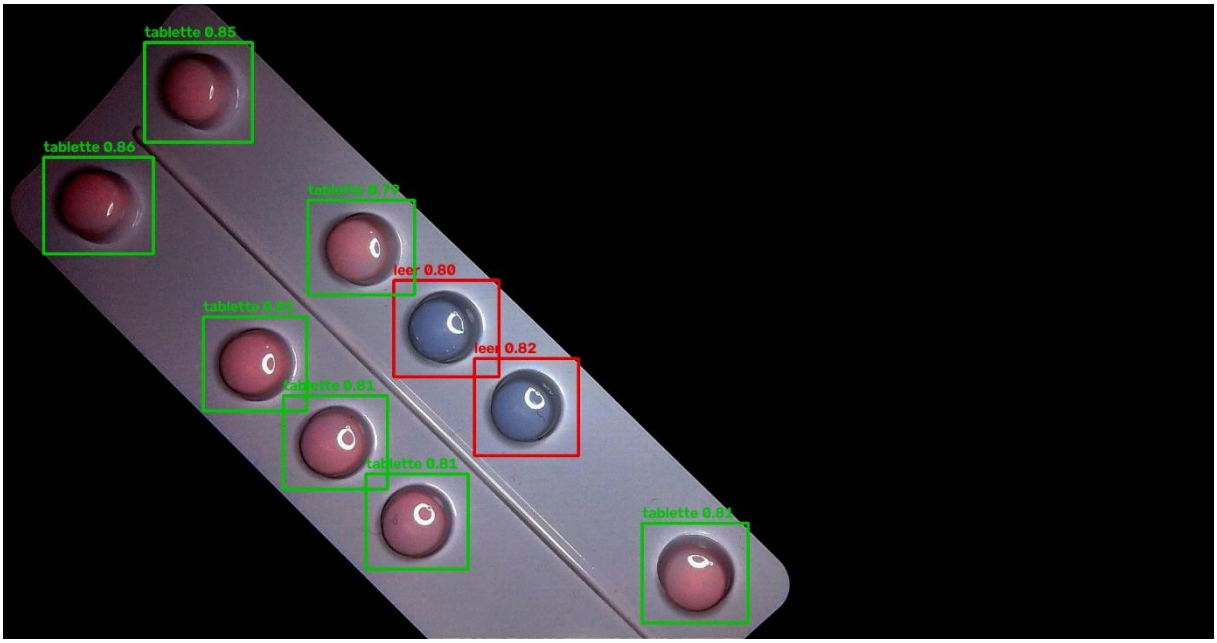


Abbildung 27: Machbarkeitsstudie Blister NIO_U (eigene Aufnahme aus Ultralytics)

7.2.6. Auswertung

- Im Normalbetrieb waren von den vollständigen und fehlerhaften Blistern (46 Bilder) 45 Entscheidungen korrekt: 97,8 %.
- In keiner Gruppe gab es Schlupf. Die ausfallsichere Prüflogik funktioniert.
- Zusätzliches Licht ist die grösste Störung. Zwei Drittel der guten Blister wurden aussortiert, als eine zusätzliche Lampe das Bild überbelichtete. Trotz fester Einstellungen nimmt die Kamera intern weiterhin Anpassungen vor, und die geringe Farbaugmentierung macht das Modell lichtempfindlich.

7.3. Machbarkeit der Cloud

7.3.1. Testaufbau

IoT Hub pruefband-test-hub-01 in der Ressourcengruppe pruefband-test-rg-01, Region Switzerland North, Tarif Free F1. Gerät edgepc-01 mit einem symmetrischen Schlüssel. Python-Bibliothek azure-iot-device 2.14.0, Verbindung über MQTT über TLS (Port 8883). Zehn simulierte Ergebnisse im Schema pruefergebnis.v1 wurden in Intervallen von 3 Sekunden gesendet.

7.3.2. Testdurchführung und Ergebnisse

Messgrösse	**Wert**
Gesendete und bestätigte Nachrichten	10 von 10
Erste Nachricht einschliesslich Verbindungsaufbau	352,1 ms
Folgende Nachrichten	111,7 bis 155,7 ms
Mittelwert	148,8 ms
Nachweis	Nachrichtenzähler in der IoT-Hub-Übersicht

REDACTED

Abbildung 28: Overview Azure IoT Hub (eigene Aufnahme)

7.3.3. Auswertung

Die Übertragung ist zuverlässig und verschlüsselt. Mit etwa 150 ms dauert sie jedoch etwa fünfmal länger als die Bildprüfung.

7.4. Ergebnisse und Entscheidungen

Ergebnis	Entscheidung für das Prüfband
Licht beeinflusst die Erkennung stark; die Consumer-Kamera ist bei Farbunterschieden nicht zuverlässig.	Das endgültige Prüfband prüft die Form und Unversehrtheit der Tabletten, nicht deren Farbe.
Zusätzliches Licht führt zu Pseudoausschuss.	Der Prüfbereich wird abgeschirmt, sodass hauptsächlich das Ringlicht wirkt.
Die ausfallsichere Prüflogik verhindert Schlupf.	Die Logik wird für das Prüfband übernommen.
Das Senden in die Cloud ist fünfmal langsamer als die Prüfung.	Entscheidung und Signalsäule laufen lokal; die Übertragung in die Cloud erfolgt asynchron und blockiert das Prüfband nie.
Ein Modell für Blister lässt sich nicht direkt auf einzelne Tabletten übertragen.	Das Prüfband wird neu mit eigenen Tablettenbildern trainiert.

7.4.1. Vergleich der beiden Ansätze

Zwischen den beiden Teilen des Projekts wechselt die KI-Methode: In der Machbarkeitsstudie wurde die Objekterkennung verwendet, da jeder Blister zehn Mulden enthält; das Prüfband nutzt Klassifizierung, da genau eine Tablette unter der Kamera liegt. Die folgende Tabelle vergleicht beide Ansätze und zeigt, warum Klassifizierung die einfachere und ausreichende Wahl für das Prüfband ist.

Aspekt	Objekterkennung (Machbarkeitsstudie)	Klassifizierung (Prüfband)
Frage	Welche Mulde enthält was?	In welchem Zustand befindet sich die Tablette?
Datenaufbereitung	Box um jede Mulde in Label Studio	Bilder sortiert in je einen Ordner pro Klasse
Aufwand pro Bild	Hoch (ca. 10 Boxen manuell zeichnen)	Sehr gering
Informationen	Position, Anzahl und Klasse jedes Objekts	Eine Klasse pro Bild
Geeignet für	Mehrere Objekte pro Bild	Genau ein Objekt pro Bild
Schwäche	Annotationsaufwand, mehr Regeln in der Prüflogik	Keine Positionsangabe; mehrere Objekte in einem Bild können nicht beurteilt werden

7.5. Fazit

Beide Teile sind machbar. Ein mit wenigen Bildern trainiertes Modell trifft in 97,8 % der Normalfälle die richtige Entscheidung, ohne Schlupf und in 24 ms; die Ergebnisse gelangen verschlüsselt in die Cloud. Die Grenzen liegen bei zusätzlichem Licht und bei der Farbprüfung; die Entscheidungen in 7.4 berücksichtigen beides. Mit dem Bau des Prüfbands konnte begonnen werden.

8. Umsetzung

Dieses Kapitel folgt der Reihenfolge der Arbeiten: gedruckte Teile, Testaufbau und Inbetriebnahme, mechanische Montage, Software, KI-Modell, Cloud-Anbindung und Betrieb. Zudem werden die aufgetretenen Probleme und deren Lösungen dokumentiert.

8.1. Herstellung der gedruckten Teile

Alle Teile wurden in einer Serie gedruckt, einschliesslich Ersatzteilen für kleine und stark beanspruchte Teile, da der Zugang zu einem 3D-Drucker danach eingeschränkt war. Kleine Mängel wurden während der Montage nachbearbeitet oder durch Ersatzteile behoben.

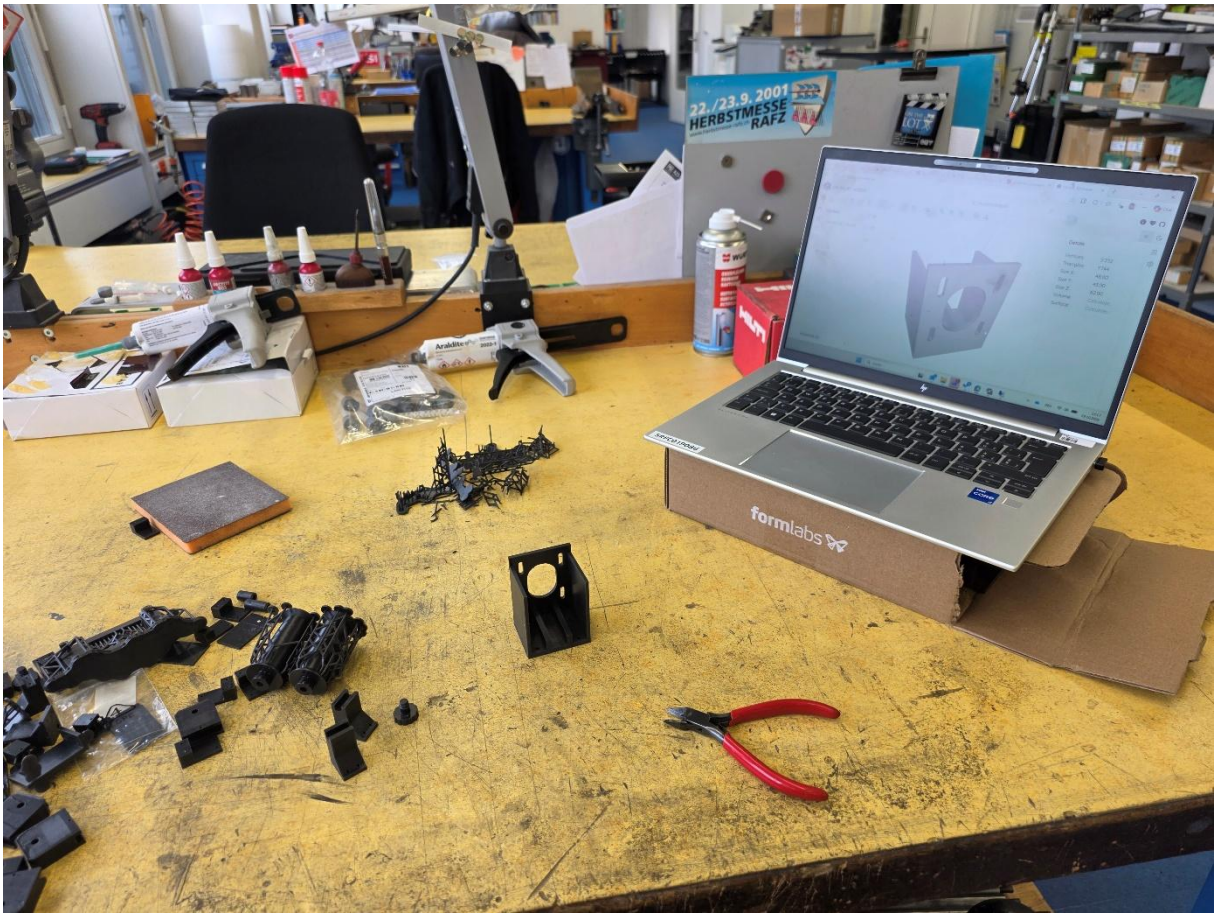


Abbildung 29: Nachbearbeitung 3D-Druckteile (eigene Aufnahme)

8.2. Testaufbau der Elektronik

Alle Komponenten wurden zunächst lose auf dem Tisch angeschlossen und einzeln getestet: Signalsäule, Sensor, Motor. Erst danach wurden sie gemeinsam getestet.

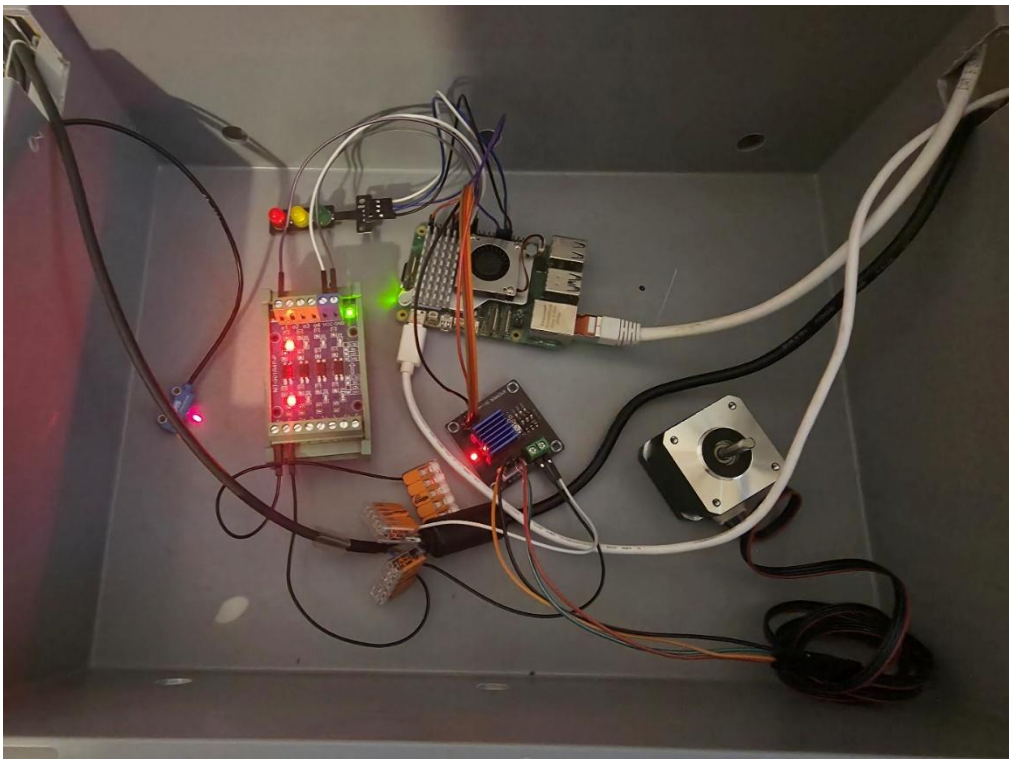


Abbildung 30: Testaufbau Elektronik (eigene Aufnahme)

Problem	Ursache	Lösung	Aufwand
Der TMC2209-Treiber und die Steuerplatine hatten keine Beschriftung.	Keine Dokumentation für die Klonplatinen	Pinbelegung durch Spannungs- und Durchgangsmessungen mit einem Multimeter ermittelt	etwa ½ Tag
Der Motor vibrierte, anstatt sich zu drehen.	Der Kontakt des roten Motorkabels war nicht vollständig eingesteckt.	Kontakt ausfindig machen und korrekt eingesteckt; der Motor lief daraufhin ruhig.	gering
Das Programm meldete beim Anhalten des Motors «bad PWM micros».	Die lgpio-Bibliothek akzeptiert keine Impulsfolge der Länge null.	Der Motor wird nun mit einem PWM-Befehl von 0 angehalten.	gering

Im Testaufbau wurden die Komponententests T02 und T03 bestanden: Jede Farbe der Signalsäule leuchtet korrekt, und der Sensor erkennt jede Tablette über den Optokoppler. Der Motor lief bei allen getesteten Geschwindigkeiten ruhig.

8.3. Inbetriebnahme des Raspberry Pi

Auf dem Raspberry Pi 5 läuft Raspberry Pi OS Lite (64-Bit) ohne Desktop. Hostname (raspberry-pi-sh), Benutzer (pruefband), WLAN und SSH wurden vor dem ersten Start im Raspberry Pi Imager eingerichtet. Der Zugriff auf den Pi erfolgt ausschliesslich über SSH; Dateien werden mit scp vom Edge-PC kopiert.

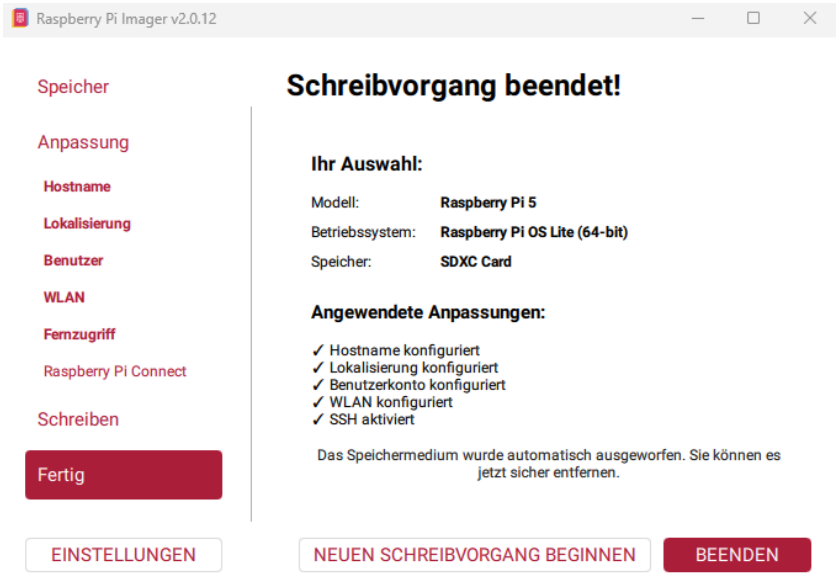


Abbildung 31: Raspberry Pi Image (eigene Aufnahme)

Der Router verwendet 192.168.1.REDACTED. Der Pi erhielt die feste Adresse 192.168.REDACTED, der Edge-PC 192.168.REDACTED.

REDACTED

Abbildung 32: DHCP statische IP-Adressen (eigene Aufnahme)

8.4. Inbetriebnahme des Edge-PCs und des MQTT-Brokers

Mosquitto 2.1.2 wurde als lokaler Broker auf dem Edge-PC installiert. Die Konfiguration erlaubt nur authentifizierte Clients (Listener 1883, kein anonymer Zugriff, Passwortdatei). Eine Firewall-Regel öffnet Port 1883 nur für das lokale Netzwerk.

Problem	Ursache	Lösung
Der Mosquitto-Windows-Dienst wurde nach dem Start ohne Fehlermeldung beendet.	Der Dienst startet ohne die Konfigurationsdatei; beim Start mit dieser konnte das SYSTEM-Konto die Passwortdatei nicht lesen («Unable to open pwfile»).	Dienst deaktiviert; Mosquitto wird als geplante Aufgabe beim Systemstart gestartet; Leseberechtigung für SYSTEM auf die Passwortdatei
paho-mqtt 2.x konnte nicht zusammen mit azure-iot-device installiert werden.	azure-iot-device 2.14 erfordert paho-mqtt unter 2.0.	paho-mqtt 1.6.1 auf dem Edge-PC; der Code ist mit beiden Versionen kompatibel

Ein Test mit einem falschen Passwort wurde vom Broker abgelehnt (N02). Der Edge-PC nutzt eine einzige Python-Umgebung für Training, Prüfung und Cloud; diese wurde aus der Machbarkeitsstudie übernommen, sodass nichts doppelt installiert werden musste.

8.5. Mechanische Montage und Justierung

Der Rahmen, die Rollen, der Motor mit dem Zahnriemen, das Gleitbett, das Förderband, die Sensorbrücke, die Kamera und die Signalsäule wurden Schritt für Schritt ohne detaillierten Montageplan zusammengebaut, gestützt auf das eigene Maschinenverständnis des Diplomanden. Einige geplante Teile wurden weggelassen, da das Prüfband auch ohne sie funktioniert, zum Beispiel die Führungen, die Rutsche und der Auffangbecher. Mehrere Probleme traten erst an der realen Maschine auf:

Problem	Lösung
Die Laufrichtung des Förderbands musste umgekehrt werden: Der Motor zieht das Band nun zu sich hin, der Sensor befindet sich am Anfang und die Kamera dahinter.	Richtung durch eine Einstellung in config.py geändert (8.6)
Die Kamera war mit einer einzigen zentralen Schraube befestigt und vibrierte, wodurch die Bilder unscharf wurden.	Zusätzlich mit Kabelbindern gesichert
Das aus einem Gummiband oder mit Stoff zugeschnittene Band «wanderte» seitlich ab, da die Enden nicht exakt gerade miteinander verbunden werden konnten. Verbindungen mit Gummi und mit Stoff versagten.	Band aus Gewebeband hergestellt. Das Band läuft nun gerade
Der Sensor erkannte beim ersten Winkel nicht jede Tablette zuverlässig.	Sensor justiert, neue Halterungen für andere Winkel entworfen

8.6. Software des Raspberry Pi 5

Die folgende Abbildung zeigt die Dateien auf beiden Geräten und ihre gegenseitigen Abhängigkeiten.

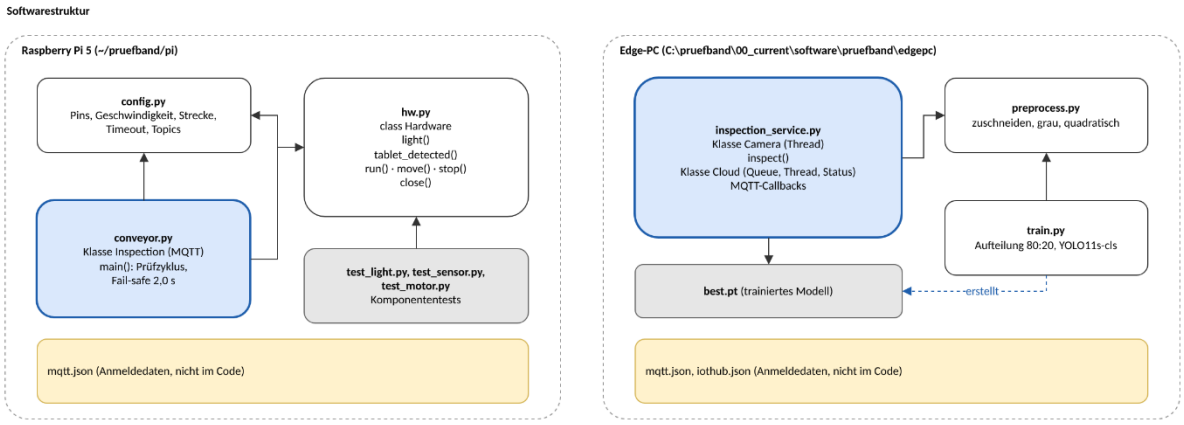


Abbildung 33: Softwarestruktur Raspberry Pi und Edge-PC (mit KI generierte Darstellung, draw.io)

Datei	Inhalt
config.py	Alle Pins und Einstellungen an einem Ort
hw.py	Klasse Hardware: Signalsäule, Sensor, Motor
conveyor.py	Prüfzyklus mit MQTT und Ausfallsicherheit

config.py

```
# Pins (BCM)
RED, YELLOW, GREEN = 5, 6, 13
SENSOR = 17
STEP, DIR, EN = 18, 23, 24

# Drive
STEPS_PER_MM = 200 * 8 * 2 / (3.1416 * 20)
DIR_FORWARD = 1
SPEED = 800
PULSE_US = 20

# Inspection
DISTANCE_TO_CAMERA_MM = 52
TIMEOUT_S = 2.0
SHOW_RESULT_S = 1.0

# MQTT
MQTT_CONFIG = "~/pruefband/config/mqtt.json"
TOPIC_REQUEST = "pruefband/request"
TOPIC_RESULT = "pruefband/result"
```

Der Antrieb rechnet Millimeter in Schritte um: 200 Vollschritte × 8 Mikroschritte × Untersetzung 2 pro Umdrehung der Rolle mit Ø 20 mm ergeben etwa 50,9 Schritte pro mm. Bei 800 Schritten pro Sekunde bewegt sich das Band mit etwa 15,7 mm pro Sekunde. Der Abstand vom Sensor zur Mitte der Kamera wurde während der Montage mehrmals angepasst und gemessen (70, dann 60, schlussendlich 52 mm) und beträgt nun 2'648 Schritte.

Methode (hw.py)

```
    def move(self, steps, speed=c.SPEED):
        self.stop()
        self.run(speed, steps)
        time.sleep(steps / speed + 0.1)
        self.stop()
```

Die Motorimpulse werden von lgpio im Hardware-Timing erzeugt. Nach einer Bewegung wartet das Programm die berechnete Laufzeit ab und stoppt dann explizit; die Busy-Abfrage von lgpio war nicht zuverlässig genug.

Prüfzyklus (conveyor.py)

```
def main():
    hw = Hardware()
    inspection = Inspection()
    count, times = 0, []
    try:
        hw.enable()
        print("Conveyor running (Ctrl+C to stop)")
        while True:
            hw.run()

            # Sensor detection
            while not hw.tablet_detected():
                time.sleep(0.001)
            hw.move(round(c.DISTANCE_TO_CAMERA_MM * c.STEPS_PER_MM))
            count += 1
            hw.light(yellow=True)

            # Inspection request
            t0 = time.perf_counter()
            result = inspection.request(f"{int(time.time())}-{count:04d}")
            ms = (time.perf_counter() - t0) * 1000
            if result is None:
                result, info = "NIO", f"NIO - {c.TIMEOUT_S} s no answer (fail-
safe)"
            else:
                times.append(ms)
                info = f"{result} ({ms:.0f} ms)"
```

8.7. Prüfsoftware auf dem Edge-PC

Datei	Inhalt
inspection_service.py	Kamera, Prüfung, Cloud, MQTT
preprocess.py	Gemeinsame Vorverarbeitung für Training und Prüfung
train.py	Aufteilung des Datensatzes und Training

Der Prüfdienst besteht aus drei Teilen:

- Ein separater Thread liest die **Kamera** kontinuierlich im MJPG-Format aus, sodass sich keine Bilder im Puffer stauen. Bei einer Anfrage wartet der Dienst auf zwei nach der Anfrage aufgenommene Bilder und verwendet das zweite davon.
- Bei der **Prüfung** selbst wird das Bild vorverarbeitet und klassifiziert. Unterhalb einer Konfidenz von 0,80 lautet das Ergebnis «NIO - uncertain»; bei einer Ausnahme (Exception) «NIO - error».
- Jedes Ergebnis wird in eine Queue gestellt; ein separater Thread sendet es an den IoT Hub. Bei einer Internetunterbrechung erfolgt alle 5 Sekunden ein erneuter Versuch. Beim Start und beim Stopp des Dienstes wird eine Statusmeldung (started, stopped) als Betriebsdaten gesendet.

Klasse Kamera (inspection_service.py)

```
# Camera
class Camera:
    def __init__(self):
        self.cam = cv2.VideoCapture(CAMERA, cv2.CAP_DSHOW)
        self.cam.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*"MJPG"))
        self.cam.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)
        self.cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)
        if not self.cam.isOpened():
            raise RuntimeError(f"Camera {CAMERA} not available")
        self.frame, self.count = None, 0
        self.new_frame = threading.Condition()
        threading.Thread(target=self._read_loop, daemon=True).start()

    def _read_loop(self):
        while True:
            ok, frame = self.cam.read()
            if ok:
                with self.new_frame:
                    self.frame, self.count = frame, self.count + 1
                    self.new_frame.notify_all()

    def capture(self):
        with self.new_frame:
            target = self.count + 2
```

```
            if not self.new_frame.wait_for(lambda: self.count >= target,
timeout=1.0):
                raise RuntimeError("No image from camera")
            return self.frame.copy()

    def release(self):
        self.cam.release()
```

Klasse Cloud (inspection_service.py)

```
# Cloud
class Cloud:
    def __init__(self):
        cfg = json.loads(IOTHUB_CONFIG.read_text(encoding="utf-8"))
        self.client =
IoTHubDeviceClient.create_from_connection_string(cfg["connection_string"])
        self.outbox = queue.Queue()
        threading.Thread(target=self._send_loop, daemon=True).start()

    def send(self, data):
        self.outbox.put(data)

    def status(self, state):
        self.send({
            "schema": "status.v1",
            "id": "edgepc-01",
            "timestamp": datetime.now(timezone.utc).isoformat(),
            "result": state,
            "confidence": 0.0,
            "inspection_ms": 0.0,
        })

    def _send_loop(self):
        while True:
            message = Message(json.dumps(self.outbox.get()),
content_encoding="utf-8",
                              content_type="application/json")
            while True:
                try:
                    self.client.send_message(message)
                    self.outbox.task_done()
                    break
                except Exception:
                    print(f"Cloud offline - retry in 5 s ({self.outbox.qsize()
+ 1} waiting)")
                    time.sleep(5)

    def close(self, timeout_s=5):
        end = time.time() + timeout_s
```

```
        while self.outbox.unfinished_tasks and time.time() < end:
            time.sleep(0.1)
        self.client.shutdown()
```

Das Bild wird vor der Prüfung gespeichert, sodass jede Anfrage ein Bild hinterlässt, auch wenn die Prüfung fehlschlägt. Die Cloud-Nachricht wird erst nach der Antwort an den Pi in die Queue gestellt, sodass die Entscheidung nie auf die Internetverbindung wartet. Ein Modus mit einem Klassennamen als Argument (zum Beispiel «python inspection_service.py nio_broken») sammelt Bilder für den Datensatz.

8.8. Best Practices im Code

Praxis	**Umsetzung**
Konfiguration getrennt vom Code	config.py auf dem Pi; Pfade (relativ zum Projektordner) und Schwellenwerte als Konstanten am Anfang jeder Datei
Keine Anmeldedaten im Code	mqtt.json und iothub.json, nur Vorlagen im Code-Ordner
Kapselung (Encapsulation)	Klasse Hardware auf dem Pi; Klassen Camera und Cloud auf dem Edge-PC
Ausfallsicherheit durch Timeout	Keine Antwort innerhalb von 2,0 s führt zu NIO
Correlation ID	Verknüpft Anfrage, Ergebnis, Bild und Cloud-Ereignis
Zustellung mindestens einmal (QoS 1, at least once)	MQTT QoS 1
Graceful Shutdown	finally-Blöcke: Motor gestoppt, Treiber deaktiviert, Leuchten aus, Cloud-Queue geleert
Eine Vorverarbeitungsfunktion	preprocess.py wird für Training und Prüfung verwendet
Entkoppelte Cloud	Queue und Thread; die Prüfung wartet nie auf das Internet
Feste Versionen	Ultralytics 8.4.172, paho-mqtt 1.6.1, requirements.txt

8.9. Bilddatensatz und Training des KI-Modells

8.9.1. Klassen

Basierend auf der Machbarkeitsstudie prüft das endgültige Prüfband nur die Form, nicht die Farbe. Es werden drei Klassen unterschieden: io (gut), nio_broken (gebrochen oder abgesplittert) und nio_wrong (anderer Tablettentyp oder Fremdkörper wie Schrauben oder Maschinenteile). Die falschen Tabletten unterscheiden sich immer in der Form, niemals nur in der Farbe.

8.9.2. Erfassung im realen Prüfzyklus

Die Bilder wurden direkt auf dem Prüfband aufgenommen. Im Erfassungsmodus (8.7) speichert der Prüfdienst jedes Bild im Ordner der angegebenen Klasse und gibt stets IO zurück, sodass das Band weiterläuft. Die Tablette befindet sich somit an derselben Position und unter denselben Lichtverhältnissen wie später im Betrieb. Pro Klasse liefen etwa 50 Tabletten durch, in leicht unterschiedlichen Positionen aufgelegt.

8.9.3. Probleme mit den Bildern

Beobachtung	**Ursache**	**Lösung**
Die Bilder waren in Laufrichtung des Bandes unscharf.	Die Bilder stammten aus dem Kamerapuffer, während sich das Band noch bewegte; in Full HD lieferte die Kamera unter DirectShow nur wenige Bilder pro Sekunde.	MJPG-Format (30 Bilder pro Sekunde) und ein Kamera-Thread (8.7)
Das erste Bild jedes Durchlaufs zeigte ein leeres Förderband, und alle Ergebnisse waren um eine Tablette verschoben.	Windows puffert Kamerabilder; das Programm erhielt ein altes Bild aus der Zeit vor der Anfrage.	Kamera-Thread, der kontinuierlich liest; es werden nur Bilder verwendet, die nach der Anfrage aufgenommen wurden

8.9.4. Vorverarbeitung

Jedes Bild wird auf die Breite des Förderbands (x = 480 bis 1'480 Pixel) bei voller Höhe zugeschnitten, in Graustufen umgewandelt und auf ein Quadrat von 1'080 × 1'080 Pixel aufgefüllt. Durch die Graustufen wird die Farbe entfernt, die sich in der Machbarkeitsstudie als unzuverlässig erwiesen hatte.

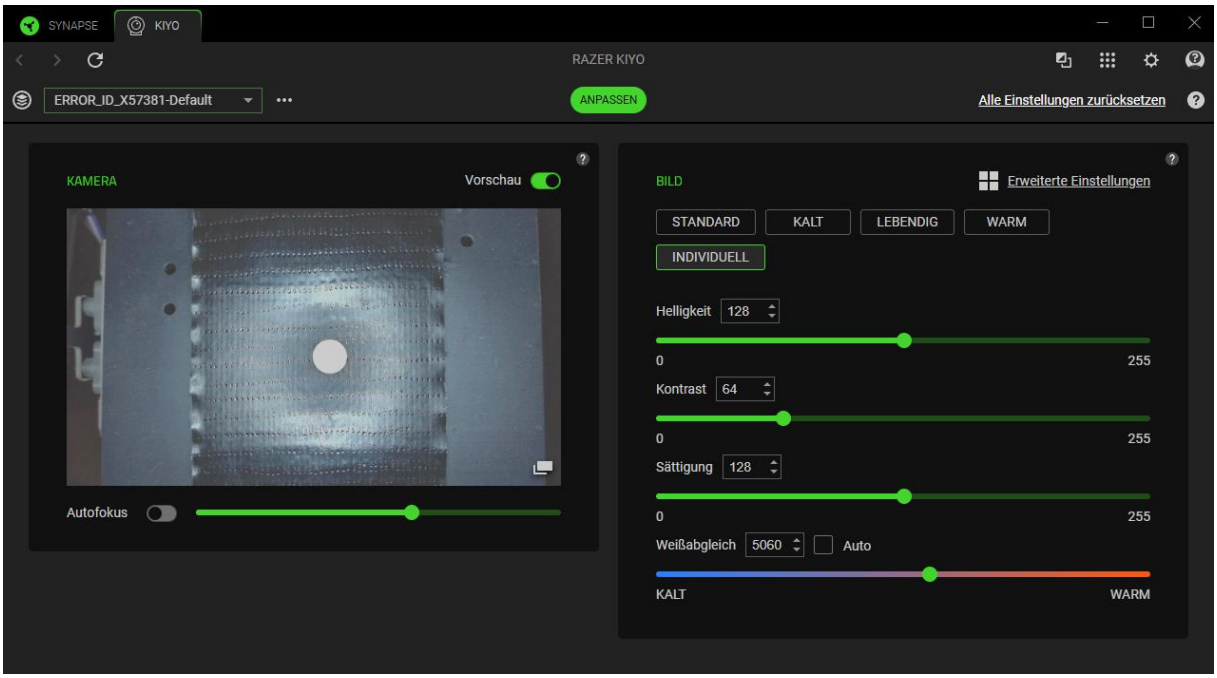


Abbildung 34: Unbearbeitete Ansicht RAZER (eigene Aufnahme)

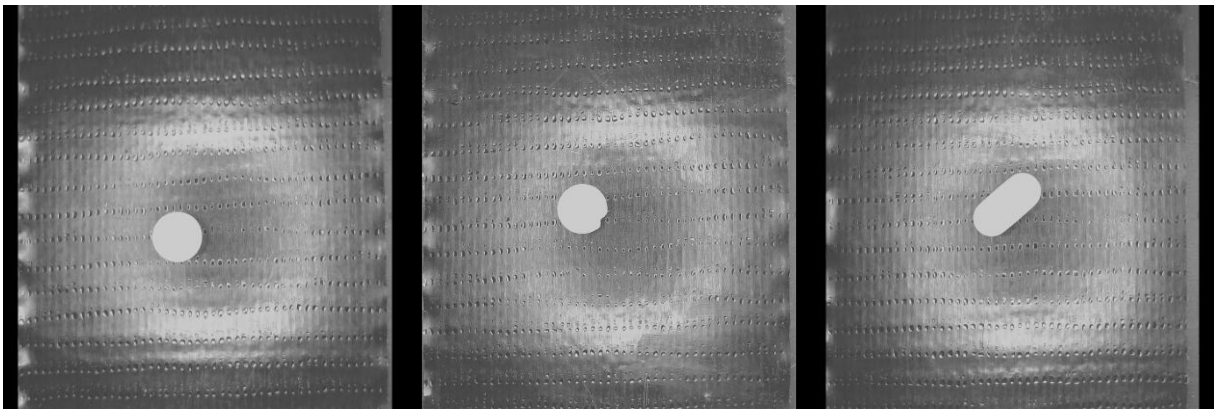


Abbildung 35: Bearbeitete Ansicht (eigene Aufnahme)

8.9.5. Training

Einstellung	Wert	Begründung
Modell	YOLO11s-cls, vortrainiert	Gleiche Grösse wie in der Machbarkeitsstudie (YOLO11s)
Bildgrösse	640 Pixel	Eine abgesplitterte Kante bleibt sichtbar
Aufteilung	80:20, stratifiziert, Seed 42	Reproduzierbar
Epochen, Batch	50, 16	Ausreichend Lernschritte für etwa 130 Bilder
Augmentierung	Helligkeit und Kontrast ±40 %, horizontale und vertikale Spiegelung	Robustheit gegenüber Licht und Position
Deaktiviert	Farbton, Sättigung, zufälliges Zuschneiden, Löschen (Erasing)	Farbe ist kein Merkmal; eine angeschnittene gute Tablette sähe gebrochen aus

Ergebnis	Wert
Bilder io / nio_broken / nio_wrong	50 / 64 / 51
Training / Validierung	132 / 33 Bilder
Trainingszeit	3,5 Minuten (50 Epochen)
Top-1 Accuracy bei der Validierung	1,0, stabil ab Epoche 31
KI-Prüfzeit	5,7 ms pro Bild
Aus dem vortrainierten Modell übernommene Gewichte	234 von 236 (nur die Ausgabeschicht wurde ersetzt: «Overriding nc=80 with nc=3» in der Trainingsausgabe)

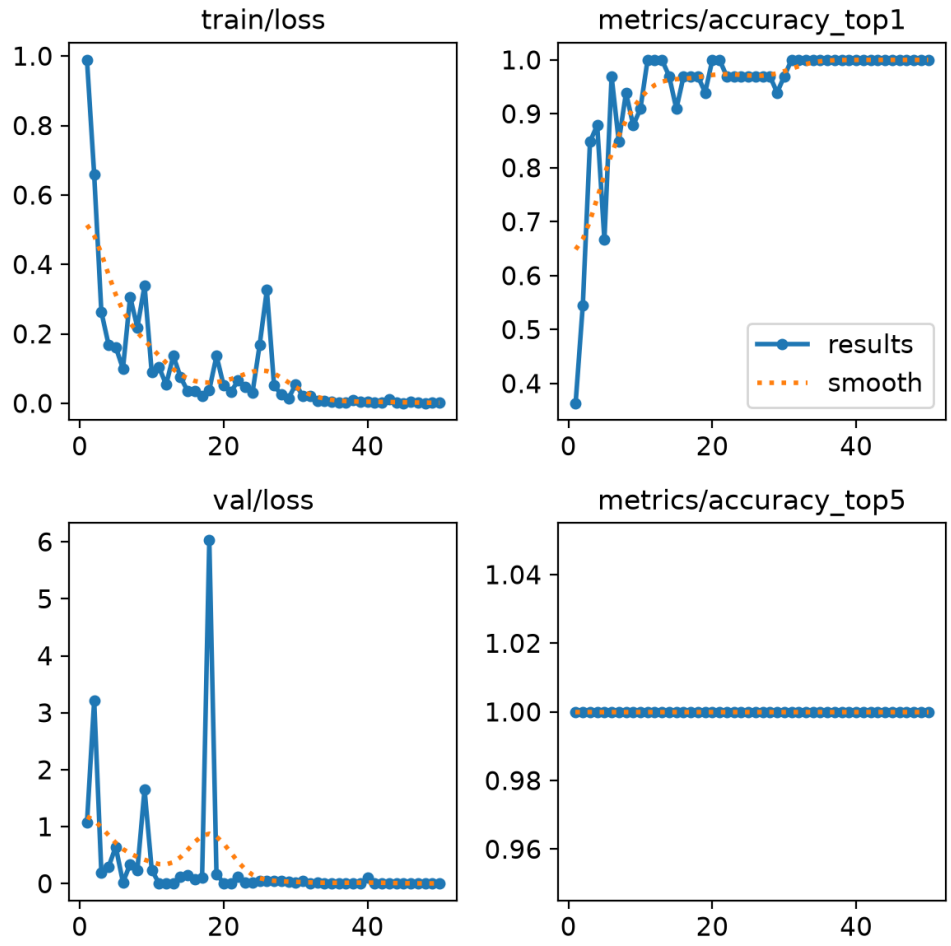


Abbildung 36: Resultat Lernkurve Prüfband (eigene Aufnahme aus Ultralytics)

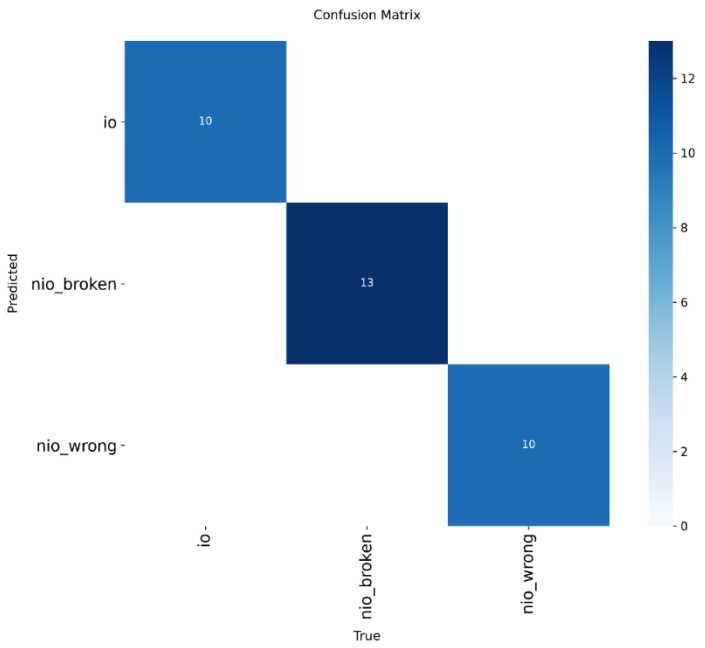


Abbildung 37: Konfusionsmatrix Prüfband (eigene Aufnahme aus Ultralytics)

In den ersten Epochen schwankt die Genauigkeit stark, da ein falsch klassifiziertes Bild das Ergebnis um 3 % verändert. Eine Top-1 Accuracy von 1,0 bei 33 Validierungsbildern ist ein gutes Zeichen, aber kein Beweis: Das Modell hat bereits ähnliche Bilder gesehen, teilweise von denselben Tabletten. Aussagekräftig ist erst der Systemtest auf dem laufenden Band (Kapitel 9).

Trainingsausgabe (Auszug aus dem Terminal)

```
(venv) PS C:\pruefband\software\edgepc> C:\pruefband\venv\Scripts\python.exe
train.py
io: 40 train, 10 val
nio_broken: 51 train, 13 val
nio_wrong: 41 train, 10 val
...
YOLO11s-cls summary: 86 layers, 5,446,851 parameters, 5,446,851 gradients,
12.3 GFLOPs
Transferred 234/236 items from pretrained weights
...
train: Scanning C:\pruefband\images\training\train... 132 images, 0 corrupt:
100% ━━━━━━━━━━━━ 132/132 3.1Kit/s 0.0s
...
Starting training for 50 epochs...

      Epoch    GPU_mem       loss  Instances       Size
       1/50       3.1G     0.9891          4        640: 100% ━━━━━━━━━━━━ 9/9
3.9s/it 35.0s
               classes   top1_acc   top5_acc: 100% ━━━━━━━━━━━━ 2/2 5.0s/it
10.0s
                   all      0.364          1


...
      Epoch    GPU_mem       loss  Instances       Size
      50/50      3.98G   0.001334          4        640: 100% ━━━━━━━━━━━━ 9/9
8.8it/s 1.0s
               classes   top1_acc   top5_acc: 100% ━━━━━━━━━━━━ 2/2 14.3it/s
0.1s
                   all          1          1

50 epochs completed in 0.059 hours.
...
Ultralytics 8.4.172  Python-3.12.10 torch-2.6.0+cu124 CUDA:0 (NVIDIA GeForce
RTX 3060 Ti, 8192MiB)
...
Speed: 0.6ms preprocess, 5.7ms inference, 0.0ms loss, 0.0ms postprocess per
image
```

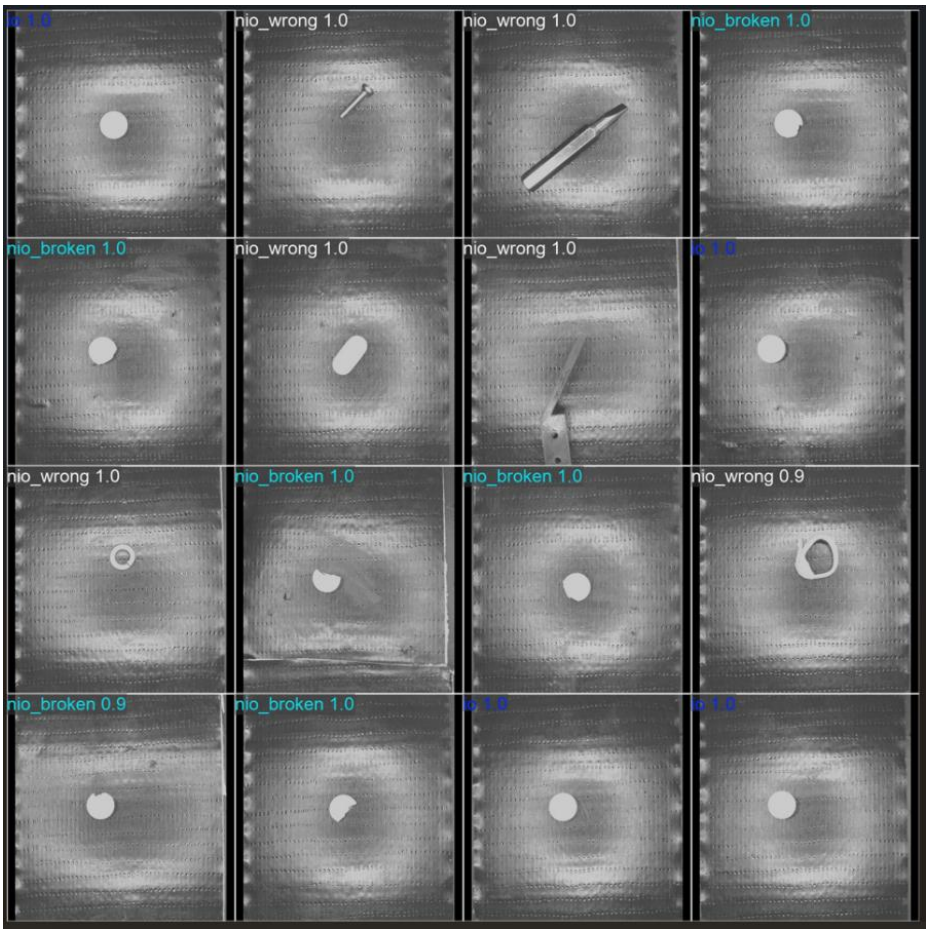


Abbildung 38: Auswertungen Prüfband (eigene Aufnahme aus Ultralytics)

8.10. Cloud-Anbindung und Visualisierung

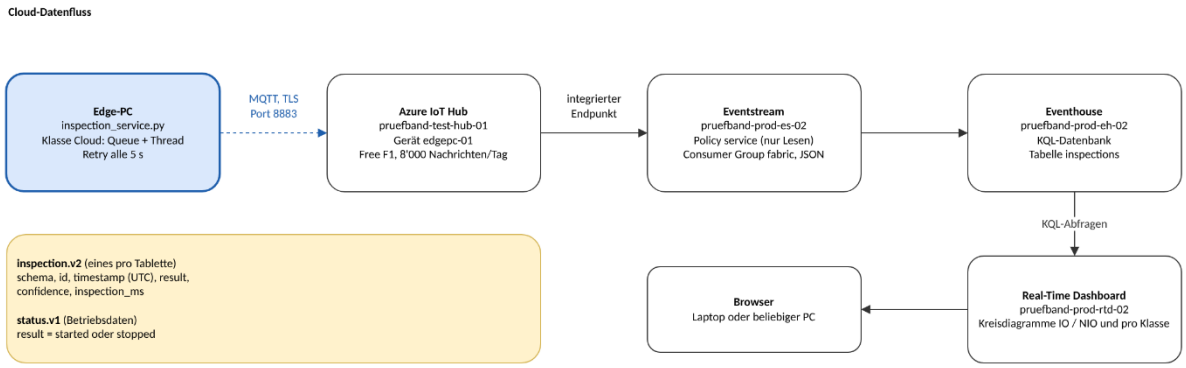


Abbildung 39: Cloud-Datenfluss vom Edge-PC zum Dashboard (mit KI generierte Darstellung, draw.io)

8.10.1. IoT Hub

Die Verbindungszeichenfolge (Connection String) des Geräts ist in iothub.json gespeichert. Ein erster Test mit 3 Tabletten erzeugte 3 Nachrichten im IoT Hub («Telemetry messages sent»), ein zweiter und dritter Test ergaben dann insgesamt 12 Tabletten und 12 Nachrichten.

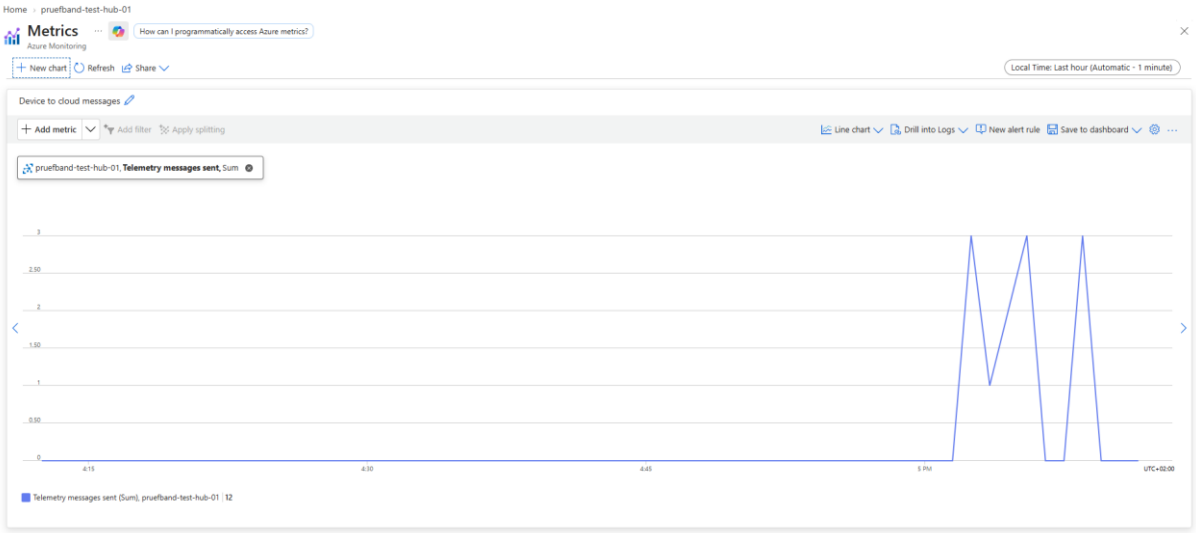


Abbildung 40: Messages im IoT Hub (eigene Aufnahme)

8.10.2. Microsoft Fabric

Der Weg zu einer funktionierenden Fabric-Umgebung erforderte mehrere Versuche:

Schritt	Problem	Lösung
Anmeldung	Fabric erfordert ein Arbeits- oder Schulkonto (Microsoft Entra ID).	Separater Benutzer für Fabric in einem eigenen Microsoft-Entra-Tenant
Testversion	«Unable to start trial»; Testversion nicht als Workspace-Typ auswählbar	Fabric-Administratorrolle zugewiesen; vorübergehend eine kostenpflichtige F2-Kapazität in Azure
Kostenpflichtige Kapazität	Eine Kapazität erfordert eine Rolle im Azure-Abonnement; F2 kostet im Dauerbetrieb etwa USD 262 bis 336 pro Monat	Contributor-Rolle im Abonnement für den Fabric-Benutzer; Kapazität bei Nichtnutzung pausiert
Erneut Testversion	Nach mehreren Aktivierungen und neuen Anmeldungen wurde die Testversion in einem neuen Workspace verfügbar	Workspace pruefband-prod-ws-02 auf der Testversion; F2-Kapazität gelöscht
Eventstream	Daten nur in der Vorschau, keine Tabelle im Eventhouse	Ziel (Destination) hinzugefügt und veröffentlicht; die Tabelle wurde nach weiteren Ereignissen erstellt

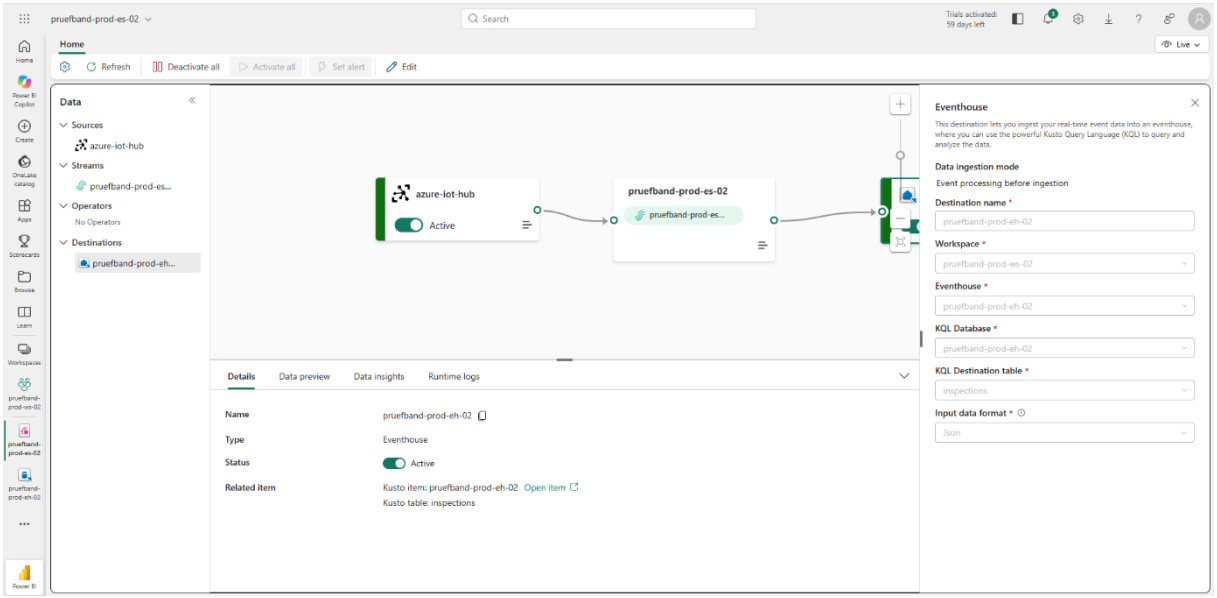


Abbildung 41: Microsoft Fabric Eventstream (eigene Aufnahme)

Der Eventstream liest den IoT Hub mit der Shared Access Policy «service» (nur Lesen) und seiner eigenen Consumer Group «fabric» im JSON-Format aus und schreibt in die Tabelle inspections.

Das Real-Time Dashboard pruefband-prod-rtd-02 zeigt zwei Kreisdiagramme: IO im Vergleich zu NIO sowie die Anzahl pro Ergebnisklasse. Jede Abfrage filtert nach dem Schema, sodass Statusmeldungen nicht als Prüfungen gezählt werden. Das Dashboard wird automatisch aktualisiert und kann in jedem Browser geöffnet werden.

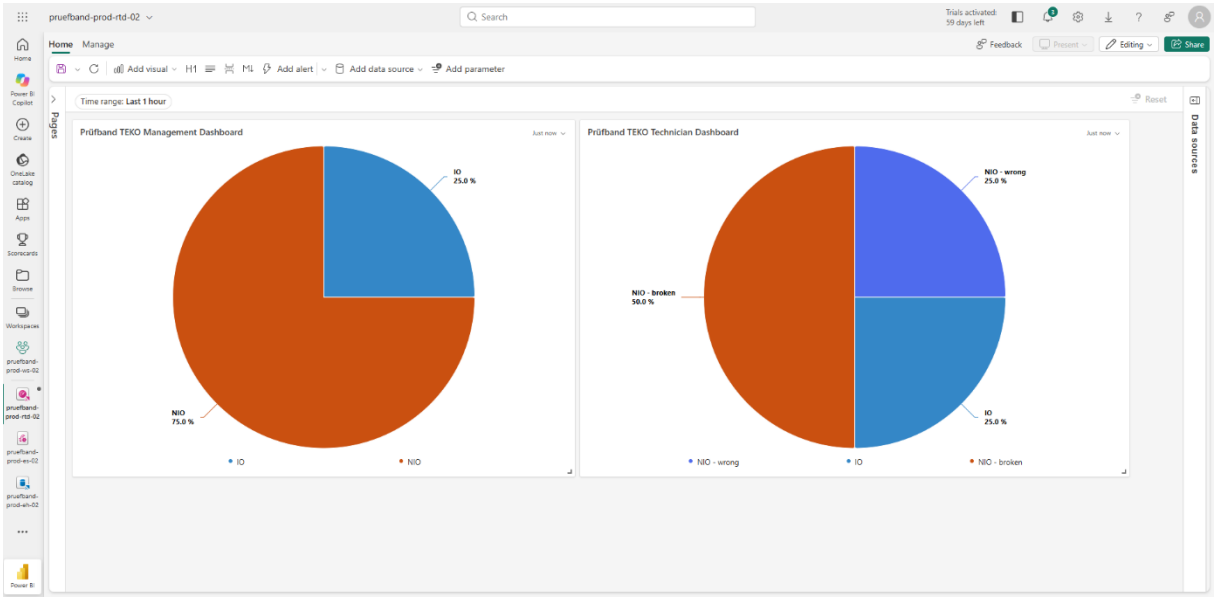


Abbildung 42: Cloud Dashboard (eigene Darstellung)

8.11. Inbetriebnahme und Betrieb

8.11.1. Start

Den Edge-PC einschalten (der Broker startet automatisch) und den Prüfdienst starten. Den Raspberry Pi einschalten, danach die 12-V-Versorgung anschliessen und das Steuerungsprogramm über SSH starten.

8.11.2. Stopp

Beide Programme mit Strg+C beenden, die 12-V-Versorgung trennen, den Pi mit «sudo shutdown -h now» herunterfahren und warten, bis die grüne LED aufhört zu blinken.

8.11.3. Beobachtung

Wird die SSH-Verbindung unterbrochen, läuft das Steuerungsprogramm weiter und das Band hält nicht an. Ein Neustart schlägt dann mit der Meldung «GPIO busy» fehl. Das laufende Programm wird mit «pkill -INT -f conveyor.py» sauber beendet. In einem realen System würde das Steuerungsprogramm als Systemdienst laufen und durch einen Not-Aus-Schalter abgesichert sein (13.4).

9. Tests

In diesem Kapitel wird das fertige Prüfband getestet. Die Tests der Machbarkeitsstudie sind in Kapitel 7 dokumentiert.

9.1. Testkonzept

9.1.1. Testziele

Die Tests belegen, dass die Muss-Anforderungen (2.2) erfüllt sind und dass die Erfolgskriterien EK2 bis EK5 mit den Zielwerten Z1 bis Z5 (2.1) erreicht werden.

9.1.2. Teststrategie und Testebenen

Die Tests reichen vom Einzelteil bis zum Gesamtsystem. In jedem Testfall wird nur ein Faktor verändert, sodass jede Ursache eindeutig identifiziert werden kann.

Ebene	Objekt	Zeitpunkt
Komponententest	Einzelteile und Programme	Vorwiegend im Testaufbau, vor der Montage
Integrationstest	Raspberry Pi, Feldgeräte, Edge-PC und Broker zusammen	Im Testaufbau und auf dem montierten Prüfband
Systemtest	Montiertes Prüfband mit echten Tabletten und Cloud	Nach der Montage

9.1.3. Testumgebung

Punkt	Wert
Ort	Wohnung des Diplomanden
Beleuchtung	Raumlicht und das Ringlicht der Kamera
Objekte	Gute runde Tabletten; gebrochene und abgesplitterte Tabletten desselben Typs; andere Objekte und Tablettentypen (länglich, grössere runde)
Modell	YOLO11s-cls

Für die Auswertung wurden die Ergebnisse aus den vorhandenen Ausgaben gelesen: der Konsolenausgabe des Pi und des Edge-PCs sowie dem Dashboard in der Cloud.

9.1.4. Fehler und Kriterien

Fehler werden eingestuft als kritisch (eine Muss-Anforderung ist nicht erfüllt, oder Schlupf), mittel (die Funktion ist eingeschränkt, es gibt jedoch eine Umgehungslösung) oder gering (kosmetisch). Ein kritischer Fehler, die Gefahr einer Beschädigung der Hardware oder eine unkontrollierte Bewegung des Förderbands führen zum Abbruch des Tests. Die Tests beginnen, wenn alle Komponententests bestanden sind, der Broker läuft, das Modell trainiert ist und die Cloud-Anbindung funktioniert; sie enden, wenn alle Testfälle durchgeführt und dokumentiert sind.

9.2. Testfälle

ID	Ebene	Testfall	Erwartetes Ergebnis	Referenz
T01	Komponente	Das Förderband transportiert eine Tablette vom Sensor unter die Kamera	Die Tablette befindet sich im Bildbereich	F01, F03
T02	Komponente	Sensor mit und ohne Tablette	Signal nur bei vorhandener Tablette	F02
T03	Komponente	Die Farben der Signalsäule nacheinander schalten	Die richtige Farbe leuchtet auf	F06
T04	Komponente	Prüfung eines Bildes	KI-Entscheidung in höchstens 100 ms	Z3
T05	Integration	Der Sensor löst die Prüfung aus, das Ergebnis schaltet die Signalsäule	Die Signalsäule zeigt das Ergebnis an	F02 bis F06
T06	Integration	Internetverbindung während des Betriebs unterbrechen	Das Prüfband prüft und signalisiert wie gewohnt	F08
T07	System	Gute Tabletten	Mindestens 95 % IO	Z2
T08	System	Gebrochene, abgesplitterte und falsche Tabletten	0 Schlupf	Z1, F05
T09	System	Ergebnisse in der Cloud	Die Anzahl stimmt mit dem Prüfband überein	F07, Z4
T10	System	Dauerbetrieb ohne Eingriff	Keine Störung	Z5
T11	Integration (zusätzlich)	Unterbrechung der Verbindung zwischen Pi und Edge-PC	Alle Tabletten NIO, automatische Wiederaufnahme	F05

9.3. Testdurchführung und Ergebnisse

9.3.1. Finale Komponenten- und Integrationstests

ID	Datum	Ergebnis	Bestanden
T01	20.10.2026	Die Tablette stoppt unter der Kamera, nachdem der Abstand auf 52 mm eingestellt wurde.	Ja
T02	20.10.2026	Jede Tablette wird vom SICK-Sensor über den Optokoppler erkannt.	Ja
T03	20.10.2026	Rot, Gelb und Grün leuchten nacheinander auf.	Ja
T04	21.10.2026	5,7 ms pro Bild (Ultralytics-Validierung).	Ja
T05	21.10.2026	Der Zyklus läuft; die Signalsäule zeigt das Ergebnis des Edge-PCs an. Ein Timeout von 2,0 s führt zu Rot.	Ja
T06	21.10.2026	Mobilfunkverbindung am Router ausgeschaltet: Prüfung und Signalsäule funktionierten wie gewohnt.	Ja
T11 (zusätzlich)	21.10.2026	Netzwerkkabel des Pi abgezogen: alle Tabletten «NIO - 2.0 s no answer (fail-safe)»; nach dem Wiedereinstecken wurde der Betrieb ohne Neustart fortgesetzt.	Ja

Beim ersten Versuch von T06 wurde das Netzwerkkabel des Pi abgezogen und damit nicht nur das Internet, sondern auch das lokale Netzwerk unterbrochen. Dies wurde zum zusätzlichen Testfall T11, und T06 wurde wiederholt, indem am Router nur die Mobilfunkverbindung via Webinterface ausgeschaltet wurde.

9.3.2. Systemtest

Der Systemtest fand am 21.10.2026 statt, wobei 27 Tabletten aller drei Gruppen in gemischter Reihenfolge durchliefen. Die Referenz (Soll-Klasse) jeder Tablette wurde anschliessend durch eine Sichtprüfung aller gespeicherten Bilder durch den Diplomanden festgelegt. Alle Ergebnisse entsprachen den Erwartungen, sodass der Systemtest mit 27 Tabletten als ausreichend beurteilt und abgeschlossen wurde.

Eine Tablette war leicht abgesplittert und wurde mit einer Konfidenz von 65 % als «NIO - uncertain» aussortiert. Das Modell war unsicher, und die Tablette wurde aussortiert. Dies ist genau das beabsichtigte ausfallsichere Verhalten (F05).

9.3.3. Vollständigkeit in der Cloud (T09)

Der Pi zählte 27 Tabletten und das Dashboard zeigte ebenso 27 Prüfungen an. Z4 ist erfüllt.

9.3.4. Betriebsdaten (EK4)

Die Statusmeldungen «started» und «stopped» des Prüfdienstes sind im Eventhouse eingegangen.

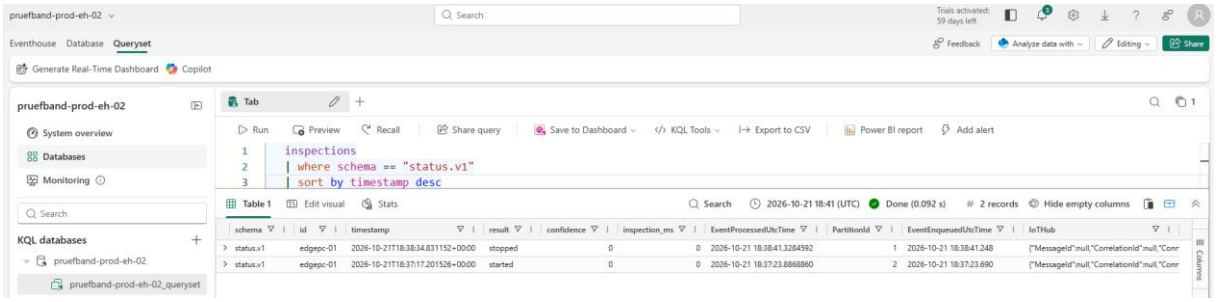


Abbildung 43: Abfrage der Statusmeldungen (eigene Aufnahme)

9.3.5. Dauerbetrieb (T10)

Der Dauerbetrieb wurde anhand der kumulierten Betriebszeit beurteilt: Das Prüfband lief während der Montage, Einstellung und Tests mehrere Stunden lang fehlerfrei; die Start- und Stoppzeiten werden durch die Statusmeldungen in der Cloud erfasst. Zudem lief der Raspberry Pi mehr als 24 Stunden lang problemlos (ohne das Steuerungsprogramm) und schaltete seinen Lüfter bei Bedarf selbstständig ein und wieder aus. Die Langzeitbeständigkeit von Motor und Förderband ist nicht Teil dieser Arbeit (2.4).

9.4. Gesamtbewertung

Alle Testfälle wurden bestanden. Keine defekte oder falsche Tablette wurde als gut bewertet, keine gute Tablette wurde aussortiert, und jedes Ergebnis gelangte in die Cloud. Das Prüfband arbeitet auch ohne Internetverbindung weiter und sortiert jede Tablette sicher aus, wenn der Edge-PC nicht erreichbar ist. Einschränkend ist zu beachten, dass die Testtabletten teilweise dieselben wie im Training waren und kein Langzeittest durchgeführt wurde.

10. Wirtschaftlichkeitsanalyse

Als Einzelanfertigung kostet das Prüfband etwa CHF 20’400 bis 23’900 und ist als Produkt nicht wirtschaftlich. Sein Wert liegt in den gewonnenen Erkenntnissen, vergleichbar mit einem Forschungs- und Entwicklungsprojekt. Als kostengünstiges Schulungskit könnte sich eine vereinfachte Version lohnen.

10.1. Annahmen

- Stundensatz CHF 100. Hierbei handelt es sich um einen geschätzten internen Kostensatz eines Unternehmens (Lohn, Sozialabgaben, Arbeitsplatz, Gemeinkosten), nicht um einen Verkaufspreis. Ein externer Satz würde bei etwa CHF 120 liegen und wird bewusst nicht herangezogen.
- Stunden, geschätzt anhand des Terminplans. 1,5 bis 2 Stunden pro Arbeits- oder Schultag, mehr als 8 Stunden pro Tag im Puffer (15. bis 26. Oktober 2026, 12 Tage à 8 Stunden), keine Arbeit während der drei Wochen mit hoher Arbeitsbelastung und Krankheit (3.7). Die Stunden werden daher als Spanne angegeben; der obere Wert ist das Maximum.
- Preise der gekauften Komponenten sind die tatsächlich bezahlten Preise, auf ganze Franken gerundet.
- Vorhandene Ausrüstung (Edge-PC, Raspberry Pi, Kamera) wird nicht verrechnet, ihr Wert wird jedoch angegeben, damit der Leser den Gesamtwert des Prototyps einschätzen kann.

10.2. Kosten des Prototyps

Phase	Stunden	Kosten in CHF
Analyse, Evaluation, Konzept, Entwurf	75 bis 100	7'500 bis 10'000
Machbarkeitsstudie	12 bis 16	1'200 bis 1'600
Testaufbau und Software	18 bis 24	1'800 bis 2'400
Montage, Training, Cloud, Tests (Puffer)	72	7'200
Fertigstellung der Dokumentation (Puffer)	24	2'400
Arbeitsaufwand insgesamt	201 bis 236	20'100 bis 23'600

Nicht eingerechnet sind die Projektinitiierung und die Themensuche (etwa 60 bis 80 Stunden): Sie gehören zur Diplomarbeit, nicht zur Entwicklung des Prototyps.

Material	Kosten in CHF
GT2-Set (Zahnriemen, Riemenscheiben)	14
Hebelklemmen (WAGO)	10
Schrauben	18
Kugellager	9
NEMA-17-Schrittmotor	8
Kabel	7
TMC2209-Treiber (2 Stück, einer als Ersatz)	10
Wellen (2 Stück)	7
Schrittmotortreiber-Steuerplatine	5
Optokoppler-Modul	15
Mini-Signalsäule	3
A4988-Treiber (Ersatz)	4
Sensor SICK WT2S-P231	20
TCRT5000 (Ersatzsensor)	3
12-V-Netzteil	10
Material für die 3D-gedruckten Teile	50
Aluminium-Profile MakerBeamXS	150
Total eingekauftes Material	**343**

Die Cloud kostete fast nichts: IoT Hub F1 und die Fabric-Testversion sind kostenlos, und die F2-Kapazität lief nur kurze Zeit (weniger als CHF 2).

Die vorhandenen Geräte sind mit ihrem Wert aufgeführt, damit der Gesamtwert des Prototyps geschätzt werden kann.

Vorhandene Ausrüstung	Aktueller Wert in CHF
Raspberry Pi 5, 8 GB, mit originalem Active Cooler	190
Edge-PC Dell XPS 8940 mit RTX 3060 Ti	850
Webcam Razer Kiyo	50
Gesamtwert der vorhandenen Ausrüstung	**1090**

Gesamtkosten des Prototyps: ca. CHF 20'443 bis 23'943 (Arbeitsaufwand und eingekauftes Material), wovon mehr als 98 % auf den Arbeitsaufwand entfallen. Unter Einbezug der vorhandenen Ausrüstung stellt der Prototyp einen Wert von ca. CHF 21’533 bis 25'033 dar.

10.3. Bewertung als Einzelanfertigung

Für ein einzelnes Gerät sind diese Kosten viel zu hoch. Das Material ist günstig, aber der Grossteil der Arbeitszeit floss in das Einarbeiten: maschinelles Lernen, IoT, Fabric sowie Troubleshooting. Ein zweites Prüfband würde deutlich weniger Zeit in Anspruch nehmen, da Konzept, Code und Druckdateien bereits vorliegen. Das Projekt sollte daher als Research & Development (R&D) betrachtet werden: Es hat Wissen, eine wiederverwendbare Softwarebasis und eine Liste dessen, was nicht funktioniert, hervorgebracht.

10.4. Serienprodukt

10.4.1. Nicht für die pharmazeutische Industrie

Eine echte pharmazeutische Prüfung erfordert eine GMP-Validierung, qualifizierte Industriekameras und Beleuchtung sowie eine dokumentierte Änderungskontrolle (Change Control) (2.4). Das Prüfband demonstriert diesen Anwendungsfall, ist jedoch nicht für die echte pharmazeutische Produktion geeignet.

10.4.2. Ein Schulungskit für Vision-AI im pharmazeutischen Bereich

Ein realistischerer Markt sind Personen, die Vision-AI anhand von Anwendungsfällen aus der pharmazeutischen Industrie erlernen möchten: Studierende, Schulen, Maker und Entwickler in Unternehmen. Das Kit orientiert sich eng am Anwendungsfall dieser Diplomarbeit, benötigt jedoch keine GMP-Validierung, da es zu Schulungszwecken und nicht für die Produktion genutzt wird. Ein fertiges, kostengünstiges Kit mit Förderband, Sensor, Signalsäule und dokumentierter Software könnte auf dem Markt Abnehmer finden.

10.4.3. Was sich bei einem Serienprodukt ändern würde

Es gibt viele Möglichkeiten, das Kit kostengünstig zu produzieren und marktreif anzubieten:

Prototyp	**Serienprodukt**
Raspberry Pi, Treiberplatine, Optokoppler und freie Verkabelung	Eine massgefertigte Leiterplatte (PCB) mit Mikrocontroller, Treiber und Eingängen, in grossen Stückzahlen produziert
Anpassbarer Rahmen aus teuren Aluminiumprofilen	Fester Rahmen aus Kunststoffguss
3D-gedruckte Teile	Standardteile und Spritzgussteile
Selbst hergestelltes Band aus Gewebeband	Gekauftes Endlos-Förderband (Normteil)
Consumer-Webcam	Industriekamera mit definierter Beleuchtung
Edge-PC des Entwicklers	Eigener PC des Kunden oder ein optionales Edge-KI-Modul
Ultralytics unter AGPL-3.0	Kommerzielle Lizenz oder ein Framework mit einer freizügigen Lizenz [32]
Von einer Person gepflegter Code	Gepflegte Codebasis mit Updates, Sicherheitspatches und Support

Der letzte Punkt wird oft unterschätzt: Betriebssysteme, Bibliotheken (wie der Konflikt zwischen paho-mqtt und azure-iot-device), Sicherheitsupdates, sich ändernde Cloud-Dienste (wie IoT Central, 5.8.2) und neue Modellversionen verursachen laufende Kosten, auch wenn keine neuen Geräte verkauft werden. Zudem benötigt ein Elektroprodukt eine Konformitätsbewertung und die Speicherpreise stellen derzeit ein Kostenrisiko dar (4.6).

10.4.4. Cloud für eine Markteinführung

Die gesamte Cloud-Infrastruktur dieser Arbeit wurde in einem Tenant erstellt. Für eine Markteinführung müsste erneut geprüft werden, ob es sich lohnt, alle Kits in einem zentralen Tenant zu betreiben, oder ob jeder Nutzer einen eigenen Tenant einrichten sollte.

[32] Ultralytics: YOLO11-Dokumentation (Training, Validierung, Lizenz), https://docs.ultralytics.com

10.4.5. Break-even

Die Einbeziehung der laufenden Softwarewartung würde auf einer sehr weitreichenden Annahme beruhen: Wie die Wartung organisiert werden kann und welche Kosten damit verbunden sind, müsste zuerst aufgebaut und getestet werden, weshalb dies in dieser Arbeit nicht bewertet wird. Die Berechnung ist daher bewusst einfach gehalten: Es wird nur die Hardware verkauft, und die vorhandene Software wird mitgeliefert, sodass Käufer die Vision-AI für pharmazeutische Produkte zu Schulungszwecken nutzen können. Alle Werte sind Annahmen.

Posten	**Annahme**
Entwicklungs- und Prototypkosten dieser Diplomarbeit (10.2)	CHF 20'443 bis 23'943
Zusätzliche Weiterentwicklung vom aktuellen Stand bis zur Marktreife (150 Stunden)	CHF 15'000
Zu deckende Kosten	CHF 35’443 bis 38’943
Stückkosten (Hardware)	CHF 90
Verpackung und Vertrieb pro Einheit	CHF 20
Verkaufspreis ohne Edge-Computer	CHF 199
Deckungsbeitrag pro Einheit	CHF 89
Break-even	bei etwa 400 bis 440 Einheiten

Die 150 Stunden kommen zu den Arbeitsstunden dieser Diplomarbeit hinzu. Sie umfassen die Weiterentwicklung vom aktuellen Stand des Prüfbands zu einem marktgerechten Produkt, beispielsweise die vereinfachte Mechanik und eine Dokumentation für Käufer. Unter diesen Annahmen sind die Kosten nach etwa 400 bis 440 verkauften Kits gedeckt. Ob dieses Volumen realistisch ist, würde eine Marktanalyse erfordern, die jedoch nicht Teil dieser Diplomarbeit ist.

11. Quellen- und Literaturverzeichnis

Alle Quellen sind an den entsprechenden Orten in der Fussnote aufgeführt.:

Allegro MicroSystems: Datenblatt A4988.
Analog Devices: Datenblatt TMC2209, Rev. 1.08.
Bendle, S.: GPU prices keep surging in 2026. Tom's Hardware, 03.08.2026. https://www.tomshardware.com/pc-components/gpus/gpu-prices-for-current-gen-nvidia-and-amd-price-increases-why-have-the-prices-not-dropped-and-can-you-still-buy-a-cheap-gpu
Bloomberg: Why the AI Boom is Making Everything More Expensive (interaktive Grafik, Datenstand 01.03.2026). https://www.bloomberg.com/graphics/2026-ai-boom-memory-chip-shortage/
Cerullo, M.: Buying a computer? AI is making it more expensive. CBS News, 30.04.2026. https://www.cbsnews.com/news/computer-price-memory-chip-shortage-artificial-intelligence/
Ferreira, B.: Data centers will consume 70 percent of memory chips made in 2026. Tom's Hardware, 18.01.2026. https://www.tomshardware.com/pc-components/ram/data-centers-will-consume-70-percent-of-memory-chips-made-in-2026-supply-shortfall-will-cause-the-chip-shortage-to-spread-to-other-segments
Label Studio: Label Studio Documentation. https://labelstud.io/guide
Lago, P.: Wegen KI: Handys und Laptops werden bis zu 20 Prozent teurer. SRF News, 04.04.2026. https://www.srf.ch/news/wirtschaft/chip-krise-wegen-ki-handys-und-laptops-werden-bis-zu-20-prozent-teurer
Microsoft: Azure IoT Hub pricing. https://azure.microsoft.com/en-us/pricing/details/iot-hub/
Microsoft Learn: Add Azure IoT Hub source to an eventstream. https://learn.microsoft.com/en-in/fabric/real-time-intelligence/event-streams/add-source-azure-iot-hub
Microsoft Learn: Azure IoT Central is evolving. https://learn.microsoft.com/en-us/azure/iot-central/core/iot-central-evolution
Microsoft Learn: Cloud Adoption Framework, Define your naming convention. https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming
Microsoft Learn: Cost Management, Create and manage budgets. https://learn.microsoft.com/en-us/azure/cost-management-billing/costs/tutorial-acm-create-budgets
Microsoft Learn: Try Microsoft Fabric for free. https://learn.microsoft.com/en-us/fabric/fundamentals/fabric-trial
Miquel, T. (RTS): KI-Boom – IT-Kosten von Städten und Kantonen steigen stark. SRF News, 07.10.2026. https://www.srf.ch/news/dialog/ki-boom-it-kosten-von-staedten-und-kantonen-steigen-stark
NVIDIA: GeForce RTX 3060 family, specifications. https://www.nvidia.com/en-ph/geforce/graphics-cards/30-series/rtx-3060-3060ti/
NVIDIA: Tensor Cores. https://www.nvidia.com/en-eu/data-center/tensorcore/
NVIDIA: Train with mixed precision. https://docs.nvidia.com/deeplearning/performance/mixed-precision-training
OASIS: MQTT Version 5.0, OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html
Pololu: A4988 Stepper Motor Driver Carrier, Produktdokumentation.
Shilov, A.: Nvidia warns of constrained supply of gaming GPUs. Tom's Hardware, 26.02.2026. https://www.tomshardware.com/pc-components/gpus/nvidia-warns-of-constrained-supply-of-gaming-gpus-we-do-believe-for-a-couple-of-quarters-it-is-going-to-be-very-tight
SICK AG: Datenblatt WT2S-P231.
Stokel-Walker, C.: How the AI revolution is triggering a hardware arms race and pushing up prices. Tom's Hardware, 22.08.2025. https://www.tomshardware.com/tech-industry/semiconductors/how-the-ai-revolution-is-triggering-a-hardware-arms-race-and-pushing-up-prices
Ultralytics: Image Classification with Ultralytics YOLO. https://docs.ultralytics.com/tasks/classify
Ultralytics: YOLO11-Dokumentation (Training, Validierung, Lizenz). https://docs.ultralytics.com
Warwick, S.: One year into the AI-induced RAM apocalypse. Tom's Hardware, 28.07.2026. https://www.tomshardware.com/pc-components/ram/one-year-into-the-ai-induced-ram-apocalypse-how-much-does-memory-actually-cost-and-is-there-hope-for-a-more-affordable-future

12. Verzeichnisse

12.1. Abbildungsverzeichnis

Abbildung 1: Lösungsidee (mit KI generierte Darstellung, draw.io) 11
Abbildung 2: IPERKA erklärt (https://www.ict-berufsbildung-bern.ch/resources/Iperka_OdA_200617.pdf) 16
Abbildung 3: IPERKA-Phasen mit Kapiteln (mit KI generierte Darstellung, draw.io) 16
Abbildung 4: Organigramm (mit KI generierte Darstellung, draw.io) 17
Abbildung 5: Projektstrukturplan (mit KI generierte Darstellung, draw.io) 18
Abbildung 6: Terminplan SOLL (eigene Aufnahme) 19
Abbildung 7: Risikomatrix (mit KI generierte Darstellung, draw.io) 20
Abbildung 8: Terminplan SOLL- / IST-Vergleich (eigene Aufnahme aus ProjectLibre) 21
Abbildung 9: Klassifizierung vs Objekterkennung (https://labelyourdata.com/articles/object-detection-vs-image-classification) 25
Abbildung 10: Publish-Subscribe (mit KI generierte Darstellung, draw.io) 30
Abbildung 11: Systemkontext (mit KI generierte Darstellung, draw.io) 31
Abbildung 12: Systemarchitektur (mit KI generierte Darstellung, draw.io) 40
Abbildung 13: Netzwerkplan (mit KI generierte Darstellung, draw.io) 41
Abbildung 14: Firewall-Regel 1883 (eigene Aufnahme) 42
Abbildung 15: Prüfzyklus (mit KI generierte Darstellung, draw.io) 44
Abbildung 16: Schaltplan (mit KI generierte Darstellung, draw.io) 49
Abbildung 17: Prüfband Aufbau Ansicht 1 (eigene Aufnahme) 49
Abbildung 18: Prüfband Aufbau Ansicht 2 (eigene Aufnahme) 50
Abbildung 19: Testplatten (eigene Aufnahme) 50
Abbildung 20: Einstellungen Kamera (eigene Aufnahme) 51
Abbildung 21: Blister mit einer leeren Mulde (eigene Aufnahme) 51
Abbildung 22: Annotation eines Blisters in Label Studio (eigene Aufnahme) 52
Abbildung 23: Resultat Lernkurve v2 (eigene Aufnahme aus Ultralytics) 53
Abbildung 24: Konfusionsmatrix v2 (eigene Aufnahme aus Ultralytics) 54
Abbildung 25: Machbarkeitsstudie Blister IO (eigene Aufnahme aus Ultralytics) 55
Abbildung 26: Machbarkeitsstudie Blister NIO_L (eigene Aufnahme aus Ultralytics) 56
Abbildung 27: Machbarkeitsstudie Blister NIO_U (eigene Aufnahme aus Ultralytics) 56
Abbildung 28: Overview Azure IoT Hub (eigene Aufnahme) 57
Abbildung 29: Nachbearbeitung 3D-Druckteile (eigene Aufnahme) 59
Abbildung 30: Testaufbau Elektronik (eigene Aufnahme) 60
Abbildung 31: Raspberry Pi Image (eigene Aufnahme) 61
Abbildung 32: DHCP statische IP-Adressen (eigene Aufnahme) 61
Abbildung 33: Softwarestruktur Raspberry Pi und Edge-PC (mit KI generierte Darstellung, draw.io) 63
Abbildung 34: Unbearbeitete Ansicht RAZER (eigene Aufnahme) 70
Abbildung 35: Bearbeitete Ansicht (eigene Aufnahme) 70
Abbildung 36: Resultat Lernkurve Prüfband (eigene Aufnahme aus Ultralytics) 72
Abbildung 37: Konfusionsmatrix Prüfband (eigene Aufnahme aus Ultralytics) 72
Abbildung 38: Auswertungen Prüfband (eigene Aufnahme aus Ultralytics) 74

Abbildung 39: Cloud-Datenfluss vom Edge-PC zum Dashboard (mit KI generierte Darstellung, draw.io) 75
Abbildung 40: Messages im IoT Hub (eigene Aufnahme) 75
Abbildung 41: Microsoft Fabric Eventstream (eigene Aufnahme) 76
Abbildung 42: Cloud Dashboard (eigene Darstellung) 77
Abbildung 43: Abfrage der Statusmeldungen (eigene Aufnahme) 82
Abbildung 44: Prüfband Abbild 1 (eigene Aufnahme) 101
Abbildung 45: Prüfband Abbild 2 (eigene Aufnahme) 101
Abbildung 46: Prüfband Abbild 3 (eigene Aufnahme) 102
Abbildung 47: Prüfband Abbild 4 (eigene Aufnahme) 102
Abbildung 48: Prüfband Abbild 5 (eigene Aufnahme) 103

12.2. Glossar

Begriff	Bedeutung
Prüfband	Das gesamte System dieser Arbeit: Mechanik, Elektronik, Raspberry Pi 5, Edge-PC und Cloud-Anbindung
Förderband	Das mechanische Band, auf dem die Tabletten transportiert werden
Edge-PC	PC mit NVIDIA RTX 3060 Ti, auf dem das trainierte KI-Modell läuft
Trainiertes KI-Modell	Mit eigenen Bildern angepasstes YOLO-Modell, das die Tabletten beurteilt
Prüfergebnis	Beurteilung einer Tablette: IO oder NIO
IO / NIO	In Ordnung (gut) / nicht in Ordnung (nicht gut)
Schlupf	Eine fehlerhafte Einheit wird als gut bewertet
Pseudoausschuss	Eine gute Einheit wird als fehlerhaft bewertet
Ausfallsicher (Fail-safe)	Verhalten, das immer dann zum sicheren Zustand (NIO) führt, wenn etwas unsicher ist oder ausfällt
Mulde	Vertiefung eines Blisters, in der eine Tablette liegt
Broker	Vermittler, der MQTT-Nachrichten zwischen Sendern und Empfängern verteilt
Eventstream, Eventhouse	Fabric-Dienste zum Empfangen und Speichern von Ereignisdaten
Transfer Learning	Anpassen eines vortrainierten Modells an eine neue Aufgabe mit eigenen Daten
Mikroschritt	Bruchteil eines Vollschritts eines Schrittmotors

12.3. Abkürzungsverzeichnis

Abkürzung	Bedeutung
AMP	Automatic Mixed Precision
AP	Arbeitspaket
CAD	Computer-Aided Design
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture (NVIDIA)
DRAM	Dynamic Random-Access Memory
EK	Erfolgskriterium
GMP	Good Manufacturing Practice
GPIO	General-Purpose Input/Output
GPU	Graphics Processing Unit
IoT	Internet of Things
KI	Künstliche Intelligenz
KQL	Kusto Query Language
mAP	Mean Average Precision
MQTT	Message Queuing Telemetry Transport
PCB	Printed Circuit Board (Leiterplatte)
QoS	Quality of Service
SSH	Secure Shell
TLS	Transport Layer Security
UC	Use Case (Anwendungsfall)

13. Reflexion

13.1. Weg zum Ziel

Der Weg zum Ziel folgte dem Vorgehensmodell IPERKA mit drei Inkrementen (3.1). Termine, Ereignisse und Abweichungen sind im Soll-Ist-Vergleich dokumentiert (3.7).

13.1.1. Massnahmen, die funktioniert haben

Offene Aufgaben wurden mit kürzeren Restlaufzeiten neu geplant, und vieles wurde aus der Machbarkeitsstudie übernommen. Testaufbau, Software und Druck liefen parallel, und der Puffer in den Kalenderwochen 42 bis 44 wurde als Arbeitszeit genutzt. Um Zeit zu sparen, liess der Diplomand die draw.io-Grafiken nach seinen Vorgaben von der KI generieren. Inhalt und Aufbau gab er vor, das Ergebnis prüfte und korrigierte er selbst.

13.2. Gespräche mit dem Diplomcoach

Nachfolgend die Zusammenfassung der relevanten Inhalte aus den beiden Gesprächen.

13.2.1. Gespräch 1 (08.10.2026)

Kennzeichnung von KI-Textstellen
Der Diplomand nutzt DeepL Write, um seine Texte umzuformulieren. Gemäss den Richtlinien müssten solche Textstellen braun markiert werden, was in diesem Fall nahezu das gesamte Dokument betreffen würde. Der Diplomcoach hielt fest, dass die mit DeepL Write umformulierten Stellen nicht zwingend braun markiert werden müssen. Dafür muss die KI-Deklaration ausführlich beschreiben, wie die KI genutzt wurde.

Terminverzug und Projektsteuerung
Der Diplomand legte seinen aktuellen Terminplan vor, der einen Verzug zeigte. Er schlug vor, eine Woche vor der Abgabe eine optimistische und eine pessimistische Prognose zu erstellen und den Umfang bei Bedarf zu kürzen (Plan A und Plan B). Der Diplomcoach bestätigte diesen Ansatz: Wie im Berufsalltag kann ein Terminverzug vorkommen; entscheidend ist, wie darauf reagiert wird. Die Prognose zeigte schliesslich, dass die Arbeit termingerecht fertig wird; Plan B war nicht nötig.

Abgabe
Für die Abgabe am 28.10.2026 genügt dem Diplomcoach die Arbeit als PDF per E-Mail.

13.2.2. Gespräch 2 (22.10.2026)

Eigenständigkeitserklärung und KI-Deklaration
Der Diplomand stellte dem Diplomcoach einen Entwurf der beiden Abschnitte über Teams zu. Der Diplomcoach empfahl, die KI-Deklaration präziser zu formulieren, damit die Bewertenden erkennen, was Eigenleistung ist und was von der KI stammt. Als Beispiel nannte er die Formulierung «die KI hat Code-Entwürfe erstellt».

Der Diplomand erläuterte, was damit gemeint ist: Wie im Berufsalltag sucht er Code-Snippets und Beispielcode im Internet, übernimmt die relevanten Stellen und passt sie an, bis der Code für seinen Anwendungsfall funktioniert. In diesem Projekt erstellte er mit der KI aus diesen Snippets einen ersten Entwurf und optimierte den Code anschliessend selbst oder mit ihr. Den Output prüfte er dabei stets kritisch; manchmal war der vorgeschlagene Code zum Beispiel gar nicht zielführend und ging nicht auf die nötigen Inputs des Diplomanten ein. Der Diplomcoach bestätigte, dass genau solche Beschreibungen zur KI-Deklaration gehört.

Bewertungsraster
Abschliessend verwies der Diplomcoach auf das Bewertungsraster. Da die Arbeit keinen Auftraggeber hat, entfällt das Feedback des Auftraggebers. Häufig vergessen werde hingegen die Wirtschaftlichkeitsanalyse mit einem angenommenen Stundensatz; sie ist in Kapitel 10 enthalten.

13.3. Zielerreichung gemäss Pflichtenheft

13.3.1. Erfolgskriterien

Nr.	Erfolgskriterium	Nachweis	Erfüllt
EK1	Evaluation von Komponenten und Plattform	Kapitel 5: Nutzwertanalysen und begründete Vergleiche für Edge-Gerät, Kamera, Sensor, Antrieb, KI-Framework, Cloud und Visualisierung	Ja
EK2	Funktionierender Demonstrator	Kapitel 8; Systemtest mit 27 Tabletten	Ja
EK3	Mindestens zwei lokal erkannte Zustände	Drei Zustände (IO, NIO - broken, NIO - wrong) auf dem Edge-PC	Ja
EK4	Betriebsdaten und Erkennungsereignisse in der Cloud	Prüfereignisse (inspection.v2) und Betriebsdaten (status.v1) in IoT Hub und Fabric; T09 27 von 27	Ja
EK5	Erkennungsqualität, Latenz und Stabilität gemessen	Kapitel 9: Schlupf, Genauigkeit, Zeiten, T06, T10, T11	Ja, T10 mit Einschränkung

Nr.	Ziel	Ergebnis	Erfüllt
Z1	0 Schlupf	0	Ja
Z2	Mindestens 95 % korrekt im Normalbetrieb	100 %	Ja
Z3	KI-Prüfzeit höchstens 100 ms	5,7 ms	Ja
Z4	100 % der Ergebnisse in der Cloud bei bestehender Verbindung	27 von 27	Ja
Z5	Keine Störung im Dauerbetrieb	Mehrere Stunden ohne Störung; kein formaler Dauertest	Teilweise

13.3.2. Anforderungen

ID	Anforderung	Status
F01	Das Förderband transportiert Tabletten mit einer einstellbaren Geschwindigkeit.	Erfüllt
F02	Ein Sensor erkennt jede ankommende Tablette und löst die Prüfung aus.	Erfüllt
F03	Nach dem Sensorsignal hält das Förderband die Tablette an einer wiederholbaren Position unter der Kamera an.	Erfüllt
F04	Das KI-Modell klassifiziert jede Tablette als gut (IO) oder nicht gut (NIO).	Erfüllt
F05	Eine unsichere Erkennung führt zu NIO.	Erfüllt
F06	Eine Signalsäule zeigt den Status an.	Erfüllt
F07	Jedes Ergebnis wird mit einem Zeitstempel an die Cloud gemeldet.	Erfüllt
F08	Bei Ausfall der Internetverbindung setzt das Prüfband die Prüfung fort und signalisiert weiterhin lokal.	Erfüllt
F09	Die Cloud zeigt die Ergebnisse in einer übersichtlichen Darstellung an.	Erfüllt
F10	Prüfbilder werden lokal gespeichert.	Erfüllt

ID	Anforderung	Status
N01	Alle Verbindungen zur Cloud sind verschlüsselt.	Erfüllt
N02	Anmeldedaten werden getrennt vom Programmcode gespeichert.	Erfüllt
N03	Das Prüfband verwendet Standardkomponenten und offene Protokolle.	Erfüllt
N04	Cloud-Dienste laufen im kostenlosen Tarif und werden überwacht.	Erfüllt
N05	Das Prüfband ist portabel und läuft auf einem Tisch.	Erfüllt
N06	Einrichtung und Software sind so detailliert dokumentiert, dass eine Fachperson das Prüfband ohne weitere Unterstützung durch den Diplomanden nachbauen kann.	Teilweise

13.4. Lessons learnt

13.4.1. Technisch

Das erste Training der Machbarkeitsstudie brach in der Aufwärmphase ab. Bei kleinen Datensätzen entscheiden Batch Size und Abbruchkriterium über den Erfolg.

Eine Testgruppe der Machbarkeitsstudie enthielt zwei Fehlerarten gleichzeitig, und beim ersten T06-Versuch wurden zwei Faktoren verändert. Pro Testfall nur einen Faktor verändern.

Die Kamera lieferte alte Bilder aus ihrem Puffer und die Ergebnisse waren dadurch um eine Tablette verschoben. Den entscheidenden Hinweis lieferte die Beobachtung, dass das erste Bild jedes Durchlaufs leer war. Vor dem Troubleshooting im Modell lohnt es sich deshalb, die Eingangsdaten zu prüfen, hier den Aufnahmezeitpunkt der Bilder.

Unbeschriftete Klonplatinen kosteten einen halben Tag Messzeit. Somit wäre es besser, dokumentierte Komponenten zu kaufen oder Zeit für das Ausmessen einzuplanen.

13.4.2. Projekt

Edge-Hardware und Cloud-Plattform mussten nach einer Preisänderung und einer Einstellung neu evaluiert werden. Darum wo möglich Kerndienste und Standardprotokolle wählen. Entscheidungen so dokumentieren, dass sie schnell wiederholt werden können.

Die Klärung der Produktion durch Dritte dauerte länger als geplant. Abhängigkeiten von Dritten frühzeitig klären, mit einem Entscheidungstermin und einer Ausweichlösung.

Die Arbeitsbelastung im Beruf sowie eine Krankheit führten zu einer Verzögerung von drei Wochen. Ein von Anfang an eingeplanter Puffer fängt solche Ereignisse ab. Besser wäre es jedoch gewesen, schon früher einen Puffer einzubauen, damit der Terminplan auch schon früher aufgeholt werden kann.

Der Stand wurde zu mehreren Zeitpunkten mit dem Terminplan verglichen, und daraus wurde die weitere Planung abgeleitet. Dadurch wurden Verzögerungen früh sichtbar, und Massnahmen wie Plan A und Plan B konnten rechtzeitig festgelegt werden.

13.4.3. Künstliche Intelligenz (Claude von Anthropic)

In früheren Projekten des Diplomanden erforderten die eigenen 3D-Entwürfe weniger Nacharbeit als die von der KI generierten. KI-Entwürfe müssen vor dem Druck überprüft werden. Die KI schlug aber nützliche Konstruktionsdetails vor, an die der Diplomand nicht gedacht hatte, zum Beispiel ein Langloch, wo es einen grossen Vorteil bringt, wenn das Teil verstellbar ist.

Die KI empfahl Azure IoT Central, ohne von dessen Einstellung zu wissen, und ihr Kameracode verursachte das Pufferproblem. KI kann gute Vorschläge oder Ideen bringen und eignet sich somit ideal als Sparringparter, ersetzt jedoch weder die eigene Überprüfung noch die Verantwortung für Entscheidungen.

13.5. Ausblick

13.5.1. Mögliche Erweiterungen

Für Vision-AI sind Bilddaten das wichtigste Kapital. Ein besseres Förderband, eine Industriekamera und eine definierte Beleuchtung mit einer Abschirmung würden konstantere Bilder liefern, eine Farbprüfung ermöglichen und das Modell unabhängig vom Raumlicht machen.

KI-Modelle können auch mit synthetischen Bildern aus einer Simulation anstelle von echten Fotos trainiert werden. Der Diplomand hat diesen Ansatz in einem Forschungs- und Entwicklungsteam beobachtet. Dies würde viele Fehlervarianten ermöglichen, ohne echte Tabletten zu zerbrechen.

Für einen realen Betrieb sollte das Steuerungsprogramm als Systemdienst (systemd) laufen und durch einen Not-Aus-Taster abgesichert sein; zudem wäre ein Dauertest von Motor und Förderband nötig.

Derzeit darf sich jeweils nur eine Tablette zwischen Sensor und Kamera befinden. Zukünftige Erweiterung um mehrere Tabletten gleichzeitig wäre denkbar.

13.6. Eigenständigkeitserklärung

Ich erkläre, dass ich die vorliegende Diplomarbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe. Wörtlich oder sinngemäss übernommene Stellen sind als solche gekennzeichnet; der Einsatz von KI wird in der KI-Deklaration weiter erläutert.

Die Arbeit wurde für diese Diplomarbeit neu verfasst. Sie ist in gleicher oder ähnlicher Form weder veröffentlicht noch einer anderen Prüfungsstelle vorgelegt worden.

13.7. KI-Deklaration

Das Prüfband und diese Dokumentation wurden unter Einsatz von KI-Werkzeugen erstellt. Zwei Werkzeuge unterstützten die Arbeit; das KI-Modell YOLO ist Gegenstand der Arbeit und keine Hilfe beim Erstellen.

Werkzeug	Anbieter	Zeitraum	Zweck
Claude (Version Sonnet 5.5 & Opus 5.5)	Anthropic	Von 27.04.2026 bis 28.10.2026	Ideenfindung, Code-Entwürfe, Grafiken, Troubleshooting, Überarbeitung der Dokumentation
DeepL Write	DeepL	Von 22.06.2026 bis 28.10.2026	Umformulieren und Korrigieren eigener Texte

Die KI wurde für folgende Zwecke genutzt:

- Entwürfe des Programmcodes für Raspberry Pi und Edge-PC
- Erstellen der OpenSCAD-Konstruktion (3D-Druckteile)
- Troubleshooting bei Inbetriebnahme und Integration
- Erstellung der draw.io-Grafiken nach meinen Vorgaben
- Korrigieren, Umformulieren und Strukturieren der Dokumentation

Alle Ergebnisse der KI habe ich geprüft und sämtliche Inhalte und Entscheide sowie die finale Ausarbeitung und Einreichung liegen in meiner Verantwortung.

Das nachfolgende KI-Proptverzeichnis zeigt, wo KI insbesondere eingesetzt wurde.

13.8. KI-Promptverzeichnis

Aufgeführt sind die Prompts, die ein Ergebnis der Arbeit wesentlich geprägt haben; Rückfragen und Konsolenausgaben sind weggelassen. Die Spalte «Verwendung» zeigt, was mit dem Ergebnis der KI geschah. (Wortlaut jeweils korrigiert und gekürzt.):

Zweck	Propmt	Verwendung
Themenfindung	«Ich möchte ein Computer-Vision-Projekt mit Industriebezug umsetzen, idealerweise ein Maschinenmodul als Teil einer grösseren Anlage, wobei… Wo wird Vision AI in der Industrie häufig eingesetzt? Gib mir einige Ideen.»	Ideen als Ausgangslage genommen, Thema selbst gewählt und eingegrenzt
Themenvalidierung	«Eignet sich die Idee … für eine Diplomarbeit in Systemtechnik Informatik? Mechanik und Elektrotechnik sollen nicht im Fokus stehen… Wie gross ist der Informatikanteil…»	Einschätzung als Entscheidungshilfe genutzt
Dokumentstruktur	«Mein Dokument hat folgende Struktur … Ergänze Unterthemen, die ich zwingend behandeln sollte, und begründe sie.»	Vorschläge geprüft, passende Unterthemen übernommen
Dokumentstruktur	«Mein Dokument sieht aktuell so aus … Fehlt noch etwas?»	Hinweise auf Lücken geprüft und ergänzt
Erfolgskriterien	«Folgend meine messbar formulierten Erfolgskriterien … Hast du Ergänzungen oder würdest du etwas anders formulieren?… Formuliere Vorschläge ebenfalls messbar.»	Eigene Kriterien beibehalten, einzelne Formulierungen geschärft
Anforderungen	«Anbei meine Anforderungen,… unterteilt in funktionale und nicht-funktionale … Schlägst du Änderungen vor oder fehlt etwas Wichtiges?»	Ergänzungen geprüft, einzelne Anforderungen hinzugefügt und umformuliert
Arbeitspakete	«Anhand von IPERKA habe ich folgende Arbeitspakete definiert … Sie dienen später als Grundlage für den Projektstrukturplan. Hast du Ergänzungen oder Anpassungen? Begründe sie und definiere wie in meinen Beispielen jeweils das Ergebnis…»	Arbeitspakete ergänzt, als Grundlage für den PSP genutzt
Prüfablauf	«Folgend mein Prüfablauf … Erstelle dazu eine passende Grafik in draw.io mit denselben Anforderungen…»	Grafiken erstellt, inhaltlich kontrolliert und von Hand wo nötig nachbearbeitet
Schwellwert	«Für dieses Vorhaben würde ich einen Schwellwert von 0,80 wählen (siehe Beispiele …). Gibt es nach deinem Wissensstand Gründe für einen anderen Wert?»	Eigene Wahl gegengeprüft, Entscheid selbst begründet
Softwarearchitektur	«In welche Teile würdest du den Code unterteilen? Meine Beispiele … haben nicht alle eine separate Maschinensteuerung… Gib mir Vorschläge.»	Vorschläge mit eigenen Beispielen verglichen, Modulaufteilung selbst festgelegt

Codebasis	«Anbei Codebeispiele von …, die sich gut für … eignen. Übernimm aus Beispiel 1… den ersten Teil (da dieser übersichtlicher ist)… und ergänze ihn mit … Beispiel 2. Schlag mir begründet vor, ob du aus den übrigen Beispielen etwas übernehmen würdest.»	Zusammengeführten Code als Grundlage genommen, getestet und wo nötig angepasst
Elektrokonzept	«Ich habe die Komponenten … wie folgt angeschlossen (Bilder anbei). Erstelle mir das passende Elektrokonzept.»	Schema mit der realen Verdrahtung abbilden lassen
3D-Druckteile	«Generiere mir die 3D-Druckteile für folgendes Vorhaben: Ich habe folgende Kugellager … und weitere Teile … Montage auf einem Aluprofil 5×5 mm …»	Teile gedruckt, getestet und bei Bedarf angepasst
Modelltraining	«Ist es korrekt, das Modell gemäss dieser Dokumentation … so zu trainieren,…oder würdest du beim folgenden Befehl noch Argumente ergänzen?…»	Argumente geprüft
Modelltraining	«Hilf mir, folgende Terminalausgaben zu interpretieren …»	Interpretation als Hilfe für die eigene Bewertung der Ergebnisse genutzt
Codequalität	«Ich möchte folgende Best Practice … im Code umsetzen. Führe sie im gesamten Code ein… Gibt es weitere Optimierungen, zeig mir je ein Beispiel und begründe, weshalb sie besser sind und wo ich sie überall implementieren könnte.»	Best Practice übernommen, weitere Vorschläge geprüft und auch eingeführt
Cloud Dashboard	«Ich habe eine Fabric-Lizenz, kann aber kein Eventhouse erstellen. Folgende Fehlermeldung … Was könnte die Ursache sein?»	Alle Lösungsansätze geprüft, Ursache selbst behoben
Grafiken	«Erstelle zu folgenden Inhalten eine Grafik in draw.io. Verwende folgende Elemente … Alle Texte sollen sauber in den Formen liegen, die Farben sinnvoll gewählt sein und wie… gestaltet..»	Grafiken erstellt, kontrolliert und nachbearbeitet wo nötig
Review	«Gib mir Ideen, was ich an der Dokumentation noch verbessern könnte. Du erhältst nur Teile davon, die Struktur musst du also nicht prüfen…»	Ideen ausgewählt und selbst umgesetzt
Zusammenfassung Gespräche	«Fasse die folgenden beiden Gespräche so zusammen, dass sie zum Rest meiner Dokumentation passen.»	Aus eigenen Notizen erstellt, abgeglichen und angepasst
Eigenständigkeits-erklärung & KI-Deklaration	«Verbessere folgende KI-Deklaration … Sie soll diese Punkte enthalten … Ergänze, wo sinnvoll.»	Text überarbeitet und vollständig formuliert

14. Fazit

Das Ziel der Arbeit ist erreicht. Das Prüfband zeigt anhand von Standardkomponenten, wie KI-Bilderkennung, Edge-Computing und Cloud-Anbindung zusammenwirken: Ein mit etwa 50 Bildern pro Klasse trainiertes Modell unterscheidet ohne Schlupf zwischen guten, gebrochenen und falschen Tabletten, trifft die Entscheidung in Millisekunden auf dem Edge-PC, arbeitet auch ohne Internetverbindung weiter und meldet jedes Ergebnis an die Cloud. Gleichzeitig zeigt das Projekt die Grenzen auf: Licht, eine Consumer-Kamera und eine Mechanik, die angepasst werden muss.

Impressionen

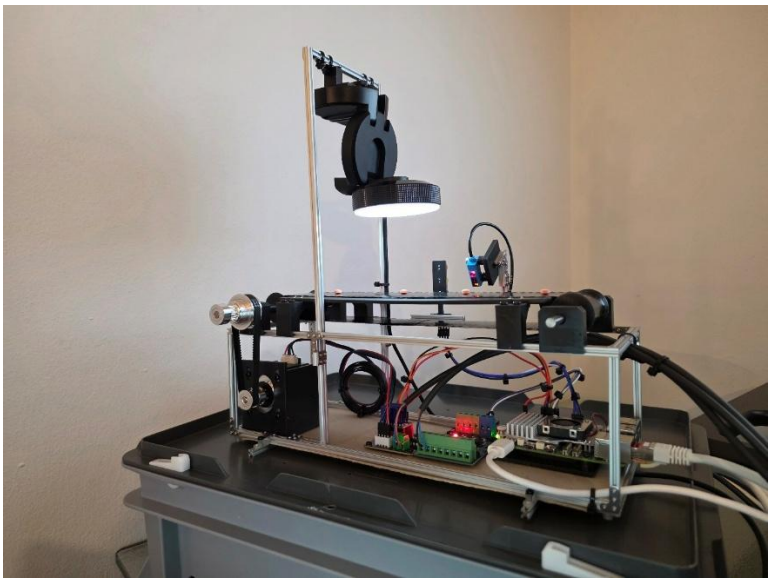


Abbildung 44: Prüfband Abbild 1 (eigene Aufnahme)

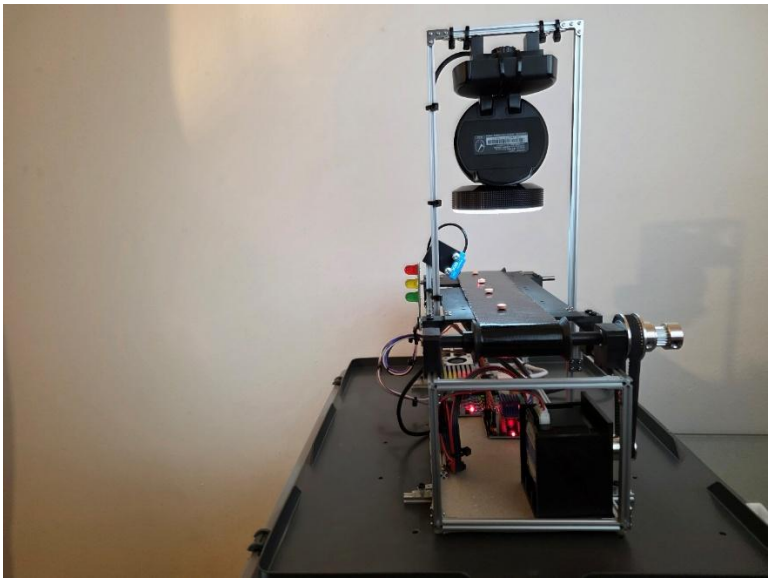


Abbildung 45: Prüfband Abbild 2 (eigene Aufnahme)

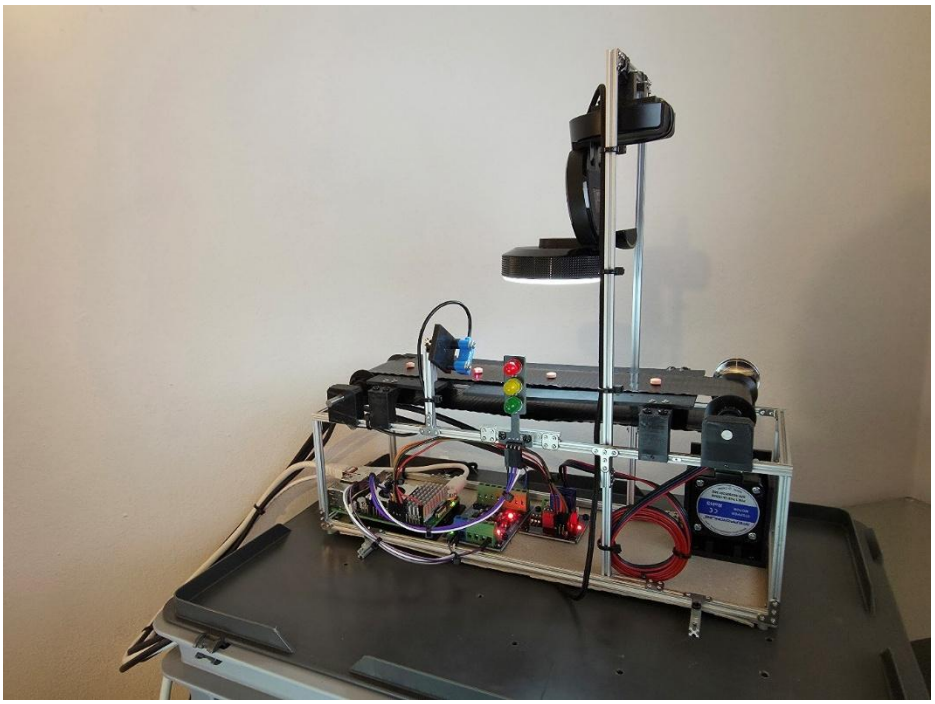


Abbildung 46: Prüfband Abbild 3 (eigene Aufnahme)

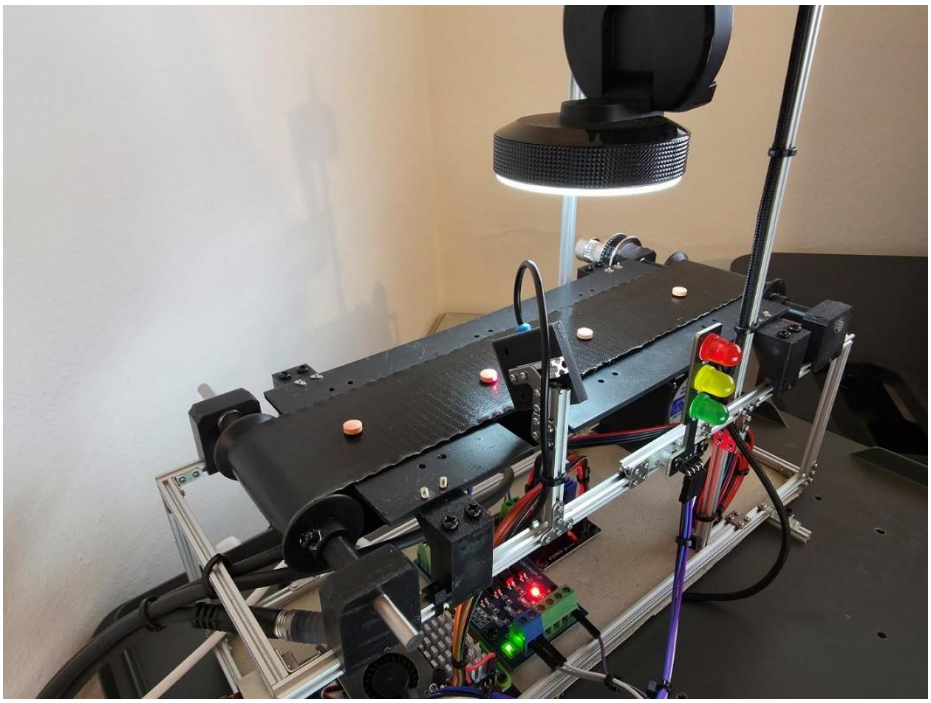


Abbildung 47: Prüfband Abbild 4 (eigene Aufnahme)

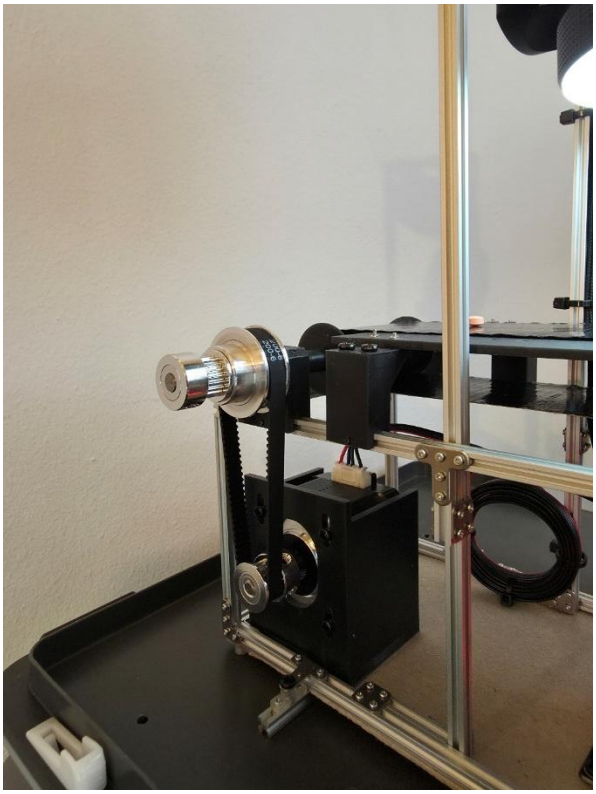


Abbildung 48: Prüfband Abbild 5 (eigene Aufnahme)

Quellcode

config.py

```
# Pins (BCM)
RED, YELLOW, GREEN = 5, 6, 13
SENSOR = 17
STEP, DIR, EN = 18, 23, 24

# Drive
STEPS_PER_MM = 200 * 8 * 2 / (3.1416 * 20)
DIR_FORWARD = 1
SPEED = 800
PULSE_US = 20

# Inspection
DISTANCE_TO_CAMERA_MM = 52
TIMEOUT_S = 2.0
SHOW_RESULT_S = 1.0

# MQTT
MQTT_CONFIG = "~/pruefband/config/mqtt.json"
TOPIC_REQUEST = "pruefband/request"
TOPIC_RESULT = "pruefband/result"
```

hw.py

```
import time

import lgpio

import config as c


class Hardware:
    def __init__(self):
        self.h = self._open_chip()
        for pin in (c.RED, c.YELLOW, c.GREEN, c.STEP):
            lgpio.gpio_claim_output(self.h, pin, 0)
        lgpio.gpio_claim_output(self.h, c.DIR, c.DIR_FORWARD)
        lgpio.gpio_claim_output(self.h, c.EN, 1)
        lgpio.gpio_claim_input(self.h, c.SENSOR, lgpio.SET_PULL_DOWN)

    # GPIO chip (Raspberry Pi 5)
    @staticmethod
    def _open_chip():
        for n in range(8):
            try:
                h = lgpio.gpiochip_open(n)
            except lgpio.error:
                continue
            if "rp1" in str(lgpio.gpio_get_chip_info(h)[3]).lower():
                return h
            lgpio.gpiochip_close(h)
        raise RuntimeError("RP1 GPIO chip not found")

    # Signal tower
    def light(self, red=False, yellow=False, green=False):
        lgpio.gpio_write(self.h, c.RED, int(red))
        lgpio.gpio_write(self.h, c.YELLOW, int(yellow))
        lgpio.gpio_write(self.h, c.GREEN, int(green))

    # Sensor
    def tablet_detected(self):
        return lgpio.gpio_read(self.h, c.SENSOR) == 1

    # Motor
    def enable(self, on=True):
        lgpio.gpio_write(self.h, c.EN, 0 if on else 1)
        time.sleep(0.1)

    def run(self, speed=c.SPEED, steps=0):
        period = int(1_000_000 / speed)
        lgpio.tx_pulse(self.h, c.STEP, c.PULSE_US, period - c.PULSE_US, 0,
steps)
```

```
    def move(self, steps, speed=c.SPEED):
        self.stop()
        self.run(speed, steps)
        time.sleep(steps / speed + 0.1)
        self.stop()

    def stop(self):
        try:
            lgpio.tx_pwm(self.h, c.STEP, 0, 0)
        except lgpio.error:
            pass
        lgpio.gpio_write(self.h, c.STEP, 0)

    # Shutdown
    def close(self):
        self.stop()
        self.enable(False)
        self.light()
        lgpio.gpiochip_close(self.h)
```

conveyer.py

```
import json
import os
import threading
import time

import paho.mqtt.client as mqtt

import config as c
from hw import Hardware


# MQTT
class Inspection:
    def __init__(self):
        with open(os.path.expanduser(c.MQTT_CONFIG), encoding="utf-8") as f:
            cfg = json.load(f)
        self.expected_id = None
        self.result = None
        self.received = threading.Event()
        self.client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2,
client_id="pruefband-pi")
        self.client.username_pw_set(cfg["user"], cfg["password"])
        self.client.on_connect = self.on_connect
        self.client.on_message = self.on_message
        self.client.connect(cfg["host"], cfg.get("port", 1883))
```

```
        self.client.loop_start()

    def on_connect(self, client, userdata, flags, reason_code, properties):
        client.subscribe(c.TOPIC_RESULT, qos=1)

    def on_message(self, client, userdata, msg):
        try:
            data = json.loads(msg.payload)
        except ValueError:
            return
        if data.get("id") == self.expected_id:
            self.result = data.get("result", "NIO")
            self.received.set()

    def request(self, tablet_id):
        self.expected_id = tablet_id
        self.received.clear()
        self.client.publish(c.TOPIC_REQUEST, json.dumps({"id": tablet_id}),
qos=1)
        return self.result if self.received.wait(c.TIMEOUT_S) else None

    def close(self):
        self.client.loop_stop()
        self.client.disconnect()


# Inspection cycle
def main():
    hw = Hardware()
    inspection = Inspection()
    count, times = 0, []
    try:
        hw.enable()
        print("Conveyor running (Ctrl+C to stop)")
        while True:
            hw.run()

            # Sensor detection
            while not hw.tablet_detected():
                time.sleep(0.001)
            hw.move(round(c.DISTANCE_TO_CAMERA_MM * c.STEPS_PER_MM))
            count += 1
            hw.light(yellow=True)

            # Inspection request
            t0 = time.perf_counter()
            result = inspection.request(f"{int(time.time())}-{count:04d}")
            ms = (time.perf_counter() - t0) * 1000
            if result is None:
```

```
                result, info = "NIO", f"NIO - {c.TIMEOUT_S} s no answer (fail-
safe)"
            else:
                times.append(ms)
                info = f"{result} ({ms:.0f} ms)"

            # Result display
            hw.light(green=result == "IO", red=result != "IO")
            print(f"Tablet {count}: {info}")
            time.sleep(c.SHOW_RESULT_S)
            hw.light()
    except KeyboardInterrupt:
        print(f"\nStopped after {count} tablets")
        if times:
            print(f"Response time: avg {sum(times) / len(times):.0f} ms, "
                  f"min {min(times):.0f} ms, max {max(times):.0f} ms")
    finally:
        hw.close()
        inspection.close()


if __name__ == "__main__":
    main()
```

inspection_service.py

```
import json
import queue
import sys
import threading
import time
from datetime import datetime, timezone
from pathlib import Path

import cv2
import paho.mqtt.client as mqtt
from azure.iot.device import IoTHubDeviceClient, Message
from ultralytics import YOLO

from preprocess import preprocess

BASE = Path(__file__).resolve().parents[3]
CONFIG = BASE / "software" / "config" / "mqtt.json"
IOTHUB_CONFIG = BASE / "software" / "config" / "iothub.json"
DATASET_DIR = BASE / "ai-model" / "pruefband" / "dataset"
INSPECTION_DIR = BASE / "tests" / "pruefband" / "inspection_images"
TOPIC_REQUEST = "pruefband/request"
TOPIC_RESULT = "pruefband/result"
```

```
MODEL = BASE / "ai-model" / "pruefband" / "model" / "weights" / "best.pt"
CAMERA = 0
MIN_CONFIDENCE = 0.8
LABELS = {"io": "IO", "nio_broken": "NIO - broken", "nio_wrong": "NIO -
wrong"}


# Camera
class Camera:
    def __init__(self):
        self.cam = cv2.VideoCapture(CAMERA, cv2.CAP_DSHOW)
        self.cam.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*"MJPG"))
        self.cam.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)
        self.cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)
        if not self.cam.isOpened():
            raise RuntimeError(f"Camera {CAMERA} not available")
        self.frame, self.count = None, 0
        self.new_frame = threading.Condition()
        threading.Thread(target=self._read_loop, daemon=True).start()

    def _read_loop(self):
        while True:
            ok, frame = self.cam.read()
            if ok:
                with self.new_frame:
                    self.frame, self.count = frame, self.count + 1
                    self.new_frame.notify_all()

    def capture(self):
        with self.new_frame:
            target = self.count + 2
            if not self.new_frame.wait_for(lambda: self.count >= target,
timeout=1.0):
                raise RuntimeError("No image from camera")
            return self.frame.copy()

    def release(self):
        self.cam.release()


# Inspection
def inspect(model, image):
    probs = model(preprocess(image), verbose=False)[0].probs
    confidence = float(probs.top1conf)
    if confidence < MIN_CONFIDENCE:
        return "NIO - uncertain", confidence
    return LABELS[model.names[probs.top1]], confidence


# Cloud
```

```
class Cloud:
    def __init__(self):
        cfg = json.loads(IOTHUB_CONFIG.read_text(encoding="utf-8"))
        self.client =
IoTHubDeviceClient.create_from_connection_string(cfg["connection_string"])
        self.outbox = queue.Queue()
        threading.Thread(target=self._send_loop, daemon=True).start()

    def send(self, data):
        self.outbox.put(data)

    def status(self, state):
        self.send({
            "schema": "status.v1",
            "id": "edgepc-01",
            "timestamp": datetime.now(timezone.utc).isoformat(),
            "result": state,
            "confidence": 0.0,
            "inspection_ms": 0.0,
        })

    def _send_loop(self):
        while True:
            message = Message(json.dumps(self.outbox.get()),
content_encoding="utf-8",
                              content_type="application/json")
            while True:
                try:
                    self.client.send_message(message)
                    self.outbox.task_done()
                    break
                except Exception:
                    print(f"Cloud offline - retry in 5 s ({self.outbox.qsize()
+ 1} waiting)")
                    time.sleep(5)

    def close(self, timeout_s=5):
        end = time.time() + timeout_s
        while self.outbox.unfinished_tasks and time.time() < end:
            time.sleep(0.1)
        self.client.shutdown()


# MQTT
def new_client():
    try:
        return mqtt.Client(mqtt.CallbackAPIVersion.VERSION2,
client_id="pruefband-edgepc")
    except AttributeError:
        return mqtt.Client(client_id="pruefband-edgepc")
```

```
def main():
    label = sys.argv[1] if len(sys.argv) > 1 else None
    folder = DATASET_DIR / label if label else INSPECTION_DIR
    folder.mkdir(parents=True, exist_ok=True)
    cfg = json.loads(CONFIG.read_text(encoding="utf-8"))
    cam = Camera()
    model = None if label else YOLO(MODEL)
    cloud = None if label else Cloud()
    if cloud:
        cloud.status("started")
    if model:
        inspect(model, cam.capture())

    def on_connect(client, userdata, flags, rc, properties=None):
        client.subscribe(TOPIC_REQUEST, qos=1)
        print(f"Connected, saving images to {folder} (Ctrl+C to stop)")

    def on_message(client, userdata, msg):
        t0 = time.perf_counter()
        try:
            request_id = json.loads(msg.payload)["id"]
        except (ValueError, KeyError):
            return
        try:
            image = cam.capture()
            cv2.imwrite(str(folder / f"{request_id}.jpg"), image)
            result, confidence = inspect(model, image) if model else ("IO",
1.0)
        except Exception as error:
            result, confidence = "NIO - error", 0.0
            print(f"Error: {error}")
        ms = (time.perf_counter() - t0) * 1000
        client.publish(TOPIC_RESULT, json.dumps({"id": request_id, "result":
result}), qos=1)
        print(f"{request_id}: {result} ({confidence:.0%}, {ms:.0f} ms)")
        if cloud:
            cloud.send({
                "schema": "inspection.v2",
                "id": request_id,
                "timestamp": datetime.now(timezone.utc).isoformat(),
                "result": result,
                "confidence": round(confidence, 3),
                "inspection_ms": round(ms, 1),
            })

    client = new_client()
    client.username_pw_set(cfg["user"], cfg["password"])
    client.on_connect = on_connect
```

```
    client.on_message = on_message
    client.connect(cfg["host"], cfg.get("port", 1883))
    try:
        client.loop_forever()
    except KeyboardInterrupt:
        pass
    finally:
        client.disconnect()
        cam.release()
        if cloud:
            cloud.status("stopped")
            cloud.close()


if __name__ == "__main__":
    main()
```

train.py

```
import random
import shutil
from pathlib import Path

import cv2
from ultralytics import YOLO

from preprocess import preprocess

BASE = Path(__file__).resolve().parents[3]
DATASET = BASE / "ai-model" / "pruefband" / "dataset"
TRAINING = BASE / "ai-model" / "pruefband" / "training"
MODELS = BASE / "ai-model" / "pruefband"
VAL_SHARE = 0.2


# Dataset preparation
def prepare():
    shutil.rmtree(TRAINING, ignore_errors=True)
    random.seed(42)
    for class_dir in sorted(p for p in DATASET.iterdir() if p.is_dir()):
        images = sorted(class_dir.glob("*.jpg"))
        random.shuffle(images)
        n_val = round(len(images) * VAL_SHARE)
        for i, path in enumerate(images):
            target = TRAINING / ("val" if i < n_val else "train") /
class_dir.name
            target.mkdir(parents=True, exist_ok=True)
            cv2.imwrite(str(target / path.name),
preprocess(cv2.imread(str(path))))
```

```
        print(f"{class_dir.name}: {len(images) - n_val} train, {n_val} val")


# Training
if __name__ == "__main__":
    prepare()
    model = YOLO("yolo11s-cls.pt")
    model.train(
        data=str(TRAINING), epochs=50, imgsz=640, batch=16, device=0,
        scale=0.0, erasing=0.0, auto_augment=None,
        hsv_h=0.0, hsv_s=0.0, hsv_v=0.4, fliplr=0.5, flipud=0.5,
        project=str(MODELS), name="model", exist_ok=True,
    )
```