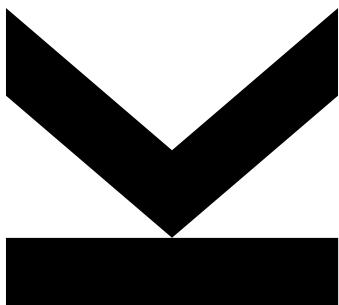


Eingereicht von
Andrea Wieser
Berkan Kalkan
Betim Sulejmani
Matthias Schwarz
Ozan Akkoyun

Betreuer
Mag. Richard Heininger
Thomas Jost, BSc MSc
Univ.-Prof. Dipl.-Ing. Dr.
Christian Stary

Februar 2023

ABSCHLUSSBERICHT



Gruppe C

Smart-shelf und smart-room

Inhaltsverzeichnis

1.	Einleitung.....	5
2.	Projektplanung.....	6
2.1	Problemstellung.....	6
2.2	Ausgangssituation	6
2.3	Projektziele und Nicht-Ziele	7
2.3.1	Projektziele:.....	7
2.3.2	Nice-To-Have:	7
2.3.3	Nicht-Ziele:	7
2.4	Projektschema.....	8
2.5	Meilensteine	9
2.6	Arbeitspakete	9
2.7	Projektstrukturplan.....	15
2.8	Gantt-Diagramm.....	16
3.	Durchführung.....	17
3.1	Planung	17
3.2	Implementierung.....	17
3.3	Testen	19
3.4	Endabgabe	19
3.5	Abgabe und Zeitaufzeichnung	20
4.	Tests und Teststrategie	21
4.1	Teststrategie.....	21
4.2	Durchgeführte Tests	22
5.	Technisches Handbuch	52
5.1	Smart-shelf.....	52
5.1.1	Verwendete Hardware.....	52
5.1.2	Schaltplan:.....	58
5.1.3	Steckplatine:.....	59
5.2	Smart-room	60

5.2.1 Verwendete Hardware	60
5.2.2 Schaltplan:.....	65
5.2.3 Steckplatine:.....	66
5.2.4 Tipp zur Inbetriebnahme:.....	67
5.3 Architektur	68
6. Installationsanleitung	70
6.1 Installationsanleitung für Windows.....	70
6.1.1 Installation Git.....	70
6.1.2 Installation Visual Studio Code	70
6.1.3 Installation Python	71
6.1.4 Repository klonen.....	71
6.1.5 Installation Node Version Manager.....	71
6.1.6 Installation Node.js	75
6.1.7 Installation PlatformIO CLI.....	75
6.1.8 Konfiguration und Flashen eines Mikrocontrollers.....	77
6.1.9 Installation Node-RED	80
6.1.10 Verbindung mit MQTT	84
6.2 Installation für Linux.....	86
6.2.1 Installation Git.....	86
6.2.2 Installation curl	87
6.2.3 Installation Visual Studio Code	88
6.2.4 Installation Python	89
6.2.5 Repository klonen.....	90
6.2.6 Installation PlatformIO CLI.....	90
6.2.7 Konfiguration und Flashen eines Mikrocontrollers.....	91
6.2.8 Installation Node Version Manager	91
6.2.9 Installation Node.js	92
6.2.10 Installation Node-RED	92
6.2.11 Verbindung mit MQTT	94
7. Betriebsanleitung	96

7.1 Betriebsanleitung für Smart-shelf.....	96
7.1.1 MQTT-Anfrage	97
7.1.2 Ermitteln des angefragten Boxtyps	97
7.1.3 Zustand bestimmen	98
7.1.4 Veranschaulichung des Zustands.....	103
7.1.5 Erweitern des JSON-Strings	104
7.1.6 Erstellen eines JSON-Objekts	105
7.1.7 Antwort auf MQTT-Anfrage.....	106
7.1.8 Aktivieren eines Lichts basierend auf dem Zustand.....	107
7.1.9 Reduzieren der Boxanzahl	109
7.1.10 Setzen von Variablen zur Veranschaulichung am User-Interface und Display	110
7.1.11 Anzeigen der vorhandenen Boxen.....	111
7.1.12 Zusammenführung der Anzahl der Boxen des Typs „A“ und „B“	112
7.1.13 Erstellen der Nachricht zur Veranschaulichung am Display	113
7.1.14 Einfügen einer Verzögerung, um Mikrocontroller nicht zu überlasten.....	114
7.2 Betriebsanleitung Smart-room	115
7.2.1 Setzen und Ändern von Variablen und Anzeige am User-Interface	115
7.2.2 Erstellen einer MQTT-Antwort zu eingehender MQTT-Anfrage.....	117
7.2.3 Setzen von Variablen zum Anzeigen am User-Interface	119
7.2.4 Erstellen von MQTT-Antworten zu eingehenden MQTT-Anfrage	120
8. Reflexion	122
8.1 Reflexion der Umsetzung	122
8.2 Reflexion des Projektverlaufs	122
8.3 Reflexion der Ideen	123
8.4 Reflexion der Hilfestellungen	123
8.5 Reflexion der Projektinhalte	124
9. Abbildungsverzeichnis	126
10. Tabellenverzeichnis	130
11. Anhang	132

1. Einleitung

Das Projekt Smart-shelf wurde am 13. Oktober 2022 gestartet und wird voraussichtlich am 28. Februar 2023 abgeschlossen werden. Das dazugehörige Team zur Umsetzung besteht aus Andrea Wieser, Matthias Schwarz, Betim Sulejmani, Berkan Kalkan und Ozan Akkoyun und die Auftraggeber sind Mag. Richard Heininger und Thomas Jost. Das Projekt beschäftigt sich mit der hardware- und softwareseitigen Erstellung eines Systems für die Verwaltung der Belegung eines Regals für Transportboxen und ist als ein Teil des Gesamtprojektes zur automatisierten und intelligenten Verpackung von Vakzinen geplant. Das geforderte Ergebnis ist ein System, welches auf die zu entnehmenden Transportboxen durch farbliche Lichter hinweist und eine Kommunikation mit einem MQTT-Broker über den Zustand der Belegung im Regal ermöglicht.

Das Ziel dieses Projektes besteht daher aus dem Entwickeln der notwendigen Hardware und Software zur Umsetzung der gewünschten Funktionalitäten. Die Hardware soll dabei aus einem Mikrokontroller in Verbindung mit einem Lichtschranken-Sensor zur Erkennung der Verfügbarkeit einer Transportbox und farblichen Lichtern zum Hinweisen auf die zu entnehmende Transportbox bestehen. Ebenso soll die Software für die Steuerung der Hardware in Form einer Installation von Node-RED und einem MQTT-Client zur Kommunikation mit anderen Geräten aufgesetzt werden. Abschließend ist eine Recherche zu für das Anwendungsszenario passenden "Lightweight Embedded Databases" vorgesehen.

Das Zeitausmaß für die Umsetzung des Projektes wurde von dem Auftraggeber mit 150 Stunden pro Person (750 Stunden insgesamt) definiert. Hinsichtlich der personellen Begrenzung wurde dieses Projekt auf fünf Personen mit dem obig genannten Stundenausmaß begrenzt. Finanzielle Begrenzungen sind nicht definiert worden. Mittels zweiwöchentlichen Meetings mit den Auftraggebern wird der Projektfortschritt laufend validiert und etwaige Fragestellungen rundum die Inhalte des Projekts geklärt. Die Projektmitarbeiter untereinander stehen in permanenten Austausch über den Fortschritt des Projekts und können bei auftretenden Problemen sofort gemeinsam an Lösungen arbeiten. Zusätzlich werden Projektfortschritt und -erkenntnisse ausführlich dokumentiert, um den Auftraggebern und Teams bei potenziellen Folgeprojekten einen Einblick in die Ergebnisse des Projekts zu ermöglichen.

2. Projektplanung

Die Zeitdokumentation des Projektes wird durch das Projektteam in Form einer Tabelle in Microsoft Excel durchgeführt. Die Aufteilung in Arbeitspakete, die Auflistung deren Inhalte und die Aufwandsschätzung aller Pakete werden in Jira dokumentiert. Die Zeitplanung wird in Microsoft Excel verwaltet.

2.1 Problemstellung

Die automatisierte Auswahl und Verwaltung von Transportboxen für die Beförderung von Vakzinen bringt ein gewisses Fehlerpotenzial für die Auswahl einer falschen oder nicht vorhandenen Transportbox in den Prozess des Verpackens. Durch die Implementierung der in diesem Projekt vorgesehenen Funktionalitäten wird dieses Fehlerpotenzial vermieden, da das Regal "intelligent" wird und den Auswahlprozess von Transportboxen unterstützen kann. Andere Teile des Gesamtprozesses wie zum Beispiel der Roboterhund Spot können per farblicher Kennzeichnung bei den Möglichkeiten zur Entnahme von den Transportboxen unterstützt werden. Eine Person kann ebenso für das Nachfüllen von einer leeren Transportboxgröße per farblicher Kennzeichnung informiert werden. Durch digitale Kommunikation über das MQTT-Protokoll können ebenso Informationen ausgetauscht werden und basierend auf diesem Austausch von Informationen soll ein entsprechendes User Interface einem Nutzer ermöglichen den Zustand des Regals zu verwalten.

2.2 Ausgangssituation

Die Umsetzung dieses Projekts ist Teil eines cyberphysischen Verpackungssystem für verschiedene Vakzine. Im Gesamtsystem sollen Vakzine laut den Herstellerangaben in verschiedenen großen Transportboxen verpackt werden. An diese Boxen sollen je nach Impfstoff verschiedene Sensoren befestigt werden, welche Umgebungsdaten erfassen, und so die Qualität und die Umstände der Lagerung und des Transports nachvollziehbar machen sollen. In diesem Teilprojekt soll ein Prototyp für das Regal, in welchem die verschiedenen Boxen lagern, entwickelt werden. Dazu ist vorgesehen das physische Regal um einen ESP32 (oder ESP2866) zu erweitern. Dieser ESP soll mittels Sensorik erkennen, ob sich Boxen in den jeweiligen Boxenplätzen befinden und diese Information anhand von LEDs wiedergeben. Weiters sollen Anfragen über die Verfügbarkeit von bestimmten Boxen eines MQTT-Brokers über eine MQTT-Schnittstelle erhalten und beantwortet werden.

2.3 Projektziele und Nicht-Ziele

In den folgenden Unterkapiteln werden die Projektziele und die Nicht-Ziele des Projektes definiert.

2.3.1 Projektziele:

Hauptziel ist die Umsetzung eines Smart-shelfs mittels Mikrocontroller und zugehöriger Sensorik. Die Sensoren sollen erkennen, ob eine bestimmte Box auf den jeweiligem Ablageplatz vorhanden ist und diese Information mittels drei unterschiedlicher Farben eine LED-Ampel wiedergeben.

Grün	Box vorhanden und herausnehmbar
Orange	Box vorhanden, aber nicht herausnehmbar (angefragt)
Rot	Box nicht vorhanden

Tabelle 1: LED- Ampel Definition

Weiters soll es eine Datenverwaltung geben, die beschreibt welche Boxen aktuell auf welchen Boxplätzen liegen.

2.3.2 Nice-To-Have:

Falls Zeit zur Umsetzung bleibt, soll eine MQTT-Schnittstelle implementiert werden, welche Anfragen eines MQTT-Brokers über bestimmte Boxgrößen erhält und die Verfügbarkeit der gewünschten Box retourniert.

Falls Zeitreserven in der Umsetzung des Projekts bleiben, soll ebenfalls eine Recherche zur Implementierung und Anwendung von Datenbanken am ESP durchgeführt werden. Dabei sollen mögliche Lösungsansätze erforscht werden und die optimale Lösung identifiziert werden.

2.3.3 Nicht-Ziele:

Folgende Aspekte werden nicht als Ziel dieses Projektes angesehen:

- Die Dimensionen der Boxen müssen nicht durch die Sensorik erkannt werden.
- Die Boxen werden nicht aus dem Regal entnommen, die jeweilige Einheit (Spot) muss sie selbst entnehmen.
- Das Userinterface des Dashboards soll nicht optisch ansprechend, sondern funktional sein.

2.4 Projektschema

Das folgende Schema gibt einen Überblick über die zu verwendenden Komponenten und deren Schnittstellen. Im Zentrum steht ein ESP32/ESP2866 an welchem über die entsprechenden Pins ein Infrarot- Sender, ein Infrarot- Empfänger sowie ein LED-Ampelmodul angeschlossen werden sollen. Die Infrarotmodule sollen erkennen, ob eine Box am jeweiligen Platz vorhanden ist, und das LED-Ampelmodul soll über den Status des Boxenplatzes informieren. Bei mehreren Boxenplätzen sind dementsprechend mehrere Sensoren und LED-Module notwendig. Der ESP soll mittels Node-RED so konfiguriert werden das er eine Kommunikation mittels MQTT mit einem MQTT-Server (Broker) ermöglicht. Weiters soll er ein Webinterface bereitstellen auf das User mittels HTTPS zugreifen können und so auf die wichtigsten Informationen zugreifen können. Dieses soll ebenfalls mittels Node-RED implementiert werden.

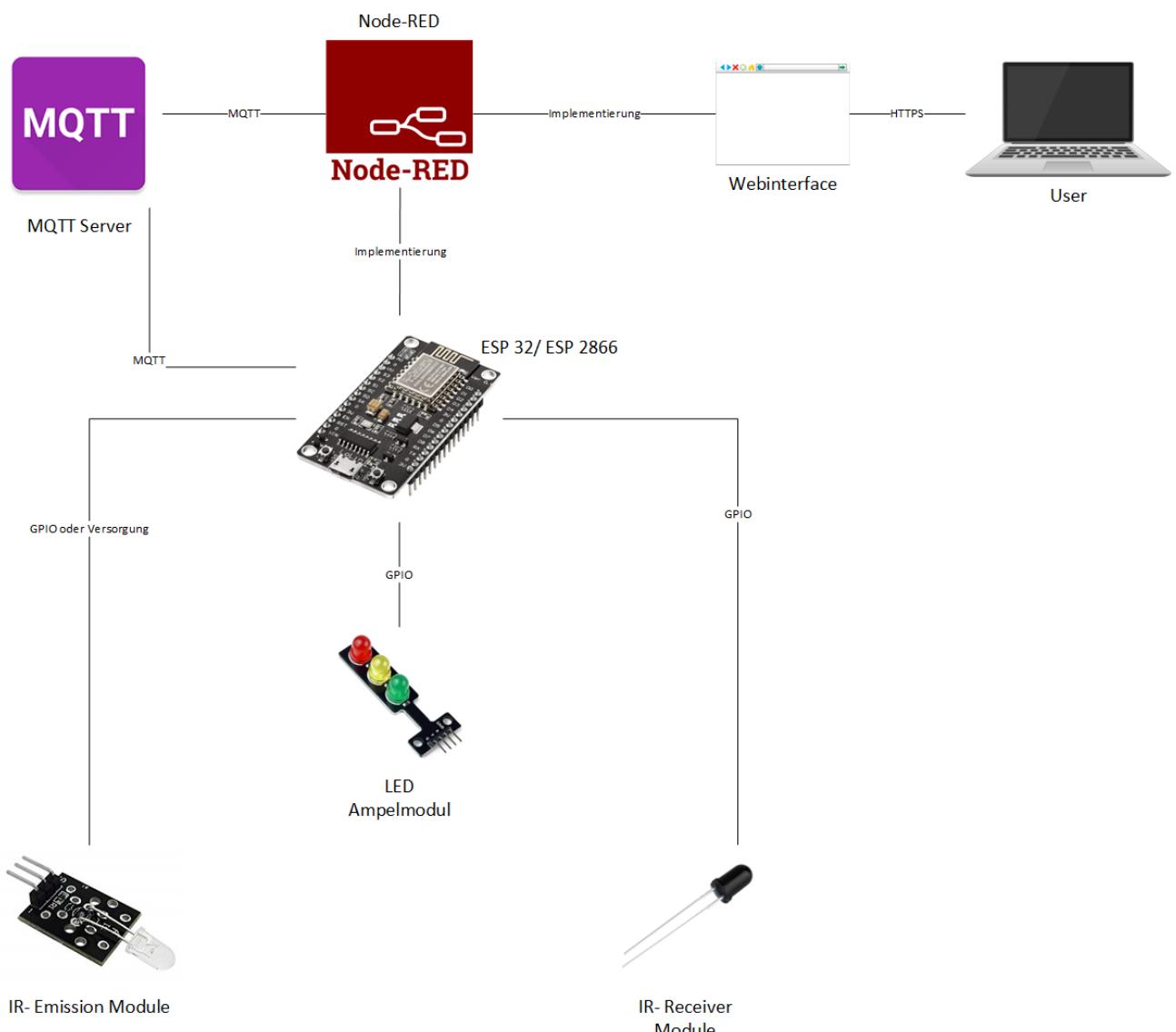


Abbildung 1: Projektschema

2.5 Meilensteine

Meilenstein	Arbeitspakete	Verantwortlichkeit	Deliverables / Abgabe
MS1: Projektplanung	P1, P2, P3, P4	Berkan Kalkan Andrea Wieser Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Projektplan (Dokument)
MS2: Node-RED aufsetzten	I1, I2	Berkan Kalkan Betim Sulejmani	Bericht der Umsetzung, Screenshots der Anwendung
MS3: Hardware-Prototyp	I1, I3	Matthias Schwarz Ozan Akkoyun	Bericht der Umsetzung, Fotos der Hardware
MS4: finale Hardware	I4	Matthias Schwarz Ozan Akkoyun	Bericht der Umsetzung, Fotos der Hardware
MS5: Node-RED UI	I5	Berkan Kalkan Andrea Wieser Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Bericht der Umsetzung, Benutzeroberfläche
MS6: MQTT Anbindung	I6	Berkan Kalkan Andrea Wieser Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Bericht der Umsetzung, Screenshot der Kommunikation
MS7: Systemtest	T1, T2	Berkan Kalkan Andrea Wieser Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Bericht der Tests
MS8: Projektbericht	E1, E2, E3, E4, I7, I8	Berkan Kalkan Andrea Wieser Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Projekbericht (Dokument), Präsentation

Tabelle 2: Meilensteine

2.6 Arbeitspakete

Die folgenden Arbeitspakete wurden mittels Top Down Ansatz erstellt, wobei das Projekt iterativ in immer kleinere Teilprojekte bis hin zu den Arbeitspaketen geteilt wurde. Daraufhin wurden allen Arbeitspaketen eine oder mehrere Hauptverantwortliche zugewiesen und eine Aufwandschätzung durchgeführt. Die definierten Arbeitspakete summieren sich auf 572 Stunden. Die summierte Aufwandsschätzung liegt unter den geforderten 600 Stunden, aber es wird davon ausgegangen, dass ein gewisser Zeitpuffer notwendig ist.

Arbeitspaketbeschreibung			Planung
PSP- Code: P1	AP-Bezeichnung: BPMN-Diagramm	AP-Verantwortung: Berkan Kalkan Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 8h
Inhalte	<ul style="list-style-type: none"> ○ P1.1 Analyse des Use-Cases (2h) ○ P1.2 Erstellen des BPMN-Diagramms (6h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Ausführbares BPMN-Diagramm des Gesamtszenarios 		

Tabelle 3: Arbeitspaket P1

Arbeitspaketbeschreibung			Planung
PSP- Code: P2	AP-Bezeichnung: Ziele festlegen	AP-Verantwortung: Berkan Kalkan Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 8h
Inhalte	<ul style="list-style-type: none"> ○ P2.1 Analyse und Diskussion über Zielsetzung (2h) ○ P2.2 Zielsetzung definieren (6h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Konsens und Definition über Zielsetzung des Teilprojekts 		

Tabelle 4: Arbeitspaket P2

Arbeitspaketbeschreibung			Planung
PSP- Code: P3	AP-Bezeichnung: Arbeitspakete definieren	AP-Verantwortung: Berkan Kalkan Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 8h
Inhalte	<ul style="list-style-type: none"> ○ P3.1 Meilensteine definieren (3h) ○ P3.2 Meilensteine in Arbeitspakete aufteilen (3h) ○ P3.3 Verantwortung, Code, Inhalt, Ziel & Aufwandsschätzung definieren (2h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Meilensteine ○ Arbeitspakete 		

Tabelle 5: Arbeitspaket P3

Arbeitspaketbeschreibung			Planung
PSP- Code: P4	AP-Bezeichnung: Projektplan erstellen	AP-Verantwortung: Berkan Kalkan Andrea Wieser Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 10h
Inhalte	<ul style="list-style-type: none"> ○ P4.1: Einleitung, Ausgangssituation und Problemstellung definieren (4h) ○ P4.2: Gantt Diagramm des Projektverlaufes erstellen (3h) ○ P4.3: Projektstrukturplan erstellen (3h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Ausgearbeiteter Projektplan ○ Gantt Diagramm ○ Projektstrukturplan 		

Tabelle 6: Arbeitspaket P4

Arbeitspaketbeschreibung			Implementation
PSP- Code: I1	AP-Bezeichnung: Tools & Technologien erlernen	AP-Verantwortung: Berkan Kalkan Andrea Wieser Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 20h
Inhalte	<ul style="list-style-type: none"> ○ I1.1 Node- RED Dokumentation studieren (8h) ○ I1.2 Hardware- Kompetenzen aneignen (ESP, Sensoren, ...) (8h) ○ I1.3 MQTT- Dokumentation erarbeiten (4h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Grundlagen der Tools & Technologien verstehen 		

Tabelle 7: Arbeitspaket I1

Arbeitspaketbeschreibung			Implementation
PSP- Code: I2	AP-Bezeichnung: Node-RED aufsetzen	AP-Verantwortung: Berkan Kalkan Betim Sulejmani	Aufwandsschätzung: Gesamt: 8h
Inhalte	<ul style="list-style-type: none"> ○ I2.1: Node-RED aufsetzen (4h) ○ I2.2: einfache Funktionen erstellen (4h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ lauffähige Node-RED Instanz ○ einfache Node-RED Funktion 		

Tabelle 8: Arbeitspaket I2

Arbeitspaketbeschreibung			Implementation
PSP- Code: I3	AP-Bezeichnung: Hardware-Prototyp erstellen	AP-Verantwortung: Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 8h
Inhalte	<ul style="list-style-type: none"> ○ I3.1: Mikrocontroller mit Lichtschranke/LED verbinden (2h) ○ I3.2: Ansteuerung einer einzelnen LED (3h) ○ I3.3: Abfragen einer einzelnen Lichtschranke (3h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ funktionsfähiger Hardware-Prototyp 		

Tabelle 9: Arbeitspaket I3

Arbeitspaketbeschreibung			Implementation
PSP- Code: I4	AP-Bezeichnung: Hardware finalisieren	AP-Verantwortung: Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 6h
Inhalte	<ul style="list-style-type: none"> ○ I4.1: Prototyp um finale Hardware erweitern (6h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ finale Hardware 		

Tabelle 10: Arbeitspaket I4

Arbeitspaketbeschreibung			Implementation
PSP- Code: I5	AP-Bezeichnung: Node-RED Anbindung	AP-Verantwortung: Berkan Kalkan Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 8h
Inhalte	<ul style="list-style-type: none"> ○ I5.1: Node-RED Hardwareanbindung umsetzen (5h) ○ I5.2: UI-Dashboard in Node-RED erstellen (3h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Hardwareanbindung ○ UI-Dashboard und Webinterface 		

Tabelle 11: Arbeitspaket I5

Arbeitspaketbeschreibung			Implementation
PSP- Code: I6	AP-Bezeichnung: MQTT-Anbindung	AP-Verantwortung: Berkan Kalkan Andrea Wieser Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 10h
Inhalte	<ul style="list-style-type: none"> ○ I6.1: MQTT-Client installieren (2h) ○ I6.2: MQTT-Client konfigurieren (8h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ MQTT-Kommunikation der Hardware mit dem Broker ○ Anfragen von Spot (via Broker) beantworten 		

Tabelle 12: Arbeitspaket I6

Arbeitspaketbeschreibung			Implementation
PSP- Code: I7	AP-Bezeichnung: Recherche zu Datenbank	AP-Verantwortung: Berkan Kalkan Betim Sulejmani	Aufwandsschätzung: Gesamt: 8h
Inhalte	<ul style="list-style-type: none"> ○ I7.1: potentielle Lightweight Embedded Databases bestimmen (6h) ○ I7.2: Auswahl der optimalen Lightweight Embedded Databases (2h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Auswahl der optimalen Lightweight Embedded Database 		

Tabelle 13: Arbeitspaket I7

Arbeitspaketbeschreibung			Implementation
PSP- Code: I8	AP-Bezeichnung: Dokumentation der Implementierung	AP-Verantwortung: Berkan Kalkan Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 20h
Inhalte	<ul style="list-style-type: none"> ○ I8.1: Laufende Dokumentation der durchgeföhrten Schritte (20h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Dokumentation der Umsetzung ○ Installationsanleitung ○ Systemdokumentation 		

Tabelle 14: Arbeitspaket I8

Arbeitspaketbeschreibung			Testphase
PSP- Code: T1	AP-Bezeichnung: Funktionstest	AP-Verantwortung: Berkan Kalkan Betim Sulejmani	Aufwandsschätzung: Gesamt: 10h
Inhalte	<ul style="list-style-type: none"> ○ T1.1: Testen der einzelnen Komponenten (3h) ○ T1.2: Testen der Kommunikation (3h) ○ T1.3: Behebung auftretender Fehler (4h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Bericht über die durchgeföhrten Tests ○ Dokumentation der Fehler und deren Behebung 		

Tabelle 15: Arbeitspaket T1

Arbeitspaketbeschreibung			Testphase
PSP- Code: T2	AP-Bezeichnung: Testen des Teilsystems	AP-Verantwortung: Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 10h
Inhalte	<ul style="list-style-type: none"> ○ T2.1: Tests im Bezug zum Teilsystems durchführen (8h) ○ T2.2 Behebung auftretender Fehler (2h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Bericht über die durchgeföhrten Tests ○ Dokumentation der Fehler und deren Behebung 		

Tabelle 16: Arbeitspaket T2

Arbeitspaketbeschreibung			Ergebnis
PSP- Code: E1	AP-Bezeichnung: Reflexion	AP-Verantwortung: Berkan Kalkan Andrea Wieser Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 8h
Inhalte	<ul style="list-style-type: none"> ○ E1.1: Soll-Ist-Vergleich zwischen Projektplanung und Umsetzung (4h) ○ E1.2: Erfahrungsbericht verfassen (4h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Reflexion ○ Erfahrungsbericht 		

Tabelle 17: Arbeitspaket E1

Arbeitspaketbeschreibung			Ergebnis
PSP- Code: E2	AP-Bezeichnung: Dokumentation fertigstellen	AP-Verantwortung: Berkan Kalkan Andrea Wieser Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 8h
Inhalte	<ul style="list-style-type: none"> ○ E2.1: Dokumentation finalisieren (8h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Abgeschlossene Dokumentation 		

Tabelle 18: Arbeitspaket E2

Arbeitspaketbeschreibung			Ergebnis
PSP- Code: E3	AP-Bezeichnung: Projektpräsentation	AP-Verantwortung: Berkan Kalkan Andrea Wieser Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 8h
Inhalte	<ul style="list-style-type: none"> ○ E3.1: Präsentation für Anwendungsszenario erstellen und vorbereiten (6h) ○ E3.2: Projektergebnis präsentieren (2h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Projektergebnis präsentieren 		

Tabelle 19: Arbeitspaket E3

Arbeitspaketbeschreibung			Ergebnis
PSP- Code: E4	AP-Bezeichnung: Projektabchluss	AP-Verantwortung: Berkan Kalkan Andrea Wieser Betim Sulejmani Matthias Schwarz Ozan Akkoyun	Aufwandsschätzung: Gesamt: 2h
Inhalte	<ul style="list-style-type: none"> ○ E4.1: Abschlussgespräch und Übernahme vom Projekt durch Auftraggeber (2h) 		
Ergebnis/Ziel	<ul style="list-style-type: none"> ○ Projektabchlussbericht 		

Tabelle 20: Arbeitspaket E4

2.7 Projektstrukturplan

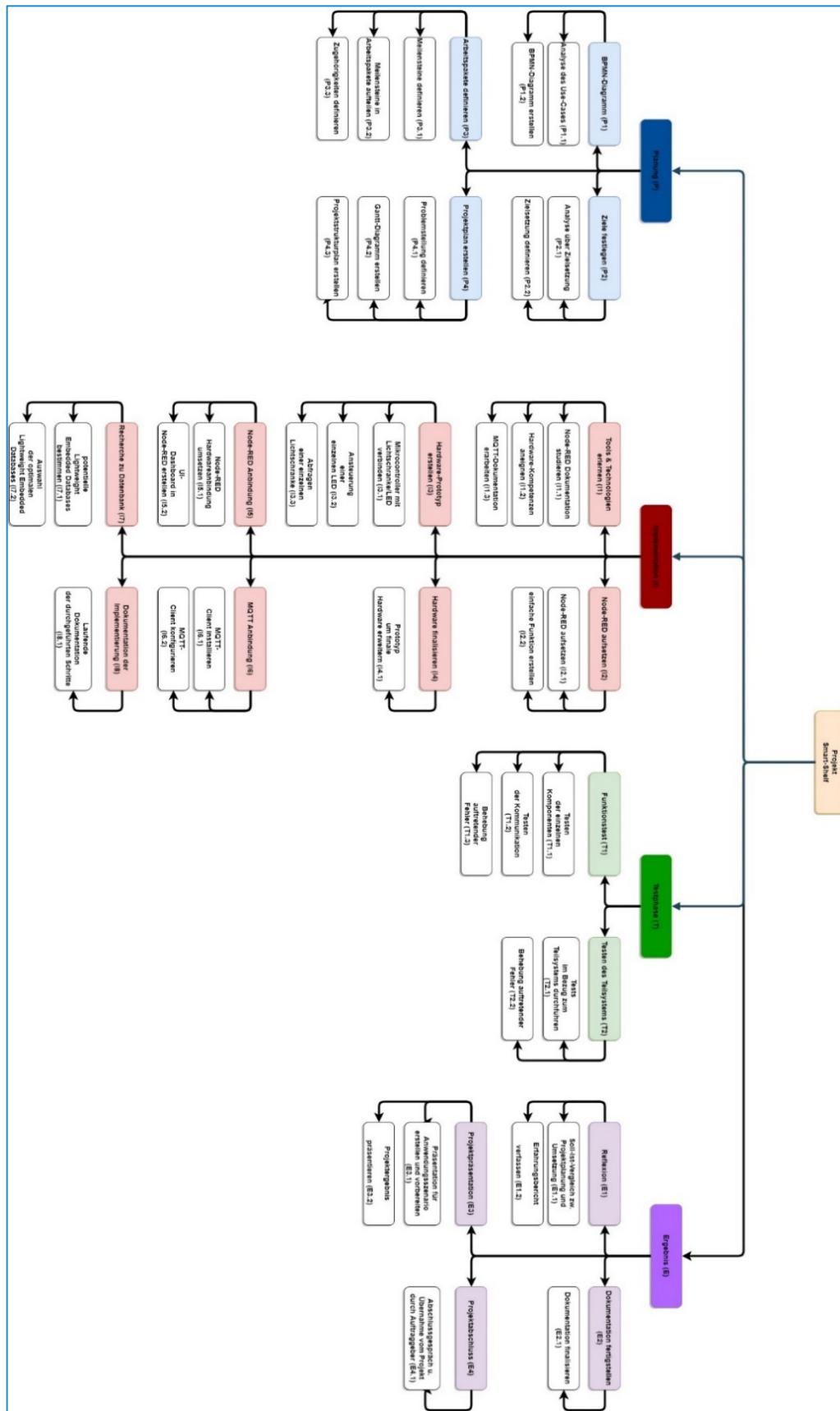


Abbildung 2: Projektstrukturplan

2.8 Gantt-Diagramm

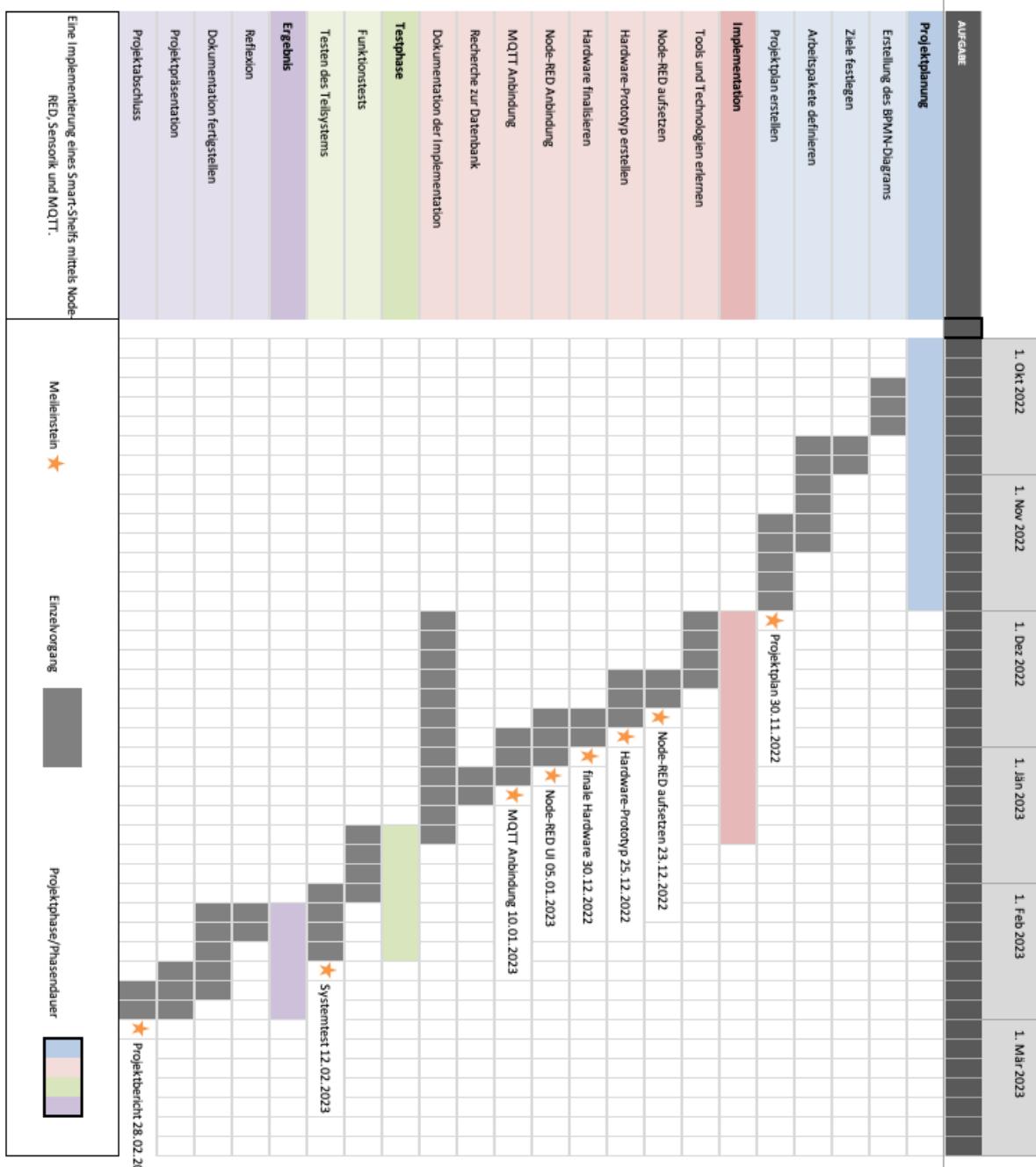


Abbildung 3: Gantt-Diagramm

3. Durchführung

Im Folgenden wird beschrieben, wie das Projekt durchgeführt worden ist. Dabei wird mit dem ersten Arbeitspaket begonnen und die Bearbeitung, sowie Abweichungen von den Arbeitspaketbeschreibungen, beschrieben.

3.1 Planung

In diesem Abschnitt wird beschrieben, wie wir bei der Planung vorgegangen sind und welche Tools wir dafür genutzt haben.

P1: Zu Beginn des Projekts wurde ein BPMN-Diagramm, für die das Tool Camunda hergenommen wurde, nach der Beschreibung des Szenarios, die wir von den LVA-Leitern zur Verfügung gestellt bekommen haben, erstellt. Nach der Analyse des Use-Cases haben wir ein ausführbares BPMN-Diagramm des Gesamtszenarios kreieren können.

P2: Als nächstes wurden die Projektziele vereinbart. Dazu haben wir mögliche Ziele analysiert und über diese diskutiert. Aus dieser Diskussion ist die Definition der Zielsetzung des Teilprojekts entstanden. Diese diente als Grundlage für die Definition der Arbeitspakete. Für die Diskussion haben wir eine Discord-Gruppe erstellt und uns in einem Sprachkanal getroffen.

P3: Aus der Definition der Zielsetzung haben wir Meilensteine abgeleitet, welche wiederum in Arbeitspakete aufgeteilt werden mussten. Diese Pakete werden anhand von Codes unterschieden und beinhalten die Verantwortung der Projektbeteiligten, sowie die Inhalte des Pakets, als auch eine Aufwandschätzung in Stunden. Wir haben diese Pakete nicht nur in Tabellen in Word beschrieben, sondern auch in Jira, also einem Tool für Projektmanagement.

P4: Um mit der Planung des Projekts abzuschließen, musste ein Projektplan erstellt werden. Dieser beinhaltet zudem ein Gantt Diagramm und einen Projektstrukturplan, um die Projektinhalte graphisch darzustellen. Für die Erstellung wurden die Tools Draw.io und Excel mit einem Template genutzt.

3.2 Implementierung

In diesem Abschnitt wird beschrieben, wie die geplanten Arbeitspakete umgesetzt wurden, als auch zusätzliche Maßnahmen, die für den Projekterfolg entscheidend waren. Vorab mussten wir uns zwischen zwei Optionen entscheiden. Low-Code-Programmierung mittels Node-RED war die

erste Option und jene die wir gewählt haben. Die andere Option wäre die „klassische“ Mikrocontroller-Programmierung gewesen.

I1: Um das Projekt Smart-shelf zu starten haben wir uns mit der Technologie Node-RED auseinandergesetzt. Dazu haben wir uns in die Node-RED Dokumentation eingelesen. Wir haben Zugang zu einem Projekt auf GitLab erhalten und dieses auf unseren Rechnern installiert. Von unseren Betreuern haben wir eine Einführung in die Funktionsweise des GitLab Projekts bekommen. Zusätzlich wurde uns der Mikrocontroller und die Sensoren vom Communications Engineering Institut zur Verfügung gestellt. Die technischen Details dazu, wurden uns in Form von Dokumentationen bereitgestellt. Wir haben uns in diese Dokumentationen eingelesen und daraus evaluiert, welche Sensoren wir für unser Projekt brauchen. Uns wurde gezeigt, wie wir die einzelnen Komponenten miteinander verbinden können und die daraus hervorgehenden Daten erklärt. Mit Daten ist hier gemeint, dass das Verhalten der Sensoren gespeichert und aufgezeichnet wird. Aus diesen Daten können grafische Darstellungen erzeugt werden. In diesem Meeting wurde uns nicht nur die Verbindung der Komponenten erklärt, sondern auch alles, was damit verbunden ist und auf was wir Acht geben müssen, um defekte Sensoren vorzubeugen. Die Informationen in den Dokumentationen haben uns dabei geholfen, die Sensoren richtig zu verbinden und auf Faktoren wie die Stromversorgung zu achten. Das genaue Vorgehen wird im technischen Handbuch erklärt.

I2: Für das Projekt brauchten wir eine lauffähige Node-RED Instanz. Wir wollten Node-RED eigentlich in Linux aufsetzen, da wir nur für dieses Betriebssystem die Befehle kannten. Aufgrund von Problemen mit Linux haben wir die Befehle für Windows vom Techniker gefordert und diese dann erfolgreich in Windows umgesetzt. Dort konnten wir mit dem Erstellen von Funktionen beginnen.

I3: Um einen funktionsfähigen Hardware-Prototyp zu erstellen, mussten wir die Sensoren mit dem Mikrocontroller verbinden. Eigentlich wollten wir eine Lichtschranke, aber der Sensor dafür war nicht mit der Batterie des Mikrocontrollers kompatibel. Wir haben vor Ort mit dem Techniker alle Sensoren getestet und nur funktionierende mit nach Hause genommen, wo wir diese dann mit unserem Node-RED in Windows verbunden haben.

I4: In diesem Paket wurde der Prototyp um die finale Hardware erweitert. Da wir noch zeitliche Ressourcen zur Verfügung hatten, haben wir unser Smart-shelf Projekt, um das Smart-room Projekt, erweitert.

I5: Es wurden Node-RED Flows erstellt, um verschiedene Iterationen darzustellen. Mit Node-RED können in einem Flow unterschiedliche Input- und Output-Nodes miteinander verbunden werden. Jede Node hat eine festgelegte und eindeutige Aufgabe. Das Display am Board hat nicht funktioniert, die Anzahl der Boxen verändert sich nicht.

I6: Der MQTT-Client wurde installiert und erfolgreich konfiguriert.

I7: Die Idee zur Implementation einer Datenbank zur Speicherung der Boxen auf dem Flash-Speicher Mikrocontroller wurde verworfen. Der Grund ist, der Flash-Speicher sich für eine Datenbank nicht eignet und bei häufigen Schreibvorgängen funktionsunfähig wird. Als Alternativen wurde die Nutzung von einem Modul für den Einsatz von resisterenteren Speichergeräten (z.B. USB-Sticks) in Betracht gezogen. Die notwendige Hardware war jedoch nicht verfügbar.

I8: Die Implementierung wurde laufend mitdokumentiert. Die Systemdokumentation und die Installationsanleitung sind am Ende des Abschlussberichts enthalten.

3.3 Testen

Im folgenden Abschnitt wird die Teststrategie erläutert.

T1: Die einzelnen Komponenten mussten auf ihre Funktionalitäten getestet werden, sowie ob diese kommunikativ sind. Alle aufgetretenen Fehler konnten behoben werden, die Dokumentation und die Tests werden später im Dokument beschrieben.

T2: Nachdem die Funktionalität und die Kommunikationsfähigkeit der Komponenten getestet wurden, mussten diese auch noch richtig im Teilsystem in Zusammenhang mit den Ausarbeitungen der Gruppe B funktionieren, also den gewünschten Output als MQTT-Response und durch das Aufleuchten des jeweiligen Lichts liefern. Alle aufgetretenen Fehler konnten behoben werden, die Dokumentation und die Tests werden später im Dokument beschrieben.

3.4 Endabgabe

Im folgenden Abschnitt werden die Vorbereitungen für die Endabgabe erläutert und wie bei der Dokumentation vorgegangen wurde. Alle Maßnahmen für den Projektabschluss werden genannt.

E1: Um unsere Erfahrungen in einem Dokument festzuhalten haben wir eine Reflexion des Projekts geschrieben, in dem wir unsere Erkenntnisse teilen. Dabei werden Unterschiede

zwischen dem Projektplan und der tatsächlichen Umsetzung des Projekts aufgezeigt. Man kann auch von einem Soll-Ist-Vergleich sprechen. Konflikte bei der Umsetzung und andere relevante Informationen werden in der Reflexion beschrieben.

E2: Für jedes Vorgehen haben wir eine Anleitung erstellt, damit das Vorgehen in unserem Projekt nachvollziehbar wird. Wir haben Abbildungen verwendet, um die Dokumentation anschaulicher und leichter verständlich zu machen.

E3: Es wurde keine Präsentation im Sinne einer PowerPoint erstellt, aber wir haben uns auf ein Gespräch mit unseren Betreuern vorbereitet, damit wir ausführlich von unseren Ergebnissen am 09. Februar 2023 berichten können.

E4: Das Projekt wurde am 27. Februar abgeschlossen und vom Auftraggeber am selben Tag übernommen.

3.5 Abgabe und Zeitaufzeichnung

Die endgültige Abgabe erfolgte am 27. Februar 2023. Der Abschlussbericht, die Node-RED Konfiguration als auch der Camunda-Showcase Prozess wurden dazu per E-Mail an die LVA-Leiter übermittelt. Die Node-RED Konfiguration ist ebenso über folgendes GitHub-Repository erreichbar: https://github.com/Ozakoyun/smart_shelf. Die Überreichung der Hardware erfolgte bereits am 09. Februar 2023.

Die Zeitaufzeichnung der jeweiligen Mitglieder ist im Anhang zu finden.

4. Tests und Teststrategie

In den folgenden Unterkapiteln werden die Teststrategie mit den angestrebten Testszenarios und den tatsächlichen Testdurchführungen erläutert.

4.1 Teststrategie

Als Teststrategie wird das Ziel verfolgt, dass Projekt während des Projektverlaufs kontinuierlich zu testen und eine lösungsorientierte Entwicklung sicherzustellen.

Folgende Funktionalitäten sollen dabei grundsätzlich getestet werden:

- Smart-shelf:
 - Ein korrekt formulierter MQTT-Request zum Abfragen der Verfügbarkeit einer Box wird mit dem Aufgehen des Lichts beantwortet.
 - Ein korrekt formulierter MQTT-Request zum Abfragen der Verfügbarkeit einer Box wird mit einer MQTT-Response über die Verfügbarkeit beantwortet.
- Smart-room:
 - Ein korrekt formulierter MQTT-Request zum Abfragen des Status des Türsensors im Smart-room wird mit einer MQTT-Response über den Status beantwortet.
 - Ein korrekt formulierter MQTT-Request zum Abfragen der Temperatur des Klimasensors im Smart-room wird mit einer MQTT-Response beantwortet.
 - Ein korrekt formulierter MQTT-Request zum Abfragen der Luftfeuchtigkeit des Klimasensors im Smart-room wird mit einer MQTT-Response beantwortet.
- Camunda-Showcase Prozess:
 - Der Camunda-Showcase Prozess kann Schritt für Schritt fehlerfrei durchgeführt werden.

4.2 Durchgeführte Tests

Test 1.1			
Test-Datum:	23.12.2022	System-Datum:	23.12.2022
Tester:	Ozan Akkoyun		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ A per Licht und MQTT-Response unter smart-shelf/response/A		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI über null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-shelf/request/A mit Inhalt "A" absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/A "Available" wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Kein Licht leuchtet auf • Keine Antwort unter smart-shelf/response/A 		
Ergebnis:	Nicht erfolgreich		
Konsequenz:	<p>Untersuchung der Fehlerursache: Inkorrekte Netzwerddaten auf Mikrocontroller geflasht</p> <p>Behebung der Fehlerursache: Korrekte Netzwerddaten auf Mikrocontroller flashen</p>		

Tabelle 21: Test 1.1

Test 1.2			
Test-Datum:	23.12.2022	System-Datum:	23.12.2022
Tester:	Ozan Akkoyun		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ A per Licht und MQTT-Response unter smart-shelf/response/A		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI über null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-shelf/request/A mit Inhalt "A" absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/A "Available" wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/A „Available“ wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status wird korrekt zurückgegeben		

Tabelle 22: Test 1.2

Test 1.3			
Test-Datum:	23.12.2022	System-Datum:	23.12.2022
Tester:	Ozan Akkoyun		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ A per Licht und MQTT-Response unter smart-shelf/response/A		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI auf null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-shelf/request/A mit Inhalt "A" absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Gelbes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/A "Incomplete" wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Kein Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/A "Incomplete" wird abgesetzt 		
Ergebnis:	Nicht erfolgreich		
Konsequenz:	<p>Untersuchung der Fehlerursache: Licht-Pin nicht angeschlossen am Mikrocontroller</p> <p>Behebung der Fehlerursache: Licht-Pin am Mikrocontroller anschließen</p>		

Tabelle 23: Test 1.3

Test 1.4			
Test-Datum:	23.12.2022	System-Datum:	23.12.2022
Tester:	Ozan Akkoyun		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ A per Licht und MQTT-Response unter smart-shelf/response/A		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI auf null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-shelf/request/A mit Inhalt "A" absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Gelbes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/A "Incomplete" wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Gelbes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/A "Incomplete" wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status der Box wird korrekt zurückgegeben		

Tabelle 24: Test 1.4

Test 1.5			
Test-Datum:	23.12.2022	System-Datum:	23.12.2022
Tester:	Ozan Akkoyun		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ A per Licht und MQTT-Response unter smart-shelf/response/A		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI über null setzen und Box nicht vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-shelf/request/A mit Inhalt "A" absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Rotes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response "Empty" wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Rotes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response "Empty" wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status der Box wird korrekt zurückgegeben		

Tabelle 25: Test 1.5

Test 2.1			
Test-Datum:	23.12.2022	System-Datum:	23.12.2022
Tester:	Ozan Akkoyun		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ B per Licht und MQTT-Response unter smart-shelf/response/B		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI über null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-shelf/request/B mit Inhalt "B" absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/B "Available" wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Kein Licht leuchtet auf • Keine Antwort per MQTT unter smart-shelf/response/B 		
Ergebnis:	Nicht erfolgreich		
Konsequenz:	<p>Untersuchung der Fehlerursache: Absturz des MQTT-Servers</p> <p>Behebung der Fehlerursache: Neustart des MQTT-Servers</p>		

Tabelle 26: Test 2.1

Test 2.2			
Test-Datum:	23.12.2022	System-Datum:	23.12.2022
Tester:	Ozan Akkoyun		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ B per Licht und MQTT-Response unter smart-shelf/response/B		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI über null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-shelf/request/B mit Inhalt "B" absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/B "Available" wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/B "Available" wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status der Box wird korrekt zurückgegeben		

Tabelle 27: Test 2.2

Test 2.3			
Test-Datum:	23.12.2022	System-Datum:	23.12.2022
Tester:	Ozan Akkoyun		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ B per Licht und MQTT-Response unter smart-shelf/response/B		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI auf null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-shelf/request/A mit Inhalt "B" absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Gelbes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/B "Incomplete" wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Kein Licht leuchtet auf • Richtige Antwort per MQTT unter smart-shelf/response/B "Incomplete" 		
Ergebnis:	Nicht erfolgreich		
Konsequenz:	<p>Untersuchung der Fehlerursache: Licht-Pin nicht in Node-RED als Digital Output konfiguriert</p> <p>Behebung der Fehlerursache: Licht-Pin in Node-RED als Digital Output konfiguriert</p>		

Tabelle 28: Test 2.3

Test 2.4			
Test-Datum:	23.12.2022	System-Datum:	23.12.2022
Tester:	Ozan Akkoyun		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ B per Licht und MQTT-Response unter smart-shelf/response/B		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI auf null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-shelf/request/A mit Inhalt "B" absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Gelbes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/B "Incomplete" wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Gelbes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response/B "Incomplete" wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status der Box wird korrekt zurückgegeben		

Tabelle 29: Test 2.4

Test 2.5			
Test-Datum:	23.12.2022	System-Datum:	23.12.2022
Tester:	Ozan Akkoyun		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ B per Licht und MQTT-Response unter smart-shelf/response/B		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI über null setzen und Box nicht vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-shelf/request/A mit Inhalt "B" absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Rotes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response "Empty" wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Rotes Licht leuchtet auf • Antwort per MQTT unter smart-shelf/response "Empty" wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status der Box wird korrekt zurückgegeben		

Tabelle 30: Test 2.5

Test 3.1			
Test-Datum:	26.01.2023	System-Datum:	26.01.2023
Tester:	Andrea Wieser		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 		
Testgegenstand:	ESP-8266		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status des Türsensors per MQTT-Response unter smart-room/response/door		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP8266 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Box vor Infrarotsensor für Türerkennung positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-room/request/door mit Inhalt "request" absetzen 		
Erwartung:	Antwort per MQTT unter smart-room/response/door "1" wird abgesetzt		
Output:	Keine Antwort per MQTT unter smart-shelf/response/door "1"		
Ergebnis:	Nicht erfolgreich		
Konsequenz:	<p>Untersuchung der Fehlerursache: Inkorrekte Netzwerkdaten auf Mikrocontroller geflasht</p> <p>Behebung der Fehlerursache: Korrekte Netzwerkdaten auf Mikrocontroller flashen</p>		

Tabelle 31: Test 3.1

Test 3.2			
Test-Datum:	26.01.2023	System-Datum:	26.01.2023
Tester:	Andrea Wieser		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 		
Testgegenstand:	ESP-8266		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status des Türsensors per MQTT-Response unter smart-room/response/door		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP8266 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Box vor Infrarotsensor für Türerkennung positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-room/request/door mit Inhalt "request" absetzen 		
Erwartung:	Antwort per MQTT unter smart-room/response/door "1" wird abgesetzt		
Output:	Antwort per MQTT unter smart-shelf/response/door "1" wird abgesetzt		
Ergebnis:	Erfolgreich		
Konsequenz:	Status des Türsensors wird korrekt zurückgegeben		

Tabelle 32: Test 3.2

Test 3.3			
Test-Datum:	26.01.2023	System-Datum:	26.01.2023
Tester:	Andrea Wieser		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 		
Testgegenstand:	ESP-8266		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status des Türsensors per MQTT-Response unter smart-room/response/door		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP8266 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Box nicht vor Infrarotsensor für Türerkennung positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-room/request/door mit Inhalt "request" absetzen 		
Erwartung:	Antwort per MQTT unter smart-room/response/door "0" wird abgesetzt		
Output:	Keine Antwort per MQTT unter smart-shelf/response/door "0"		
Ergebnis:	Nicht erfolgreich		
Konsequenz:	<p>Untersuchung der Fehlerursache: Infrarotsensor fehlerhaft</p> <p>Behebung der Fehlerursache: Austausch Infrarotsensor</p>		

Tabelle 33: Test 3.3

Test 3.4			
Test-Datum:	26.01.2023	System-Datum:	26.01.2023
Tester:	Andrea Wieser		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 		
Testgegenstand:	ESP-8266		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status des Türsensors per MQTT-Response unter smart-room/response/door		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP8266 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Box nicht vor Infrarotsensor für Türerkennung positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart-room/request/door mit Inhalt "request" absetzen 		
Erwartung:	Antwort per MQTT unter smart-room/response/door "0" wird abgesetzt		
Output:	Antwort per MQTT unter smart-shelf/response/door "0" wird abgesetzt		
Ergebnis:	Erfolgreich		
Konsequenz:	Status des Türsensors wird korrekt zurückgegeben		

Tabelle 34: Test 3.4

Test 3.5			
Test-Datum:	26.01.2023	System-Datum:	26.01.2023
Tester:	Andrea Wieser		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 		
Testgegenstand:	ESP-8266		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe der Temperatur des Klimasensors per MQTT-Response unter smart-room/response/temperature		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP8266 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • MQTT-Explorer öffnen • MQTT-Request unter smart-room/request/temperature mit Inhalt "request" absetzen 		
Erwartung:	Antwort per MQTT unter smart-room/response/temperature "Raumtemperatur" wird abgesetzt		
Output:	Keine Antwort per MQTT unter smart-shelf/response/temperature "Raumtemperatur"		
Ergebnis:	Nicht erfolgreich		
Konsequenz:	<p>Untersuchung der Fehlerursache: Sensor DHT11 fehlerhaft</p> <p>Behebung der Fehlerursache: Austausch des Sensors DHT11</p>		

Tabelle 35: Test 3.5

Test 3.6			
Test-Datum:	26.01.2023	System-Datum:	26.01.2023
Tester:	Andrea Wieser		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 		
Testgegenstand:	ESP-8266		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe der Temperatur des Klimasensors per MQTT-Response unter smart-room/response/temperature		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP8266 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • MQTT-Explorer öffnen • MQTT-Request unter smart-room/request/temperature mit Inhalt "request" absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Antwort per MQTT unter smart-room/response/temperature "Raumtemperatur" wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Antwort per MQTT unter smart-shelf/response/temperature "Raumtemperatur" wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Temperatur des Klimasensors wird korrekt zurückgegeben		

Tabelle 36: Test 3.6

Test 3.7			
Test-Datum:	26.01.2023	System-Datum:	26.01.2023
Tester:	Andrea Wieser		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 		
Testgegenstand:	ESP-8266		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe der Luftfeuchtigkeit des Klimasensors per MQTT-Response unter smart-room/response/humidity		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP8266 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • MQTT-Explorer öffnen • MQTT-Request unter smart-room/request/humidity mit Inhalt "request" absetzen 		
Erwartung:	Antwort per MQTT unter smart-room/response/humidity "Raumluftfeuchtigkeit" wird abgesetzt		
Output:	Antwort per MQTT unter smart-shelf/response/humidity "Raumluftfeuchtigkeit" wird abgesetzt		
Ergebnis:	Erfolgreich		
Konsequenz:	Luftfeuchtigkeit des Klimasensors wird korrekt zurückgegeben		

Tabelle 37: Test 3.7

Test 4.1			
Test-Datum:	05.02.2023	System-Datum:	05.02.2023
Tester:	Berkan Kalkan		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ A per Licht und MQTT-Response unter smart_shelf/response		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI über null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart_shelf/request mit Inhalt <code>{"processID": "ProzessID", "status": "A"}</code> absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Available"}</code> wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Available"}</code> wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status der Box wird korrekt zurückgegeben		

Tabelle 38: Test 4.1

Test 4.2			
Test-Datum:	05.02.2023	System-Datum:	05.02.2023
Tester:	Berkan Kalkan		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ A per Licht und MQTT-Response unter smart_shelf/response		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI auf null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart_shelf/request mit Inhalt <code>{"processID": "ProzessID", "status": "A"}</code> absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Gelbes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Incomplete"}</code> wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Gelbes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Incomplete"}</code> wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status der Box wird korrekt zurückgegeben		

Tabelle 39: Test 4.2

Test 4.3			
Test-Datum:	05.02.2023	System-Datum:	05.02.2023
Tester:	Berkan Kalkan		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ A per Licht und MQTT-Response unter smart_shelf/response		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI über null setzen und Box nicht vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart_shelf/request mit Inhalt <code>{"processID": "ProzessID", "status": "A"}</code> absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Rotes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Empty"}</code> wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Kein Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Empty"}</code> wird abgesetzt 		
Ergebnis:	Nicht erfolgreich		
Konsequenz:	<p>Untersuchung der Fehlerursache: Licht-Pin nicht korrekt eingesteckt am Mikrocontroller</p> <p>Behebung der Fehlerursache: Licht-Pin am Mikrocontroller anstecken</p>		

Tabelle 40: Test 4.3

Test 4.4			
Test-Datum:	05.02.2023	System-Datum:	05.02.2023
Tester:	Berkan Kalkan		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ A per Licht und MQTT-Response unter smart_shelf/response		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI über null setzen und Box nicht vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart_shelf/request mit Inhalt <code>{"processID": "ProzessID", "status": "A"}</code> absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Rotes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Empty"}</code> wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Rotes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Empty"}</code> wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status der Box wird korrekt zurückgegeben		

Tabelle 41: Test 4.4

Test 5.1			
Test-Datum:	05.02.2023	System-Datum:	05.02.2023
Tester:	Berkan Kalkan		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ B per Licht und MQTT-Response unter smart_shelf/response		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI über null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart_shelf/request mit Inhalt <code>{"processID": "ProzessID", "status": "B"}</code> absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Available"}</code> wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Available"}</code> wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status der Box wird korrekt zurückgegeben		

Tabelle 42: Test 5.1

Test 5.2			
Test-Datum:	05.02.2023	System-Datum:	05.02.2023
Tester:	Berkan Kalkan		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ B per Licht und MQTT-Response unter smart_shelf/response		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI über null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart_shelf/request mit Inhalt <code>{"processID": "ProzessID", "status": "B"}</code> absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Incomplete"}</code> wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Incomplete"}</code> wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status der Box wird korrekt zurückgegeben		

Tabelle 43: Test 5.2

Test 5.3			
Test-Datum:	05.02.2023	System-Datum:	05.02.2023
Tester:	Berkan Kalkan		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP32 		
Testgegenstand:	ESP-32		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status von Verpackungsboxen des Typ B per Licht und MQTT-Response unter smart_shelf/response		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP32 mit aktuellen Netzwerddaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Anzahl an Boxen in UI über null setzen und Box vor Infrarotsensor positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart_shelf/request mit Inhalt <code>{"processID": "ProzessID", "status": "B"}</code> absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Empty"}</code> wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Grünes Licht leuchtet auf • Antwort per MQTT unter smart_shelf/response : <code>{"processID": "ProzessID", "status": "Empty"}</code> wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status der Box wird korrekt zurückgegeben		

Tabelle 44: Test 5.3

Test 6.1			
Test-Datum:	05.02.2023	System-Datum:	05.02.2023
Tester:	Betim Sulejmani		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 		
Testgegenstand:	ESP-8266		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status des Türsensors per MQTT-Response unter smart_room/response/door		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP8266 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Box vor Infrarotsensor für Türerkennung positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart_room/request/door mit Inhalt {„processID“:”ProzessID”, „status“:”request”} absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Antwort per MQTT unter smart_room/response/door : {„processID“:”ProzessID”, „status“:”1”} wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Antwort per MQTT unter smart_room/response/door : {„processID“:”ProzessID”, „status“:”1”} wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status des Türsensors wird korrekt zurückgegeben		

Tabelle 45: Test 6.1

Test 6.2			
Test-Datum:	05.02.2023	System-Datum:	05.02.2023
Tester:	Betim Sulejmani		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 		
Testgegenstand:	ESP-8266		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe des Status des Türsensors per MQTT-Response unter smart_room/response/door		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP8266 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Box nicht vor Infrarotsensor für Türerkennung positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart_room/request/door mit Inhalt <code>{"processID": "ProzessID", "status": "request"}</code> absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Antwort per MQTT unter smart_room/response/door : <code>{"processID": "ProzessID", "status": "0"}</code> wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Antwort per MQTT unter smart_room/response/door : <code>{"processID": "ProzessID", "status": "0"}</code> wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Status des Türsensors wird korrekt zurückgegeben		

Tabelle 46: Test 6.2

Test 6.3			
Test-Datum:	05.02.2023	System-Datum:	05.02.2023
Tester:	Betim Sulejmani		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 		
Testgegenstand:	ESP-8266		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe der Temperatur des Klimasensors per MQTT-Response unter smart_room/response/temperature		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP8266 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Box vor Infratotsensor für Türerkennung positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart_room/request/temperature mit Inhalt {„processID“:”ProzessID”, „status“:”request”} absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Antwort per MQTT unter smart_room/response/temperature : {„processID“:”ProzessID”, „status“:”Raumtemperatur”} wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Antwort per MQTT unter smart_room/response/temperature : {„processID“:”ProzessID”, „status“:”Raumtemperatur”} wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Temperatur des Klimasensors wird korrekt zurückgegeben		

Tabelle 47: Test 6.3

Test 6.4			
Test-Datum:	05.02.2023	System-Datum:	05.02.2023
Tester:	Betim Sulejmani		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 		
Testgegenstand:	ESP-8266		
Beschreibung:	Test zum Sicherstellen der richtigen Rückgabe der Luftfeuchtigkeit des Klimasensors per MQTT-Response unter smart_room/response/humidity		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller ESP8266 mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren 		
Testablauf:	<ul style="list-style-type: none"> • Box vor Infrarotsensor für Türerkennung positionieren • MQTT-Explorer öffnen • MQTT-Request unter smart_room/request/humidity mit Inhalt <code>{"processID": "ProzessID", "status": "request"}</code> absetzen 		
Erwartung:	<ul style="list-style-type: none"> • Antwort per MQTT unter smart_room/response/humidity: <code>{"processID": "ProzessID", "status": "Raumluftfeuchtigkeit"}</code> wird abgesetzt 		
Output:	<ul style="list-style-type: none"> • Antwort per MQTT unter smart_room/response/humidity: <code>{"processID": "ProzessID", "status": "Raumluftfeuchtigkeit"}</code> wird abgesetzt 		
Ergebnis:	Erfolgreich		
Konsequenz:	Luftfeuchtigkeit des Klimasensors wird korrekt zurückgegeben		

Tabelle 48: Test 6.4

Test 7.1			
Test-Datum:	07.02.2023	System-Datum:	07.02.2023
Tester:	Matthias Schwarz Ozan Akkoyun		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 • ESP32 		
Testgegenstand:	Camunda-Showcase Prozess		
Beschreibung:	Test zum Sicherstellen der korrekten Funktionalität des Teilsystems (durch den Camunda-Showcase Prozess)		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren • Camunda-Showcase Prozess modellieren 		
Testablauf:	<ul style="list-style-type: none"> • Boxen vor Infrarotsensoren positionieren • Schritt für Schritt Camunda-Showcase Prozess durchführen 		
Erwartung:	<ul style="list-style-type: none"> • Fehlerlose Durchführung des Prozesses bis zum Abschluss 		
Output:	<ul style="list-style-type: none"> • Camunda-Showcase Prozess kann nicht Schritt für Schritt durchgeführt werden 		
Ergebnis:	Nicht erfolgreich		
Konsequenz:	<p>Untersuchung der Fehlerursache: Inkorrekte Message-Expression bei Throw-Message verwendet</p> <p>Behebung der Fehlerursache: Korrekte Message-Expression bei Throw-Message eingefügt</p>		

Tabelle 49: Test 7.1

Test 7.2			
Test-Datum:	08.02.2023	System-Datum:	08.02.2023
Tester:	Matthias Schwarz Ozan Akkoyun		
Testinfrastruktur:	<ul style="list-style-type: none"> • PC • ESP8266 • ESP32 		
Testgegenstand:	Camunda-Showcase Prozess		
Beschreibung:	Test zum Sicherstellen der korrekten Funktionalität des Teilsystems (durch den Camunda-Showcase Prozess)		
Vorbereitung:	<ul style="list-style-type: none"> • Mikrocontroller mit aktuellen Netzwerkdaten flashen • WLAN-Access Point starten • MQTT-Broker starten • Node-RED Instanz starten und konfigurieren • Camunda-Showcase Prozess modellieren 		
Testablauf:	<ul style="list-style-type: none"> • Boxen vor Infarotsensoren positionieren • Schritt für Schritt Camunda-Showcase Prozess durchführen 		
Erwartung:	<ul style="list-style-type: none"> • Fehlerloser Durchführung des Prozesses bis zum Abschluss 		
Output:	<ul style="list-style-type: none"> • Fehlerloser Durchführung des Prozesses bis zum Abschluss 		
Ergebnis:	Erfolgreich		
Konsequenz:	Camunda-Showcase Prozess funktioniert korrekt		

Tabelle 50: Test 7.2

5. Technisches Handbuch

Im folgenden Kapitel werden die verwendete Hardware, die Verschaltung der Teilprojekte Smart-shelf und Smart-room und die Schnittstellen zwischen den einzelnen technischen Komponenten beschrieben.

5.1 Smart-shelf

5.1.1 Verwendete Hardware

ESP 32 (ESP-WROOM-32):

Beim ESP32 handelt es sich um einen kostengünstigen 32Bit- Microcontroller, welcher sich durch seine offene Bauweise und zahlreichen Input- und Output Pins optimal für schlichte Schaltungen mit Sensoren und Aktuatoren eignet. Microcontroller der ESP32- Serie verfügen entweder über einen internen Flashspeicher oder werden durch externe Flashspeicher erweitert.

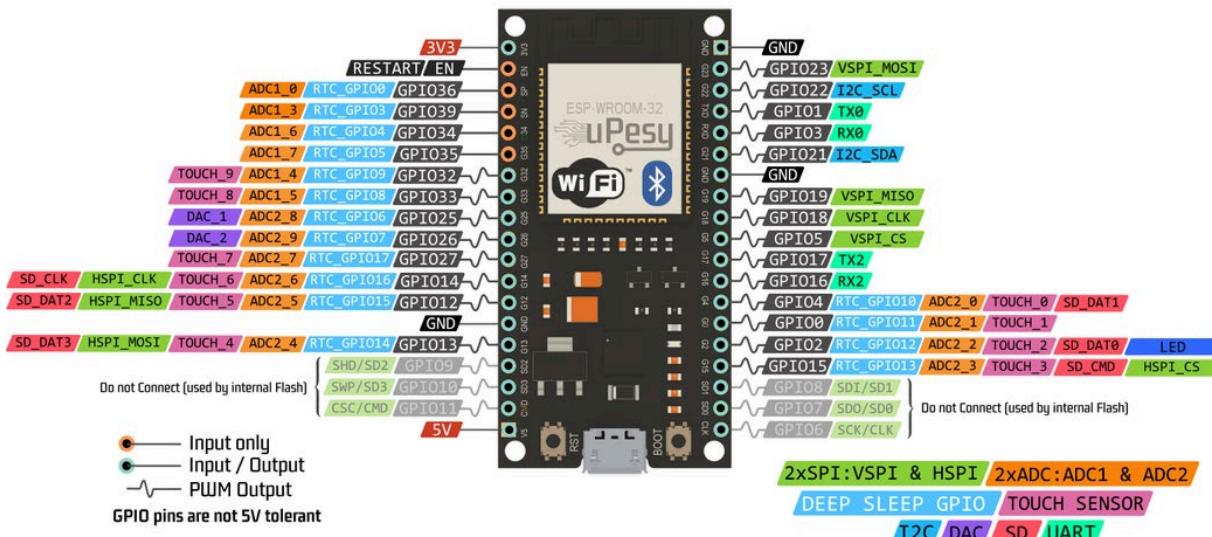


Abbildung 4: ESP32 Pinbelegung

Unser ESP 32 verfügt über 38 Pins, welche größtenteils für unterschiedliche Zwecke konfiguriert und beschalten werden können. In unserer konkreten Umsetzung wurden folgende Pin-Typen verwendet:

3V3	Versorgung des ESP mit 3,3V
GND	Versorgung und Masse des ESP
GPIO (konfiguriert als Input)	Als Eingang für Daten von Sensoren
GPIO (konfiguriert als Output)	Als Ausgang zum Schalten von Aktuatoren (LEDs)
I2C_SCL & I2C_SDA	Als Ausgänge zum Schalten des I2C-Displays

Tabelle 51: Verwendung der Pins am ESP32

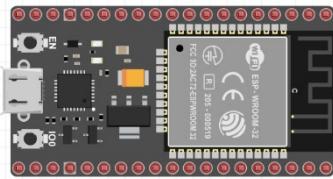
Hardware	Abbildung Schaltplan	Abbildung Steckplatine
		

Tabelle 52: Abbildungen ESP32

Die genaue Beschaltung mit den einzelnen Komponenten wird im Folgenden genauer beschrieben oder kann im Schaltplan betrachtet werden.

Weitere Informationen zum verwendeten ESP32 sind im zugehörigem Datenblatt abrufbar:

https://cdn.shopify.com/s/files/1/1509/1638/files/ESP_-_32_NodeMCU_Developmentboard_Datenblatt_AZ-Delivery_Vertriebs_GmbH_10f68f6c-a9bb-49c6-a825-07979441739f.pdf?v=1598356497

Versorgung (9V):

Als Versorgung dient eine handelsübliche 9V- Blockbatterie oder wie in unserem Fall ein 9V-Akku. Dieser wird mittels einem passenden Batterie-Clip mit der Steckplatine und den weiteren Komponenten verbunden.

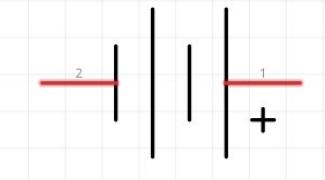
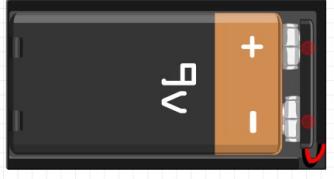
Hardware	Abbildung Schaltplan	Abbildung Steckplatine
		

Tabelle 53: Abbildungen 9V- Batterie/ Akku

Spannungswandler:

Da wir von der Batterie eine Spannung von 9V erhalten, unser ESP und die restlichen Komponenten aber mit 3,3V beschalten werden müssen, wurde ein Spannungswandler verbaut welcher die Eingangsspannung von 9V auf eine Ausgangsspannung von 3,3V transformiert.

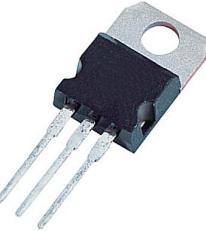
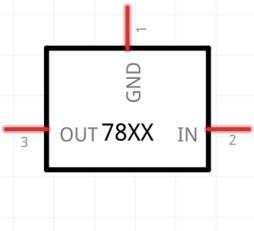
Hardware	Abbildung Schaltplan	Abbildung Steckplatine
		

Tabelle 54: Abbildungen Spannungswandler

Weitere Informationen zum verwendeten LF33- Spannungswandler sind im zugehörigem Datenblatt abrufbar: https://cdn-reichelt.de/documents/datenblatt/A200/LF33CDT_LF33CV_LF50CV%23STM.pdf

Beschaltung	
GND	Minus-Pol der Batterie
IN	Plus-Pol der Batterie
OUT	Plus- Leiste der Steckplatine & 3V3 am ESP

Tabelle 55: Beschaltung des Spannungswandlers

I2C- Display:

Bei einem I2C- Display handelt es sich um ein Display, welches mittels I2C- Bus angesteuert werden kann. Der I2C- Standard ist ein Datenbus, welcher es bis zu 128 Teilnehmern ermöglicht über nur zwei Datenleitungen zu kommunizieren. Die serielle Datenübertragung wird dabei durch den I2C- Standard definiert und geregelt. In unserem Fall wird am Display ausgegeben, wie viele Stück von den jeweiligen Boxtypen laut System aktuell vorhanden sind.

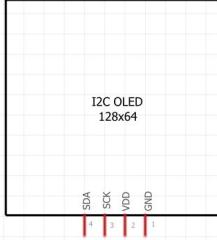
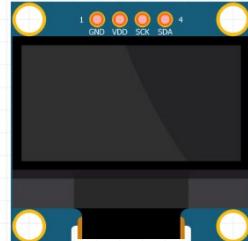
Hardware	Abbildung Schaltplan	Abbildung Steckplatine
		

Tabelle 56: Abbildungen Display

Weitere Informationen zum verwendeten I2C-Display sind im zugehörigem Datenblatt abrufbar: https://cdn.shopify.com/s/files/1/1509/1638/files/0_96_Zoll_Display_Datenblatt_AZ-Delivery_Vertriebs_GmbH_241c4223-c03f-4530-a8c0-f9ef2575872f.pdf?v=1622442722

Beschaltung	
GND	Masse via Steckplatine
VDD	3,3V via Steckplatine
SCK	GPIO22 des ESP als I2C_SCK konfiguriert
SDA	GPIO21 des ESP als I2C_SDA konfiguriert

Tabelle 57: Beschaltung des I2C-Displays

Infrarotsensor:

Infrarot-Sensoren bestehen grundlegend aus 2 Dioden. Eine Emitterdiode gibt Infrarotstrahlung ab und einer Empfängerdiode die Infrarotstrahlung erkennt. Steht nun ein physikalisches Objekt vor den beiden Dioden wird die Infrarotstrahlung des Emitters reflektiert und fällt zurück zur Empfängerdiode, welche diese Information am Ausgang bereitstellt. Die tatsächlich verwendeten Infrarot-Sensoren haben sich im Verlauf des Projekts immer wieder geändert da wir anfangs nicht genau wussten welche Sensoren uns genau zur Verfügung stehen und sich herausstellte da manche nicht oder nur sehr schlecht funktionierten. Die Funktion sowie die Anschlüsse der verschiedenen Typen unterscheiden sich allerdings nicht.

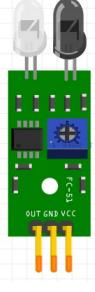
Hardware	Abbildung Schaltplan	Abbildung Steckplatine
		

Tabelle 58: Abbildungen Infrarotsensor

Beschaltung Infrarotsensor Typ A	
GND	Masse der Steckplatine
VCC	3,3V der Steckplatine
OUT	GPIO17 des ESP als Digital Input konfiguriert
Beschaltung Infrarotsensor Typ B	
GND	Masse der Steckplatine
VCC	3,3V der Steckplatine
OUT	GPIO27 des ESP als Digital Input konfiguriert

Tabelle 59: Beschaltung Infrarotsensoren

Weitere Informationen zum den verwendeten IR-Sensoren sind in den zugehörigen Datenblättern abrufbar:

Typ1: https://cdn.shopify.com/s/files/1/1509/1638/files/Hindernis_Sensor_Datenblatt_1.pdf?4459799226942350178

Typ2: https://cdn.shopify.com/s/files/1/1509/1638/files/IR-Abstand_Sensor_Modul_Datenblatt_ecbf7c3f-92d9-4242-b75b-ab8986f5c47b.pdf?v=1605115975

LED-Ampel:

Das LED-Ampelmodul besteht aus 3 farbigen LEDs, welche über eine gemeinsame Masse und jeweils einen Versorgungspin angesteuert werden. In unserem Anwendungsfall sind zwei LED-Ampeln verbaut für die zwei verschiedenen Boxtypen verbaut. Dabei leuchtet jeweils eine LED und signalisiert so den Status des Boxentyps.

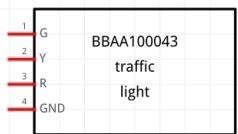
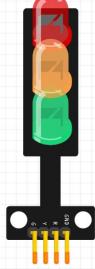
Hardware	Abbildung Schaltplan	Abbildung Steckplatine
	 BBA100043 traffic light	

Tabelle 60: Abbildungen LED-Ampel

Beschaltung LED Typ A	
GND	Masse via Steckplatine
R (rot)	GPIO4 des ESP als Digital Output konfiguriert
Y (gelb)	GPIO2 des ESP als Digital Output konfiguriert
G (grün)	GPIO0 des ESP als Digital Output konfiguriert
Beschaltung LED Typ B	
GND	Masse via Steckplatine
R (rot)	GPIO19 des ESP als Digital Output konfiguriert
Y (gelb)	GPIO18 des ESP als Digital Output konfiguriert
G (grün)	GPIO16 des ESP als Digital Output konfiguriert

Tabelle 61: Beschaltung LED-Ampeln

Weitere Informationen zu der verwendeten LED-Ampel sind im zugehörigem Datenblatt abrufbar: https://cdn.shopify.com/s/files/1/1509/1638/files/LED_Ampel_Modul_Datenblatt_AZ-Delivery_Vertriebs_GmbH.pdf?v=1607630369

5.1.2 Schaltplan:

Der Schaltplan wurde mit dem Tool Fritzing Beta, welches in der Version 0.9.3 kostenlos verfügbar ist, erstellt. Da nicht alle im Projekt verwendeten Bauteile verfügbar waren wurde für die Darstellung in der Software teilweise auf baugleiche Alternativen zurückgegriffen.

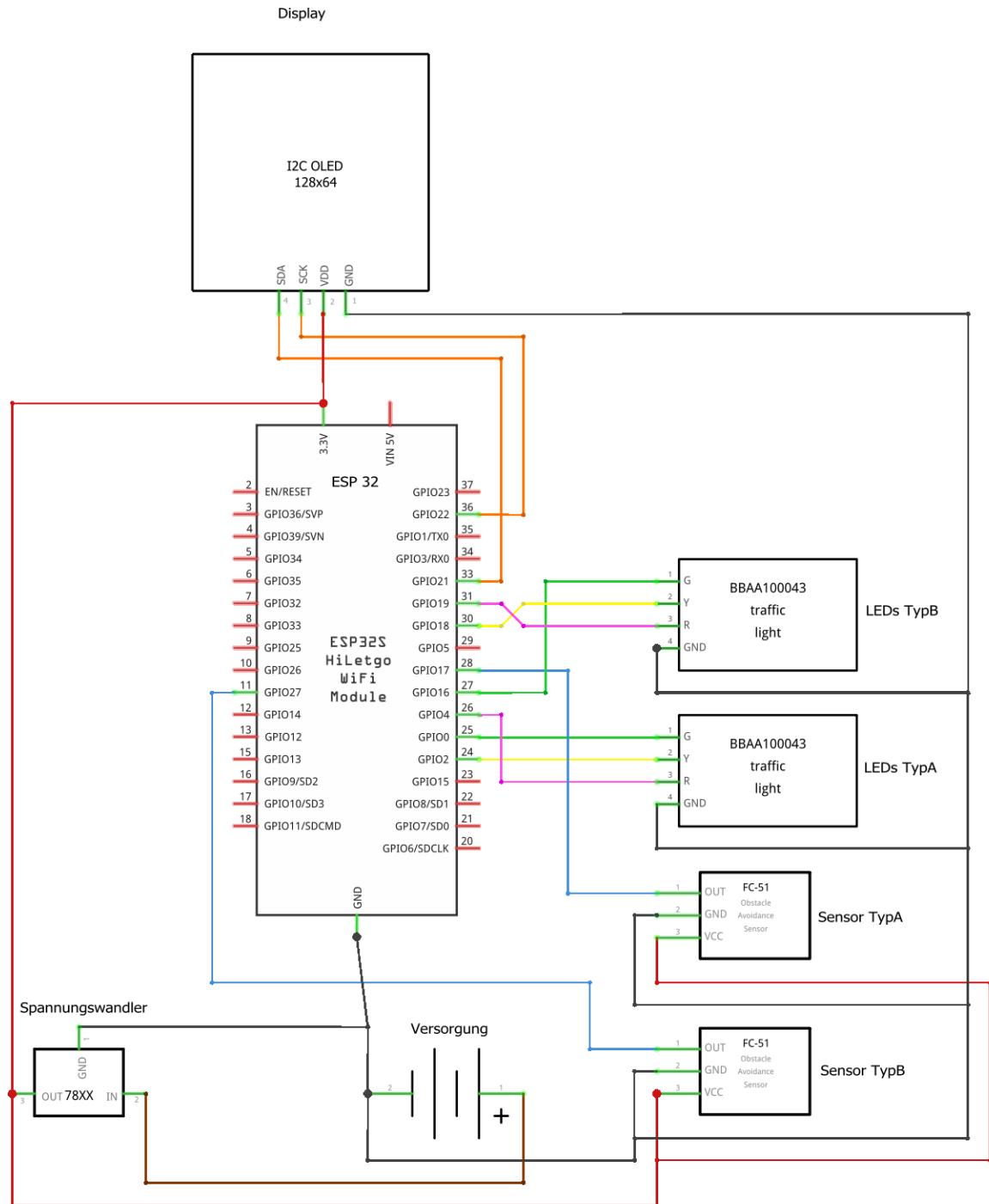


Abbildung 5: Schaltplan Smart-shelf

5.1.3 Steckplatine:

Die Steckplatine wurde mit dem Tool Fritzing Beta, welches in der Version 0.9.3 kostenlos verfügbar ist, erstellt. Da nicht alle im Projekt verwendeten Bauteile verfügbar waren wurde für die Darstellung in der Software teilweise auf baugleiche Alternativen zurückgegriffen. Die Farben und die exakte Verdrahtung ist ebenfalls so übersichtlich wie möglich gestaltet und nicht 1:1 wie bei unserem physischen Prototypen. Die Logik und Funktion ist allerdings exakt gleich.

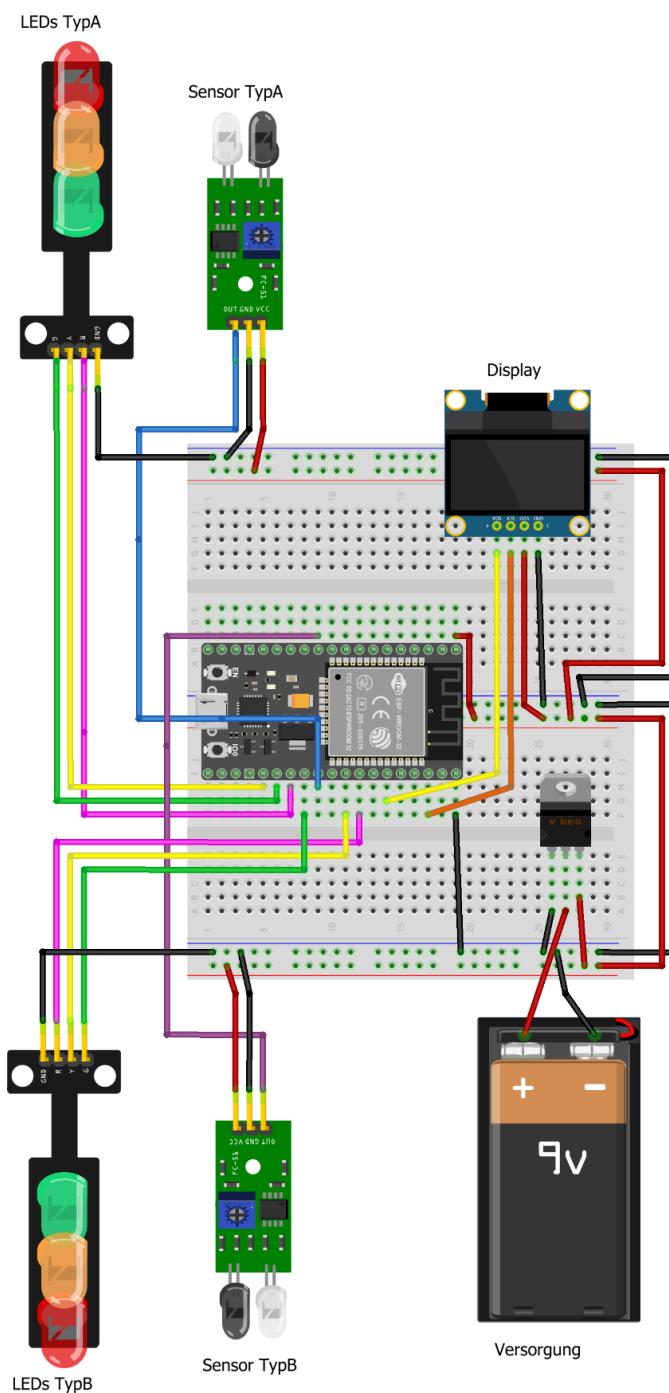


Abbildung 6: Steckplatine Smart-shelf

5.2 Smart-room

5.2.1 Verwendete Hardware

ESP 8266 (ESP8266 D1 Mini):

Beim ESP8266 handelt es sich um den Vorgänger des bereits erwähnten ESP32 und somit ebenfalls um einen kostengünstigen 32Bit-Microcontroller, welcher sich durch seine offene Bauweise und zahlreichen Input- und Output-Pins optimal für schlichte Schaltungen mit Sensoren und Aktuatoren eignet. Anders als bei ihren Nachfolgern ist bei Microcontrollern der ESP8266-Familie kein interner Flash-Speicher vorhanden.

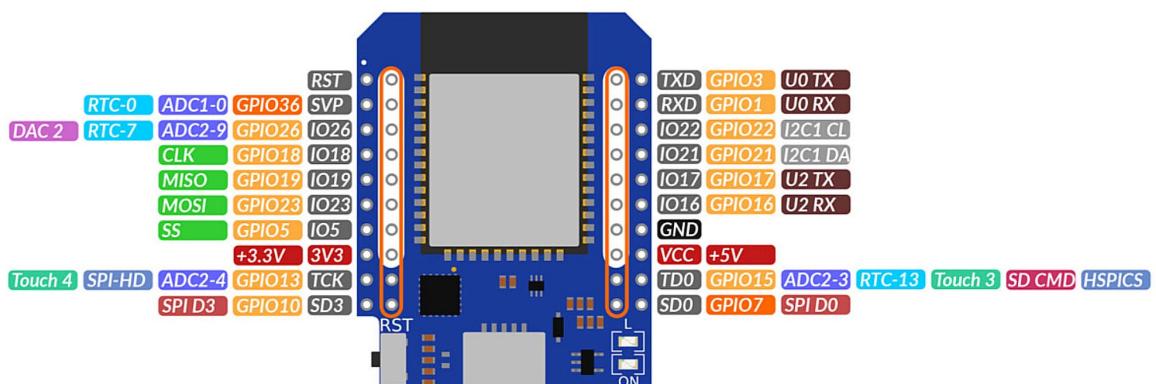


Abbildung 7: ESP8266 Pinbelegung

Unser ESP 8266 verfügt über 20 Pins, welche größtenteils für unterschiedliche Zwecke konfiguriert und beschalten werden können. In unserer konkreten Umsetzung wurden folgende Pin-Typen verwendet:

VCC/+5V	Versorgung des ESP mit 3,3V
GND	Versorgung und Masse des ESP
+3,3V/ 3V3	Als Ausgangsversorgung für weitere Komponenten
GPIO (konfiguriert als Input)	Als Eingang für Signal von Infrarotsensor
GPIO (konfiguriert als 1-Wire Bus)	Als Eingang für Daten von Temperatur und Feuchtigkeitssensor

Tabelle 62: Verwendung der Pins am ESP32

Die genaue Beschaltung mit den einzelnen Komponenten wird im Folgenden genauer beschrieben oder kann im Schaltplan betrachtet werden.

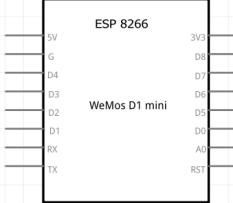
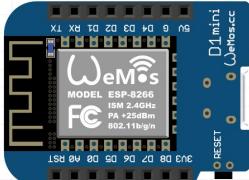
Hardware	Abbildung Schaltplan	Abbildung Steckplatine																
	 <p>ESP 8266 WeMos D1 mini</p> <table border="1"> <tr><td>5V</td><td>3V3</td></tr> <tr><td>G</td><td>D8</td></tr> <tr><td>D4</td><td>D7</td></tr> <tr><td>D3</td><td>D6</td></tr> <tr><td>D2</td><td>D5</td></tr> <tr><td>D1</td><td>D0</td></tr> <tr><td>RX</td><td>A0</td></tr> <tr><td>TX</td><td>RST</td></tr> </table>	5V	3V3	G	D8	D4	D7	D3	D6	D2	D5	D1	D0	RX	A0	TX	RST	
5V	3V3																	
G	D8																	
D4	D7																	
D3	D6																	
D2	D5																	
D1	D0																	
RX	A0																	
TX	RST																	

Tabelle 63: Abbildungen ESP8266

Weitere Informationen zum verwendeten ESP8266 sind im zugehörigem Datenblatt abrufbar:

https://cdn.shopify.com/s/files/1/1509/1638/files/Betriebsanleitung-AZ-D1miniV1.2_2.pdf?v=1590603445

Versorgung (9V):

Als Versorgung dient eine handelsübliche 9V- Blockbatterie oder wie in unserem Fall ein 9V- Akku. Dieser wird mittels einem passenden Batterie-Clip mit der Steckplatine und den weiteren Komponenten verbunden.

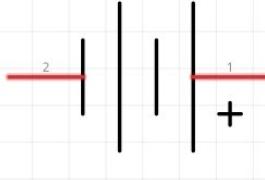
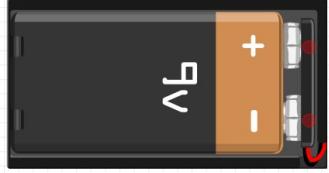
Hardware	Abbildung Schaltplan	Abbildung Steckplatine
		

Tabelle 64: Abbildungen 9V- Batterie/ Akku

Spannungswandler:

Da wir von der Batterie eine Spannung von 9V erhalten, unser ESP und die restlichen Komponenten aber mit 3,3V beschalten werden müssen, wurde ein Spannungswandler verbaut welcher die Eingangsspannung von 9V auf eine Ausgangsspannung von 3,3V transformiert.

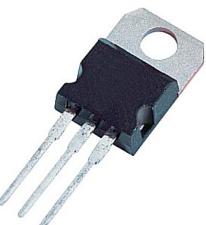
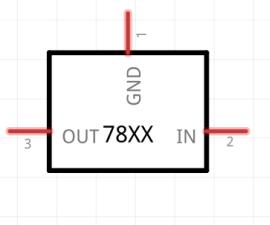
Hardware	Abbildung Schaltplan	Abbildung Steckplatine
		

Tabelle 65: Abbildungen Spannungswandler

Beschaltung	
GND	Minus-Pol der Batterie
IN	Plus-Pol der Batterie
OUT	3V3 am ESP

Tabelle 66: Beschaltung des Spannungswandlers

Weitere Informationen zum verwendeten LF33- Spannungswandler sind im zugehörigem Datenblatt abrufbar: https://cdn-reichelt.de/documents/datenblatt/A200/LF33CDT_LF33CV_LF50CV%23STM.pdf

Infrarotsensor:

Infrarot-Sensoren bestehen grundlegend aus 2 Dioden. Eine Emitterdiode gibt Infrarotstrahlung ab und einer Empfängerdiode die Infrarotstrahlung erkennt. Steht nun ein physikalisches Objekt vor den beiden Dioden wird die Infrarotstrahlung des Emitters reflektiert und fällt zurück zur Empfängerdiode, welche diese Information am Ausgang bereitstellt. Die tatsächlich verwendeten Infrarot-Sensoren haben sich im Verlauf des Projekts immer wieder geändert da wir anfangs nicht genau wussten welche Sensoren uns genau zur Verfügung stehen und sich herausstellte da manche nicht oder nur schlecht funktionierten. Die Funktion sowie die Anschlüsse der verschiedenen Typen unterscheiden sich allerdings nicht.

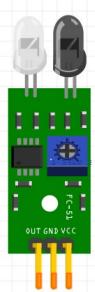
Hardware	Abbildung Schaltplan	Abbildung Steckplatine
		

Tabelle 67: Abbildungen Infrarotsensor

Beschaltung Infrarotsensor Typ A	
GND	Masse des ESP
VCC	3,3V des ESP
OUT	D7 (GPIO23) des ESP als Digital Input konfiguriert

Tabelle 68: Beschaltung Infrarotsensoren

Weitere Informationen zum den verwendeten IR-Sensoren sind in den zugehörigen Datenblättern abrufbar:

Typ1: https://cdn.shopify.com/s/files/1/1509/1638/files/Hindernis_Sensor_Datenblatt_1.pdf?4459799226942350178

Typ2: https://cdn.shopify.com/s/files/1/1509/1638/files/IR-Abstand_Sensor_Modul_Datenblatt_ecbf7c3f-92d9-4242-b75bab8986f5c47b.pdf?v=1605115975

Temperatur- und Feuchtigkeitssensor

Beim Verbauten Temperatur- und Feuchtigkeitssensor handelt es sich um einen DHT-11 Sensor. Dieser verfügt über 4 im Gehäuse eingelassene Pins, von denen allerdings nur 3 genutzt werden. Dabei werden 2 Pins für Versorgung und Masse beansprucht. Der Dritte funktionale Pin liefert Daten über Temperatur und Feuchtigkeit an den ESP. Die Kommunikation erfolgt über 1-Wire Bus. Die Werte werden dabei in Grad Celsius für die Temperatur und in Prozent für die Luftfeuchtigkeit übertragen.

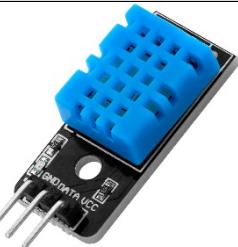
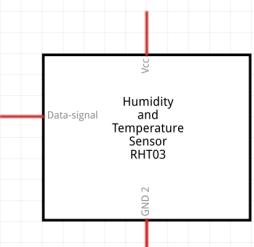
Hardware	Abbildung Schaltplan	Abbildung Steckplatine
		

Tabelle 69: Abbildungen Temperatur- und Feuchtigkeitssensor

Beschaltung Temperatur- und Feuchtigkeitssensor	
GND	Masse des ESP
VCC	3,3V des ESP
Data-Signal	D8 (GPIO5) als 1-Wire-Bus konfiguriert

Tabelle 70: Beschaltung Temperatur- und Feuchtigkeitssensor

Weitere Informationen zum verwendeten DHT-11- Sensors sind im zugehörigem Datenblatt abrufbar: https://cdn.shopify.com/s/files/1/1509/1638/files/DHT_22_AM2302_Temperatur_und_Luftfeuchtigkeitssensor_Datenblatt.pdf?11983326290748777409

5.2.2 Schaltplan:

Der Schaltplan wurde mit dem Tool Fritzing Beta, welches in der Version 0.9.3 kostenlos verfügbar ist, erstellt. Da nicht alle im Projekt verwendeten Bauteile verfügbar waren wurde für die Darstellung in der Software teilweise auf baugleiche Alternativen zurückgegriffen.

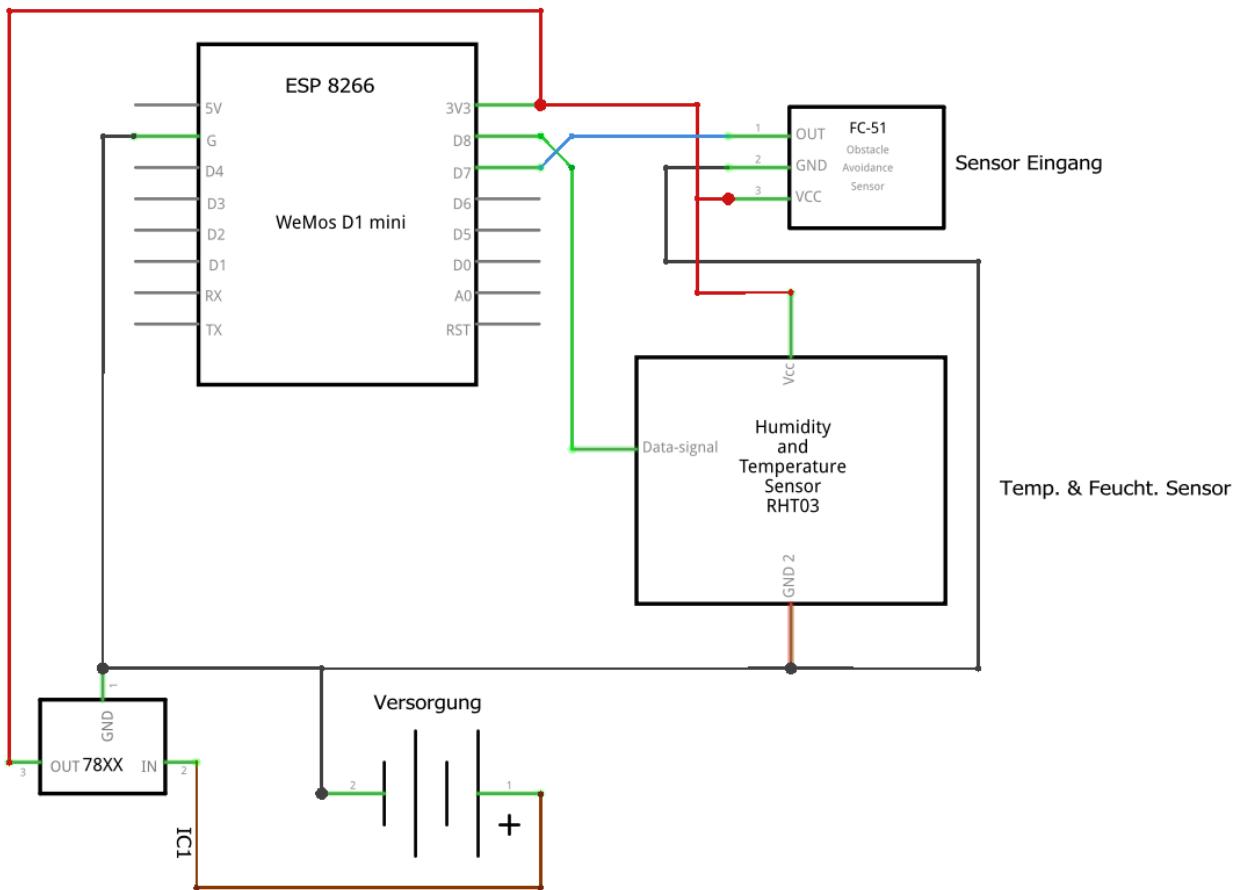


Abbildung 8: Schaltplan Smart-shelf

5.2.3 Steckplatine:

Die Steckplatine wurde mit dem Tool Fritzing Beta, welches in der Version 0.9.3 kostenlos verfügbar ist, erstellt. Da nicht alle im Projekt verwendeten Bauteile verfügbar waren wurde für die Darstellung in der Software teilweise auf baugleiche Alternativen zurückgegriffen. Die Farben und die exakte Verdrahtung ist ebenfalls so übersichtlich wie möglich gestaltet und nicht 1:1 wie bei unserem physischen Prototypen. Die Logik und Funktion ist allerdings exakt gleich.

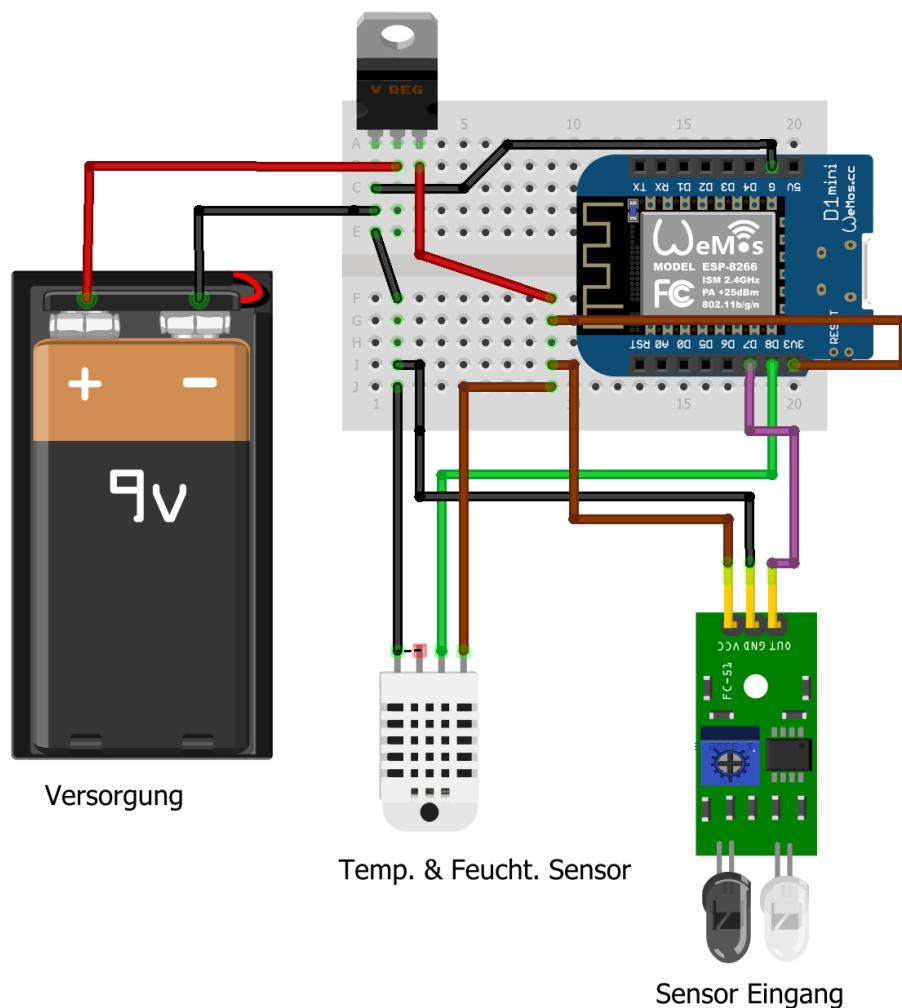


Abbildung 9: Steckplatine Smart-room

5.2.4 Tipp zur Inbetriebnahme:

Bei der Inbetriebnahme der Hardware mit der Batterie als Versorgung, kann es vorkommen, dass der ESP8266 nicht ordnungsgemäß starten und keine WiFi-Verbindung aufbaut. Das liegt unserer Einschätzung nach daran, dass beim Boot-Vorgang kurzzeitig mehr Leistung benötigt wird, als der Spannungswandler liefern kann.

Das Problem kann mit folgenden 3 Schritten umgangen werden:

1. Die Versorgungspins der beiden Sensoren bei abgeklemmter Batterie vom Board trennen.
2. Batterie an das Board anklemmen und 5 Sekunden warten.
3. Die Versorgungspins der beiden Sensoren wieder verbinden.

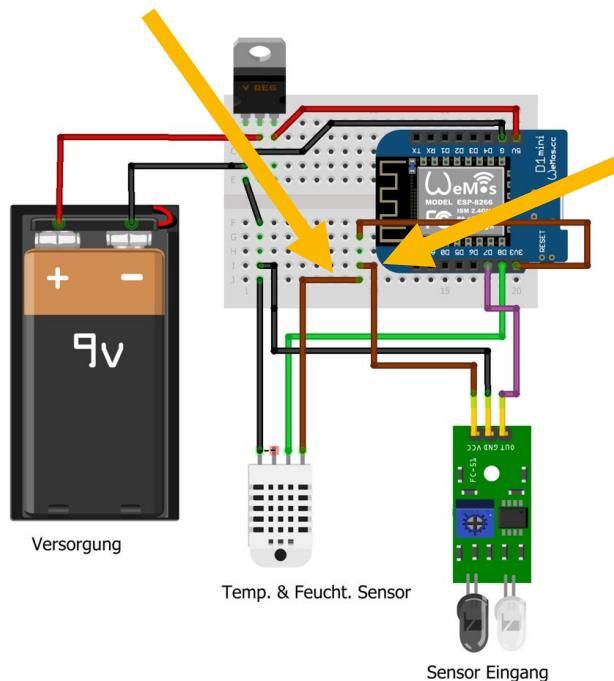


Abbildung 10: Versorgungspins der Sensoren

5.3 Architektur

Die Gesamtarchitektur besteht aus den drei folgenden Teilprojekten:

- Hardware Smart-shelf
- Hardware Smart-room
- Konfiguration und Webinterface mittels Node RED

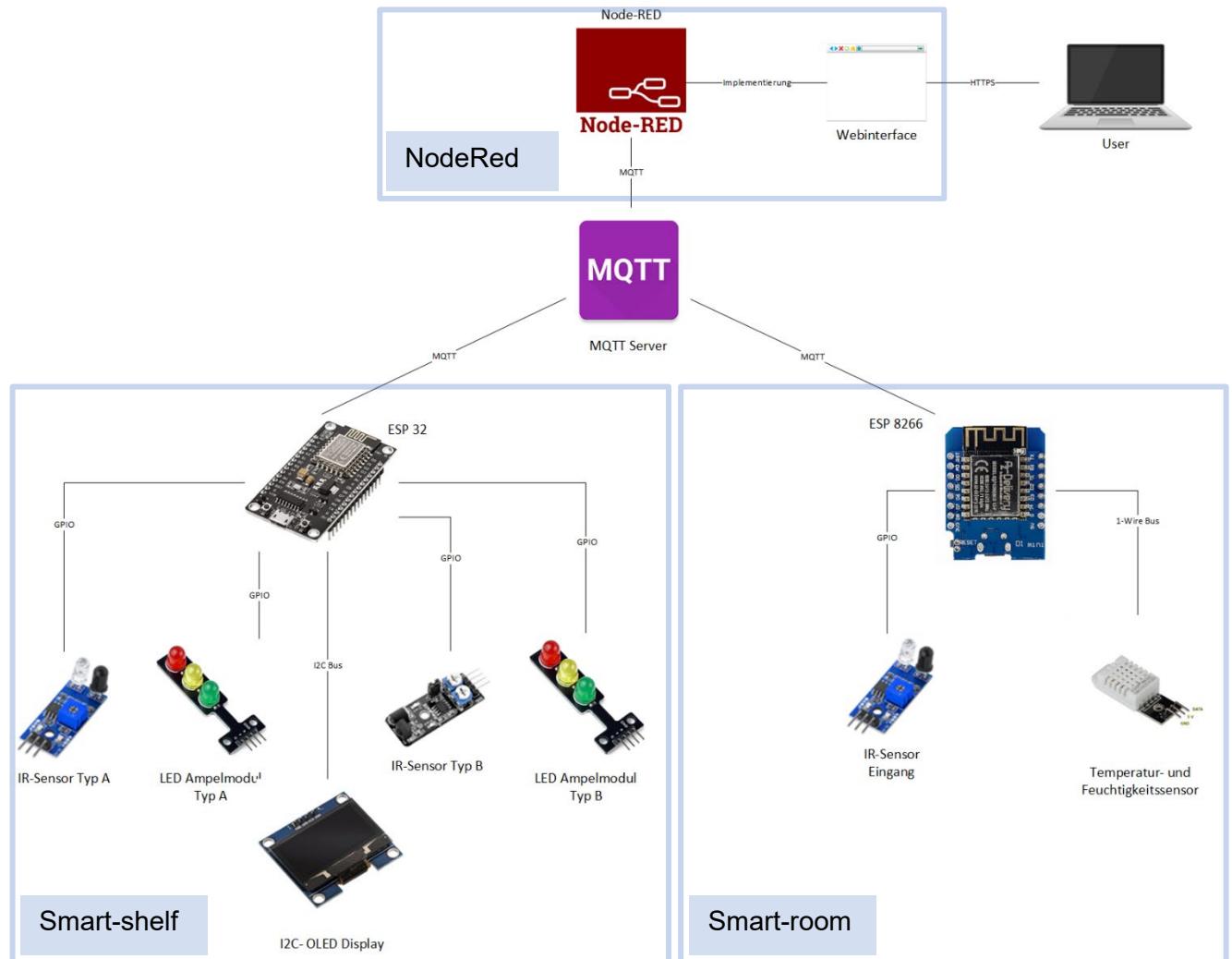


Abbildung 11: Gesamtarchitektur Smart-shelf & Smart-room

Wie in Abbildung 7 ersichtlich gehen Anfragen an das System vom MQTT-Server (Broker) ein und werden von einer Node-RED Instanz bearbeitet. Diese Anfragen werden mittels MQTT übertragen. In der Node-RED Instanz wird nach Art der Anfrage unterschieden und im ersten Schritt erkannt, ob eine Information vom Smart-shelf oder dem Smart-room benötigt wird. Je nach gewünschter Information wird der jeweilige ESP mittels Wifi- Verbindung angesprochen. Dieser liefert dann anhand der angeschlossenen Peripheriekomponenten die jeweilige Information über folgende Schnittstellen:

Schnittstellen Smart-shelf	
IR- Sensoren	GPIO-Eingang
LED- Ampelmodul	GPIO-Ausgang
I2C Display	I2C- Bus
Schnittstellen Smart-room	
IR- Sensor	GPIO-Eingang
Temp.- & Feuchtigkeitssensor	1-Wire Bus

Tabelle 71: Schnittstellen Smart-shelf & Smart-room

Zusätzlich ist innerhalb der Node-RED Instanz ein Webserver implementiert, welcher über alle aktuellen Zustände des Systems informiert. Das Webinterface ist mittels HTTP erreichbar. Weitere Information zu den ESPs und zu der Projektentwicklung sind in GitLab zum Projekt unter <https://gitlab.ce.jku.at/CE/digital-twin-with-node-red/-/tree/main/docs> einzusehen.

6. Installationsanleitung

In den folgenden Unterkapiteln wird die Installation der notwendigen Software für dieses Projekt für Microsoft Windows als auch für Linux erläutert.

6.1 Installationsanleitung für Windows

Um das vom externen Partner konfigurierte Setup „Digital Twin with Node-RED“ bei der Erstellung des Prozesses des gegebenen Use-Cases zu verwenden, muss das Repository, welches dem Projektteam unter <https://gitlab.ce.jku.at/CE/digital-twin-with-node-red> zur Verfügung gestellt wurde, geklont und erweitert werden. Diese Installationsanleitung folgt der README-Datei innerhalb des Repository des externen Partners.

6.1.1 Installation Git

Um das Repository gemäß der README-Datei zu klonen, muss Git installiert sein. Dies kann unter folgender URL erfolgen: <https://gitforwindows.org>. Das Projekt nutzt Git in der Version 2.38.1.

Falls nicht gewiss ist, ob Git schon installiert ist, kann man in der Eingabeaufforderung unter folgendem Kommando feststellen, welche Version von Git installiert ist.

C:\....\...> git --version

```
C:\Users\Berkan>git --version
git version 2.36.1.windows.1
```

Abbildung 12: Git-Versionsüberprüfung

In Abbildung 12 sieht man eine Ausführung des Kommandos und dessen Ausgabe zur Git-Version.

6.1.2 Installation Visual Studio Code

Da als Entwicklungsumgebung des „Digital Twin with Node-RED“-Projekts Visual Studio Code verwendet wurde, wird Visual Studio Code für die Umsetzung des Projekts empfohlen. Die Software kann unter der folgenden URL heruntergeladen werden:

<https://code.visualstudio.com/download>

Zu Beginn des Projekts wurde die Version 1.74.3 verwendet. Durch Aktualisierungen der Software wurde beim Abschluss des Projekts die Version 1.75.0 verwendet.

6.1.3 Installation Python

Zur Umsetzung und Durchführung des Prozesses muss Python installiert werden. Die Installation kann unter folgender URL erfolgen: <https://www.python.org/downloads/>
 Unter der URL <https://docs.python.org/3/using/windows.html#the-full-installer> ist eine Anleitung zur Installation von Python zu finden. Die Version 3.6.8 wurde verwendet und wird empfohlen.

6.1.4 Repository klonen

Vor dem Klonen des Repositorys sollte am besten ein neuer Ordner, in welches das Repository geklont werden soll, erstellt werden.



Abbildung 13: Erstellung eines Ordners

Wie in Abbildung 13 zu sehen ist, wurde ein Ordner für das Klonen des Repository erstellt.

```
C:\Users\Berkan>cd PJ_IT-Projekt
C:\Users\Berkan\PJ_IT-Projekt>
```

Abbildung 14: Navigieren zu erstelltem Ordner

Danach muss in der Eingabeaufforderung zu diesem erstellten Ordner (Abbildung 19) navigiert werden. Dies kann unter dem Kommando „cd <Ordnername>“ bzw. „cd <Pfad>“ erfolgen. Nun kann man das Repository des externen Partners mit folgendem Kommando im Ordner „PJ_IT-Projekt“ klonen.

Kommando: git clone <https://gitlab.ce.jku.at/CE/digital-twin-with-node-red.git>

« Benutzer > Berkan > PJ_IT-Projekt		
Name	Änderungsdatum	Typ
digital-twin-with-node-red	21.01.2023 01:02	Dateiordner

Abbildung 15: geklontes Repository

In Abbildung 15 wird ersichtlich, dass nach dem Klonen im „PJ_IT-Projekt“-Ordner ein Ordner „digital-twin-with-node-red“-Ordner, welches das geklonte Repository ist, hinzugefügt wurde.

6.1.5 Installation Node Version Manager

Um ein Node-RED Projekt zu kreieren bzw. starten müssen davor Voraussetzungen getroffen werden. Eine Voraussetzung ist die Installation von Node-Version-Manager unter folgender URL: <https://github.com/coreybutler/nvm-windows/releases>. Version 1.1.10 wurde verwendet und ist empfohlen.

Assets		10
 nvm-noinstall.zip	4.31 MB	Nov 1, 2022
 nvm-noinstall.zip.checksum.txt	34 Bytes	Nov 1, 2022
 nvm-setup.exe	5.22 MB	Nov 12, 2022
 nvm-setup.zip	4.72 MB	Nov 1, 2022
 nvm-setup.zip.checksum.txt	34 Bytes	Nov 1, 2022
 nvm-update.exe	7.08 MB	Nov 12, 2022
 nvm-update.zip	4.08 MB	Nov 1, 2022
 nvm-update.zip.checksum.txt	34 Bytes	Nov 1, 2022
 Source code (zip)		Nov 1, 2022
 Source code (tar.gz)		Nov 1, 2022

Abbildung 16: NVM-Installationsschritt 1

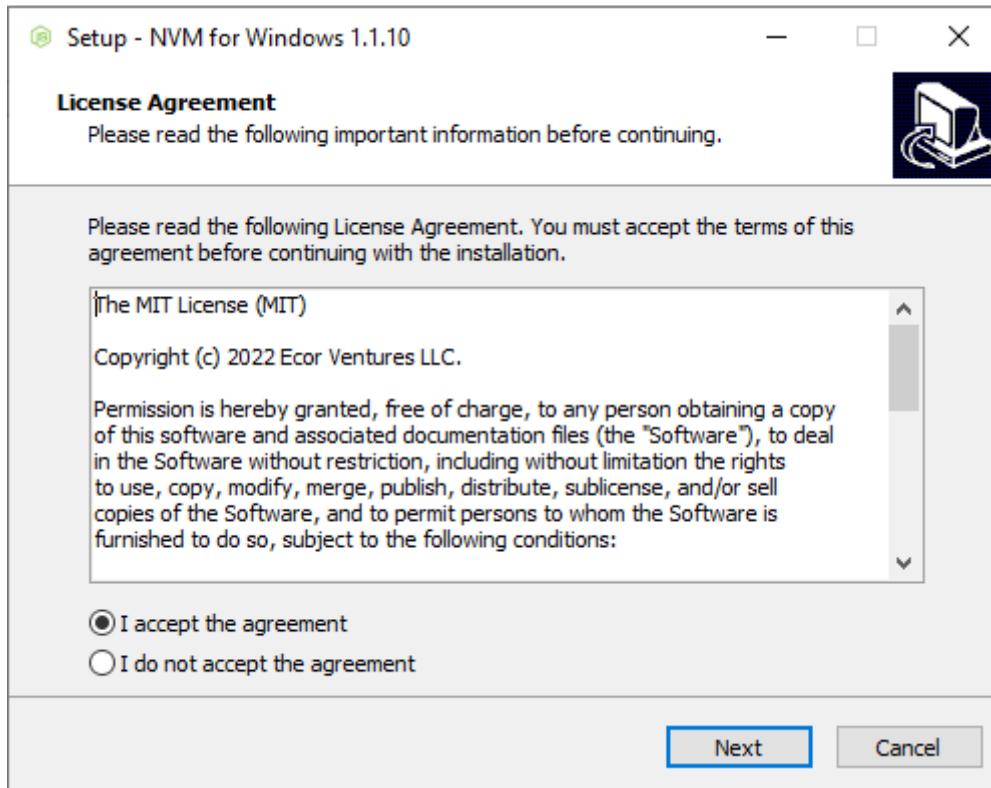


Abbildung 17: NVM-Installationsschritt 2

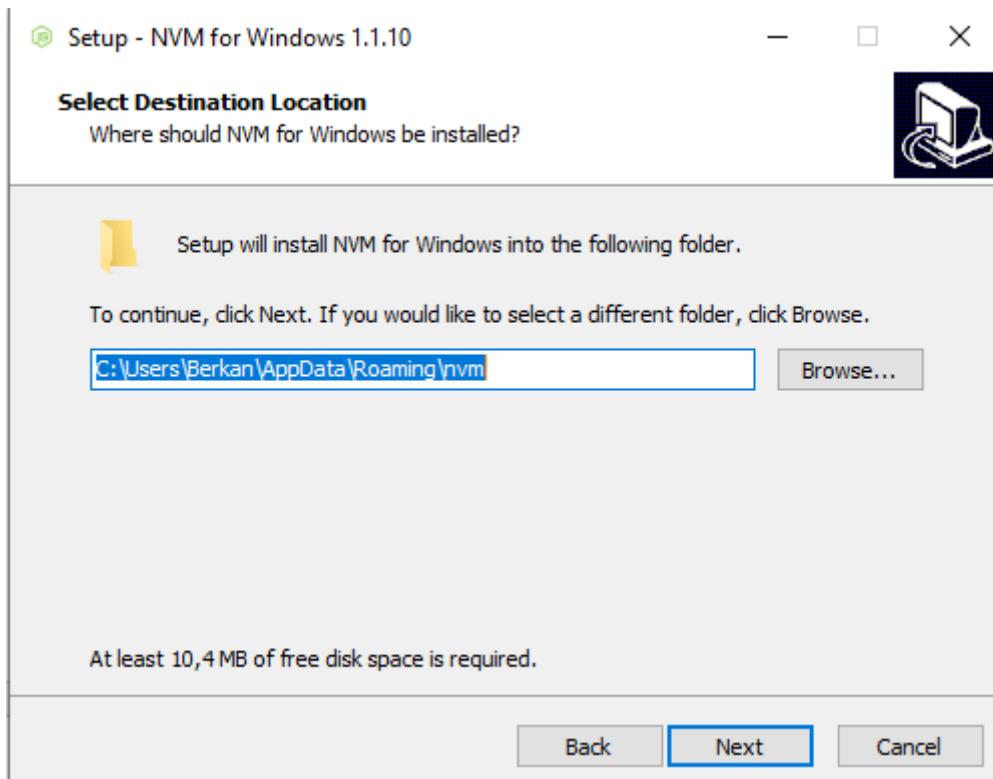


Abbildung 18: NVM-Installationsschritt 3

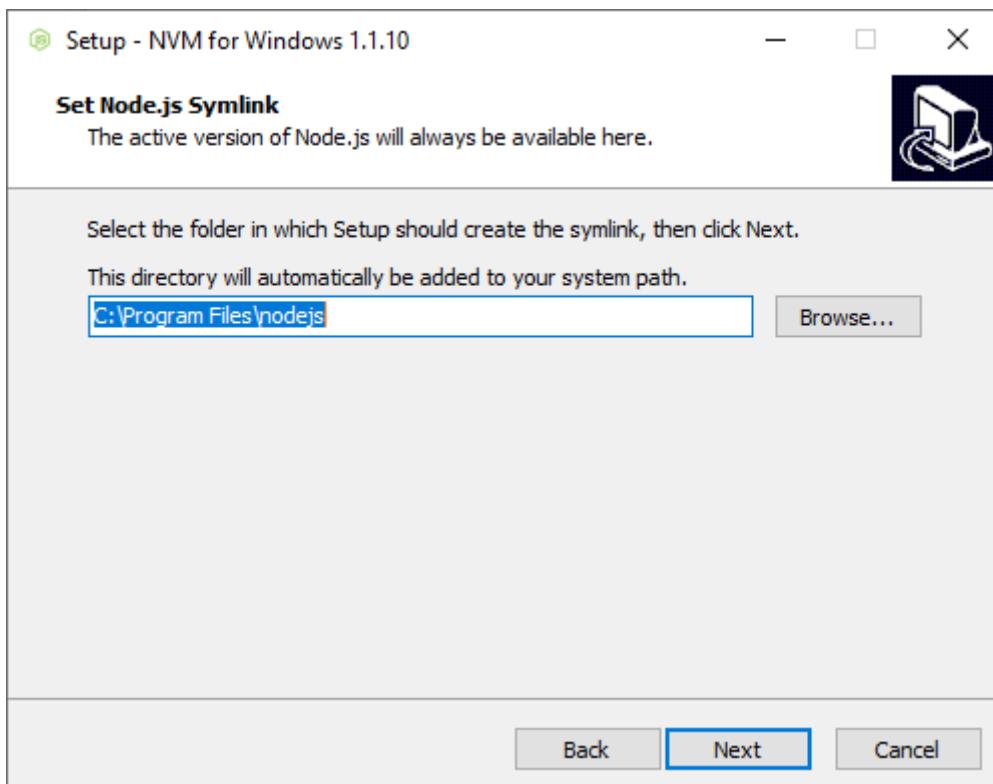


Abbildung 19: NVM-Installationsschritt 4

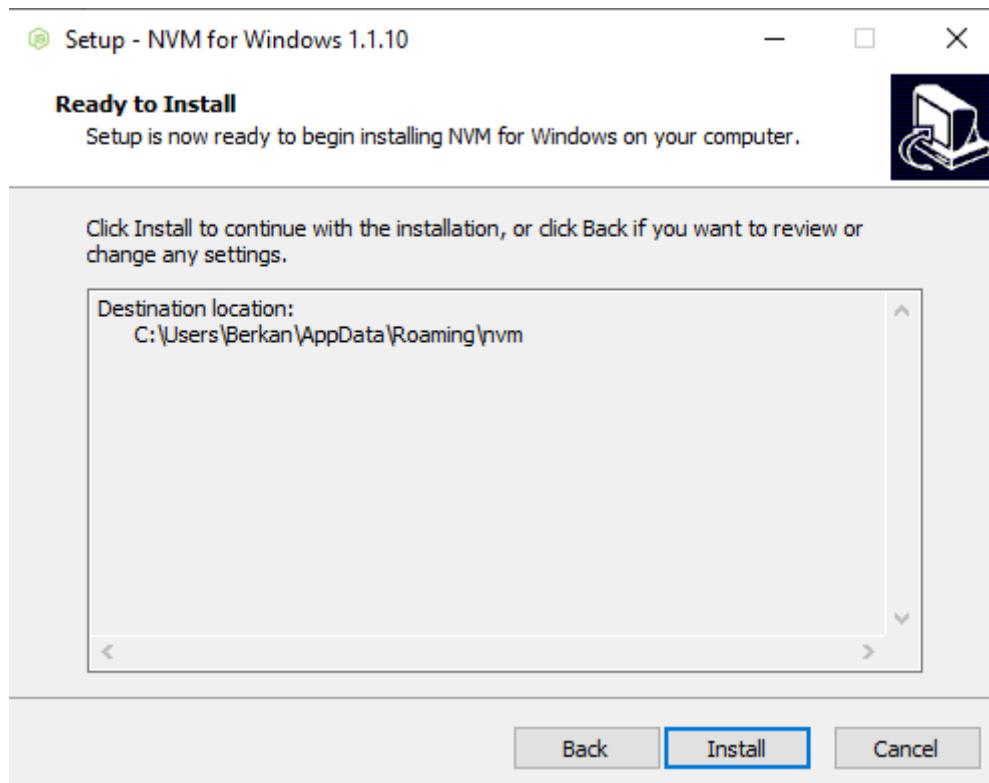


Abbildung 20: NVM-Installationsschritt 5

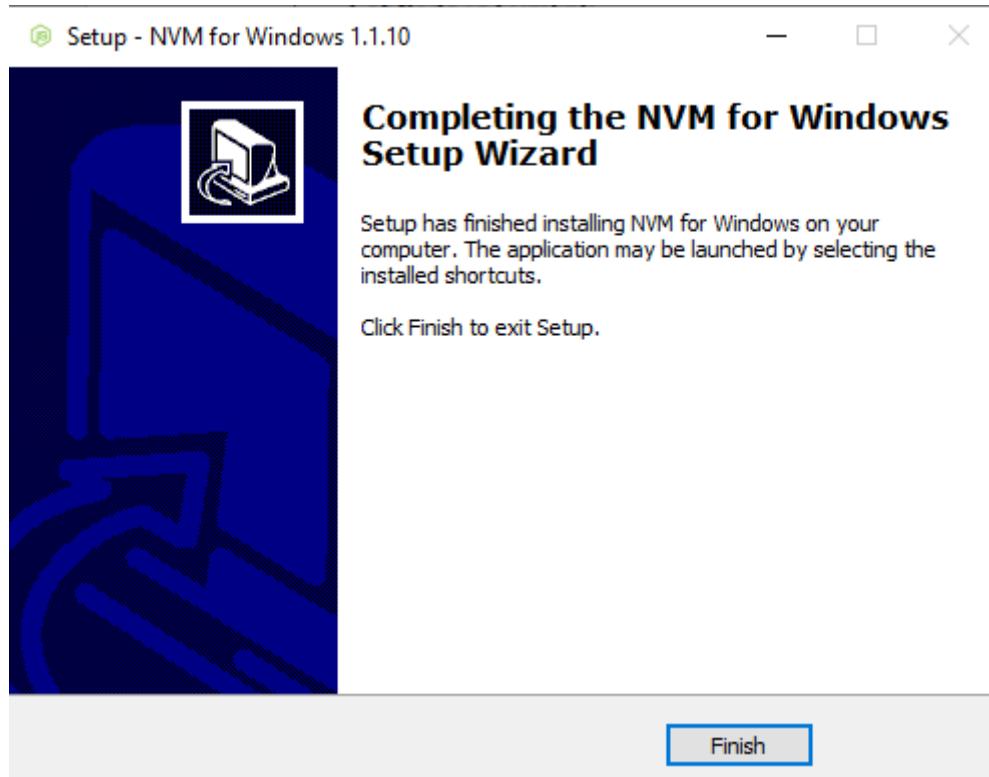


Abbildung 21: NVM-Installationsschritt 6

In Abbildung 16 bis 21 sind die einzelnen Installationsschritte von Node-Version-Manager ersichtlich.

```
C:\Users\Berkan>nvm --version  
1.1.10
```

Abbildung 22: Versionsüberprüfung

Unter dem Kommando „nvm --version“ (Abbildung 22) kann man überprüfen, ob die Installation erfolgreich war und welche Version, 1.1.10 in unserem Fall, installiert wurde.

6.1.6 Installation Node.js

Zunächst muss „Node.js“ (Abbildung 23) in der Version 18 mit folgendem Kommando installiert werden.

Kommando: „nvm install 18“

```
C:\Users\Berkan>nvm install 18  
Downloading node.js version 18.14.0 (64-bit)...  
Extracting node and npm...  
Complete  
npm v9.3.1 installed successfully.  
  
Installation complete. If you want to use this version, type  
nvm use 18.14.0  
C:\Users\Berkan>
```

Abbildung 23: Installationskommando Node.js

```
C:\Users\Berkan>node --version  
v18.13.0
```

Abbildung 24: Versionsüberprüfung Node.js

Die Installation und Version kann unter dem Kommando „node --version“ (Abbildung 24) überprüft werden.

6.1.7 Installation PlatformIO CLI

Die Installation von „PlatformIO CLI“ ist zur Kommunikation mit den jeweiligen Mikrocontrollern notwendig. Das Projekt nutzt und empfiehlt Version 6.1.5. Als erster Schritt muss das Skript der folgenden URL in eine Textdatei gespeichert werden:

<https://raw.githubusercontent.com/platformio/platformio-core-installer/master/get-platformio.py>

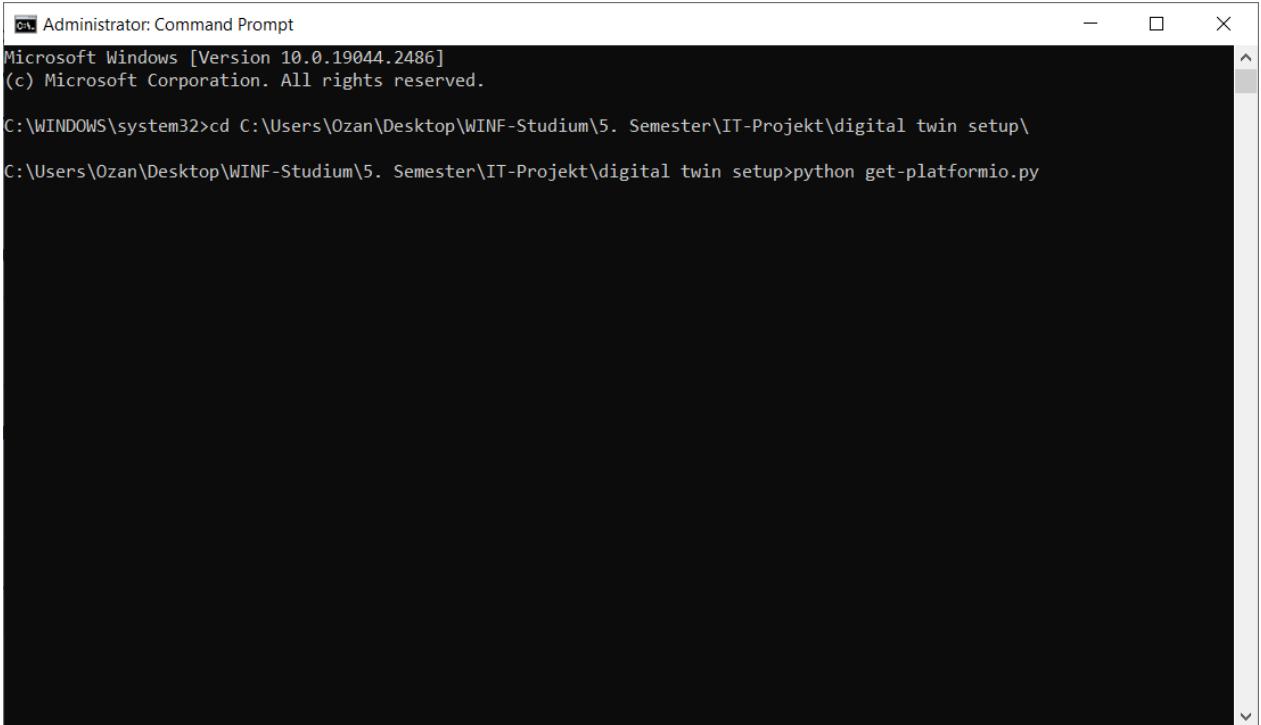
Die Textdatei muss anschließend wie folgend in ein Skript umbenannt werden (Achtung! Hier ist darauf zu achten, dass die Dateinamenserweiterungen aktiviert sind und angezeigt werden!):
get-platformio.py

Im nächsten Schritt muss eine Kommandozeile in dem Ordner, wo sich die get-platformio.py-Datei befindet, mit Administratorrechten geöffnet werden. Folgendes Kommando kann in einer Kommandozeile zum Wechseln in einen Ordner verwendet werden:

cd *ORDNERPFAD*

Darauffolgend muss das Skript mit dem folgenden Befehl in der Kommandozeile ausgeführt werden (siehe nächste Abbildung):

python get-platformio.py



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". The window title bar includes standard icons for minimize, maximize, and close. The command line shows the following text:
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19044.2486]
(c) Microsoft Corporation. All rights reserved.
C:\WINDOWS\system32>cd C:\Users\Ozan\Desktop\WINF-Studium\5. Semester\IT-Projekt\digital twin setup\
C:\Users\Ozan\Desktop\WINF-Studium\5. Semester\IT-Projekt\digital twin setup>python get-platformio.py

Abbildung 25: Ausführung von PlatformIO CLI Skript

Als nächster Schritt muss für die Nutzung der Kommandos von PlatformIO die entsprechende Systemvariable gesetzt werden. Unter den Umgebungsvariablen von der Windows-Installation ist folgender Eintrag bei der Systemvariable PATH einzufügen (UserName ist mit dem Benutzernamen zu ersetzen):

C:\Users\USERNAME\.platformio\penv\Scripts

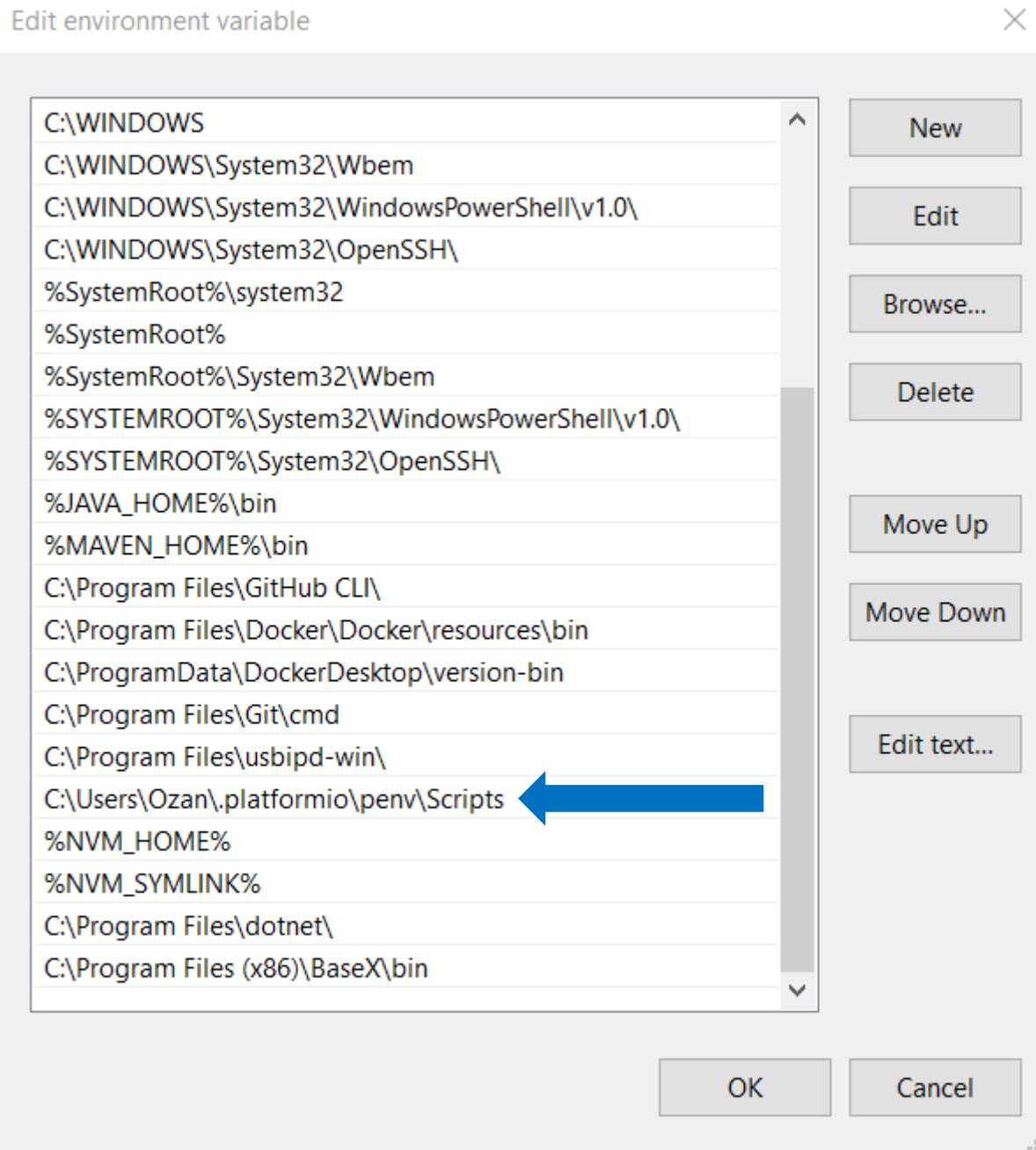


Abbildung 26: Setzen der Systemvariablen für PlatfotmIO CLI

6.1.8 Konfiguration und Flashen eines Mikrocontrollers

Die Konfiguration und das Flashen eines Mikrocontrollers ist bei Änderungen in der Netzwerkumgebung (WLAN-SSID oder WLAN-Passwort) und/oder der IP-Adresse/Port des MQTT-Servers notwendig.

Hinweis: Die folgenden Ordner- und Skriptangaben sind in Relation zu der Ordnerstruktur, welche sich in dem Git-Repository zum Zeitpunkt der Verfassung dieser Dokumentation wiederfinden lässt.

Zuerst muss der Port des mittels USB angeschlossenen Mikrocontrollers mit folgendem Kommando bestimmt werden (siehe nächste Abbildung mit COM3):

```
pio device list
```

```
C:\Users\Ozan\Desktop\WINF-Studium\5. Semester\IT-Projekt\digital-twin\digital-twin-with-node-red>pio device list
COM3
-----
Hardware ID: USB VID:PID=10C4:EA60 SER=0001 LOCATION=1-2
Description: Silicon Labs CP210x USB to UART Bridge (COM3)
```

Abbildung 27: Bestimmung des Ports eines Mikrocontrollers

In der Datei platformio.ini in dem Ordner arduino/dynamic-board ist der entsprechende Port (siehe nächste Abbildung) bei upload_port und monitor_port beim jeweiligen Mikrocontroller zu setzen:

```
[env:az-delivery-devkit-v4]
platform = espressif32
board = az-delivery-devkit-v4
upload_port = COM3
monitor_port = COM3

[env:esp12e]
platform = espressif8266
board = esp12e
upload_port = COM4
monitor_port = COM4
```

Abbildung 28: Setzen der Ports für PlatformIO

Der erste Eintrag ist für den Mikrocontroller ESP-32 V4 von az-delivery. Der zweite Eintrag ist für den Mikrocontroller ESP8266.

Als vorletzter Schritt für die Konfiguration der Mikrocontroller ist das Setzen der Netzwerkdaten (WLAN-SSID und WLAN-Passwort) und der IP-Adresse/Port des MQTT-Servers beim jeweiligen Mikrocontroller notwendig. Für einen ESP32 sind diese Daten zum Beispiel in der Datei config.json in dem Ordner arduino/dynamic-board/esp32-001 zu setzen. Für einen ESP8266 sind diese Daten zum Beispiel in der Datei config.json in dem Ordner arduino/dynamic-board/esp8266-001 zu setzen. Diese Ordner repräsentieren dabei jeweils einen spezifischen Mikrocontroller. Es ist daher zum Beispiel möglich bei zwei verschiedenen ESP32 zwei verschiedene Konfigurationen zu setzen (entsprechend jeweils unter dem Ordner esp32-001 oder esp32-002).

```
{
  "device": {
    "name": "ESP32-DEV-001",
    "pins": {
    }
  },
  "wifi": [
    "ssid": "LAPTOP-HT83L1DR 2375",
    "password": ":h914L17"
  ],
  "mqtt": {
    "server": {
      "host": "192.168.137.1",
      "port": 1883
    },
    "client": {
      "id": "ESP32-DEV-001"
    },
    "base-topic": "dynamic-board/ESP32-DEV-001/"
  }
}
```

Abbildung 29: Konfigurieren des Mikrocontrollers vor dem Flashen

Als finaler Schritt ist es notwendig die veränderten Konfigurationen zu „flashen“. Flashen bezeichnet das Schreiben des Speichers des Mikrocontrollers mit der auszuführenden Software. Das Flashen wird mittels folgenden Kommandos zum Aufrufen des jeweilig notwendigen Scripts durchgeführt:

Windows: upload-config.cmd *MIKROCONTROLLERNAME* (zum Beispiel esp32-001)

Linux: ./upload-config.sh *MIKROCONTROLLERNAME* (zum Beispiel esp32-001)

Achtung: Es kann sein, dass beim Flashen des Mikrocontroller ESP32 beim Schritt „Connecting.....“ das Drücken des Knopfs „Boot“ am Mikrocontroller notwendig ist. Das ist notwendig, falls der Mikrocontroller sich nicht selbstständig in den richtigen Modus zum Beschreiben der Konfiguration setzt.

6.1.9 Installation Node-RED

```
C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red>
```

Abbildung 30: Navigieren zum geklonten Repository

Im nächsten Schritt muss zum geklonten Repository navigiert werden (Abbildung 30) um innerhalb des geklonten Repositorys Node-RED zu klonen. Innerhalb dieses Ordners muss nun das Node-RED Repository mit dem Kommando „git clone <https://github.com/node-red/node-red.git>“ geklont werden. Für dieses Projekt wurde Node-RED in der Version 3.0.2 verwendet und empfohlen.

```
C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red>cd node-red  
C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red\node-red>
```

Abbildung 31: Navigieren zum geklonten Node-RED-Ordner

Danach muss zum geklonten Node-RED-Ordner (Abbildung 31) navigiert werden, um anschließend die folgenden Kommandos (Abbildung 32 und 33) innerhalb dieses Ordners auszuführen.

Kommandos: „npm install“ und „npx grunt build“.

```
C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red\node-red>npm install  
up to date, audited 865 packages in 7s  
72 packages are looking for funding  
  run `npm fund` for details  
10 vulnerabilities (5 moderate, 5 high)  
To address issues that do not require attention, run:  
  npm audit fix  
To address all issues (including breaking changes), run:  
  npm audit fix --force  
Run `npm audit` for details.
```

Abbildung 32: Ausführung von "npm install"

```
C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red>npx grunt build
Running "clean:build" (clean) task
>> 5 paths cleaned.

Running "jsonlint:messages" (jsonlint) task
>> 35 files lint free.

Running "jsonlint:keymaps" (jsonlint) task
>> 1 file lint free.

Running "concat:build" (concat) task

Running "concat:vendor" (concat) task

Running "copy:build" (copy) task
Created 17 directories, copied 223 files

Running "uglify:build" (uglify) task
>> 4 files created 2.41 MB → 897 kB

Running "sass:build" (sass) task

Running "attachCopyright:js" (attachCopyright) task
Attached copyright to packages/node_modules/@node-red/editor-client/public/red/red.min.js
Attached copyright to packages/node_modules/@node-red/editor-client/public/red/main.min.js

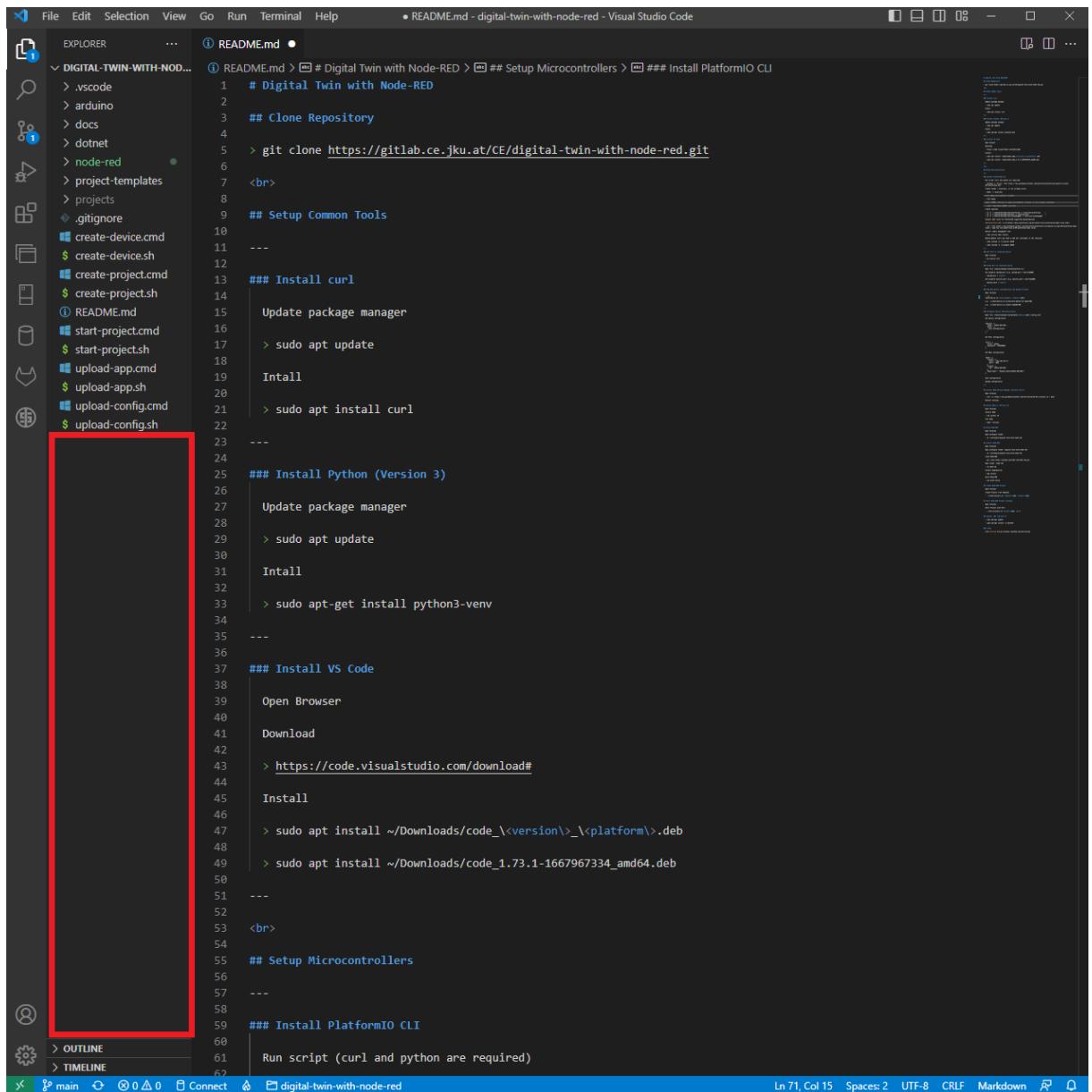
Running "attachCopyright:css" (attachCopyright) task
Attached copyright to packages/node_modules/@node-red/editor-client/public/red/style.min.css

Done.
```

Abbildung 33: Ausführung von "npx grunt build"

Damit werden Abhängigkeiten installiert und Node-RED „gebaut“ (build-Funktion). Da nun Visual-Studio-Code sowie auch das Repository des externen Partners geklont und mit Node-RED erweitert wurde, kann der Ordner innerhalb Visual Studio Codes geöffnet werden, um das Kreieren und Starten eines Node-RED-Projekts zu erleichtern.

Grundsätzlich können die folgenden Schritte bis zum Öffnen des Node-REDs auch in der Eingabeaufforderung von Windows durchgeführt werden.



```

File Edit Selection View Go Run Terminal Help README.md - digital-twin-with-node-red - Visual Studio Code

EXPLORER ... README.md ●
DIGITAL-TWIN-WITH-NODE-RED ...
  .vscode
  > arduino
  > docs
  > dotnet
  > node-red ●
  > project-templates
  > projects
    .gitignore
    create-device.cmd
    $ create-device.sh
    create-project.cmd
    $ create-project.sh
    README.md
    start-project.cmd
    $ start-project.sh
    upload-app.cmd
    $ upload-app.sh
    upload-config.cmd
    $ upload-config.sh

OUTLINE TIMELINE

```

```

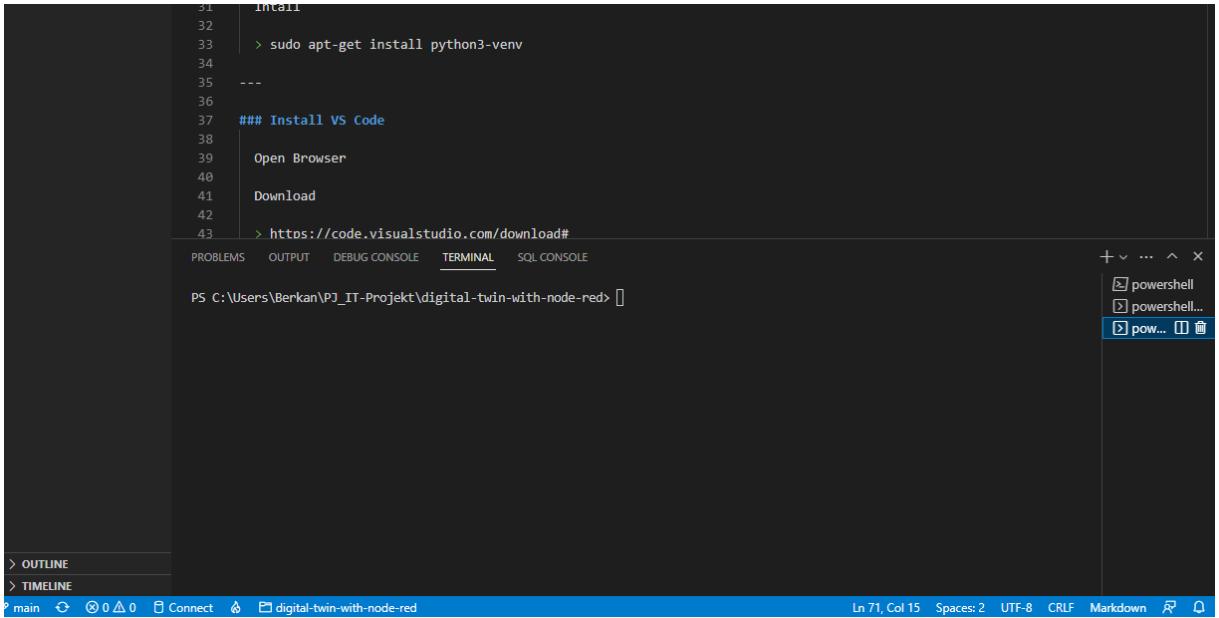
1 # Digital Twin with Node-RED
2
3 ## Clone Repository
4
5 > git clone https://gitlab.ce.jku.at/CE/digital-twin-with-node-red.git
6
7 <br>
8
9 ## Setup Common Tools
10 ---
11
12
13 ### Install curl
14
15 Update package manager
16
17 > sudo apt update
18
19 Intall
20
21 > sudo apt install curl
22
23 ---
24
25 ### Install Python (Version 3)
26
27 Update package manager
28
29 > sudo apt update
30
31 Intall
32
33 > sudo apt-get install python3-venv
34
35 ---
36
37 ### Install VS Code
38
39 Open Browser
40
41 Download
42 > https://code.visualstudio.com/download#
43
44 Install
45
46 > sudo apt install ~/Downloads/code_<version>_<platform>.deb
47
48 > sudo apt install ~/Downloads/code_1.73.1-1667967334_amd64.deb
49
50
51 ---
52
53 <br>
54
55 ## Setup Microcontrollers
56
57 ---
58
59 ### Install PlatformIO CLI
60
61 Run script (curl and python are required)

```

Ln 71, Col 15 Spaces: 2 UTF-8 CRLF Markdown

Abbildung 34: Geklontes Repository innerhalb VSC

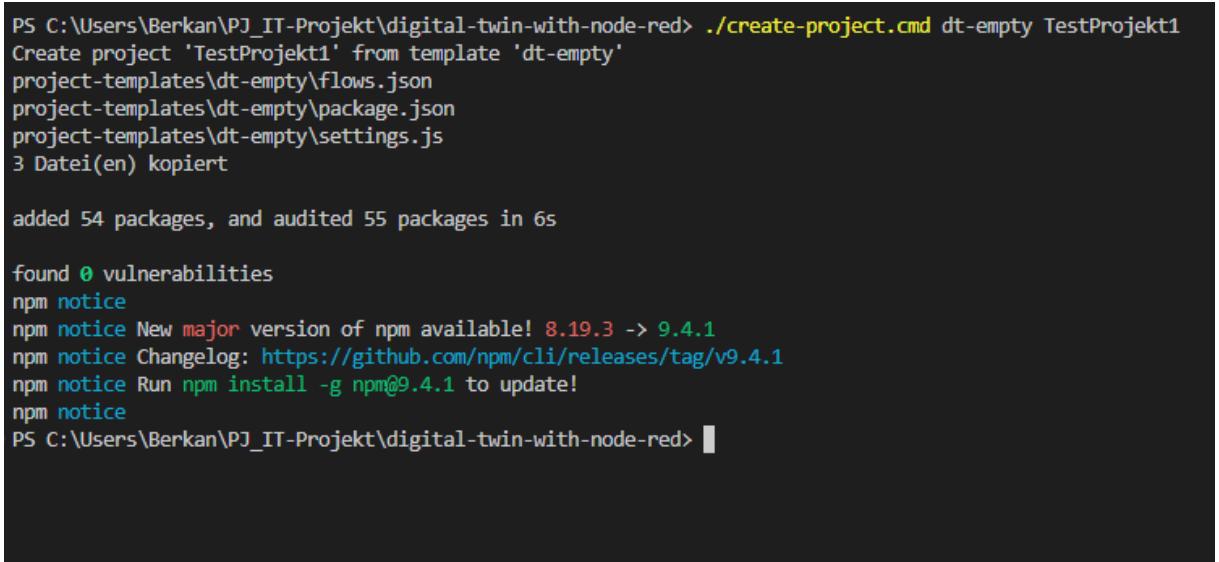
In Abbildung 34 sieht man den erstellten und konfigurierten Ordner in Visual-Studio-Code. Um ein Projekt zu starten, muss ein Terminal bzw. eine integrierte Eingabeaufforderung innerhalb des Visual-Studio-Codes geöffnet werden. Dies kann innerhalb der des rot markierten Bereichs (Abbildung 34) erfolgen. Durch ein Rechtsklick innerhalb des Bereichs kann man unter „Open in integrated Terminal“ beziehungsweise „Öffnen in integrierten Terminal“ ein Terminal bzw. eine Eingabeaufforderung innerhalb von Visual-Studio-Code öffnen.



The screenshot shows a Visual Studio Code interface with a dark theme. A terminal window is open in the center, displaying a script with numbered lines from 31 to 43. Lines 31-34 show the installation of Python 3-vENV. Lines 35-36 are empty. Lines 37-42 show steps to install VS Code: 'Open Browser', 'Download', and a URL 'https://code.visualstudio.com/download#'. Line 43 shows the command to run the script. Below the terminal are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and SQL CONSOLE. The status bar at the bottom shows the path 'PS C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red>', file statistics ('Ln 71, Col 15 Spaces: 2 UTF-8 CRLF'), and icons for Markdown, Find, and Replace. On the right side, there's a sidebar with a 'pow...' entry.

Abbildung 35: Terminal innerhalb VSC

Da nun eine integrierte Eingabeaufforderung (Abbildung 35) geöffnet wurde, muss nicht erneut zum gewünschten Ordner, wie anfangs zum Klonen des Repository und Node-REDs, navigiert werden. Im nächsten Schritt kann ein Node-RED-Projekt kreiert und gestartet werden.



The screenshot shows a terminal window with the following output:

```

PS C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red> ./create-project.cmd dt-empty TestProjekt1
Create project 'TestProjekt1' from template 'dt-empty'
project-templates\dt-empty\flows.json
project-templates\dt-empty\package.json
project-templates\dt-empty\settings.js
3 Datei(en) kopiert

added 54 packages, and audited 55 packages in 6s

found 0 vulnerabilities
npm notice
npm notice New major version of npm available! 8.19.3 -> 9.4.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.4.1
npm notice Run npm install -g npm@9.4.1 to update!
npm notice
PS C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red>
  
```

Abbildung 36: Erstellung eines Node-RED-Projekts

Unter „./create-project.cmd <template name> <project name>“ kann ein Node-Projekt (Abbildung 36) erstellt werden. Der Template-Name ist je nach Projekt-Anwendungsbereich unter den Templates, welche sich im Ordner „project-templates“ befinden und in der Abbildung 34 zu sehen ist, auszusuchen, wobei in unserem Fall bei der Erstellung des Projekts das Template „dt-empty“ verwendet wurde. Der Projektname ist frei wählbar und ist unter dem Ordner „projects“ (Abbildung 34) wiederzufinden.

```
PS C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red> ./start-project.cmd TestProjekt1 8080
Start project 'TestProjekt1' on port '8080'

6 Feb 01:36:00 - [info]
Welcome to Node-RED
=====
6 Feb 01:36:00 - [info] Node-RED version: v3.0.2
6 Feb 01:36:00 - [info] Node.js version: v18.13.0
6 Feb 01:36:00 - [info] Windows_NT 10.0.19045 x64 LE
6 Feb 01:36:00 - [info] Loading palette nodes
6 Feb 01:36:08 - [info] Dashboard version 3.2.3 started at /ui
6 Feb 01:36:08 - [info] Settings file : C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red\projects\TestProjekt1\settings.json
6 Feb 01:36:08 - [info] Context store : 'default' [module=memory]
6 Feb 01:36:08 - [info] User directory : C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red\projects\TestProjekt1
6 Feb 01:36:08 - [warn] Projects disabled : editorTheme.projects.enabled=false
6 Feb 01:36:08 - [info] Flows file : C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red\projects\TestProjekt1\flows.json
6 Feb 01:36:08 - [warn] Encrypted credentials not found
6 Feb 01:36:08 - [info] Server now running at http://127.0.0.1:8080/
6 Feb 01:36:08 - [info] Starting flows
6 Feb 01:36:08 - [info] Started flows
6 Feb 01:36:29 - [info] [mqtt-broker:29da46e4e24b94d5] Connection failed to broker: mqtt://192.168.155.9:1883
[]
```

Abbildung 37: Starten des erstellten Projekts

Unter „./start-project.cmd <projekt name> <port>“ (Abbildung 37) kann das erstellte Projekt nun unter dem angegeben Port gestartet werden, jedoch noch ohne MQTT-Verbindung (Abbildung 37).

6.1.10 Verbindung mit MQTT

Um die Verbindung mit MQTT herzustellen, muss „.NET 6.0“, welche unter <https://dotnet.microsoft.com/en-us/download/dotnet/6.0> verfügbar ist, heruntergeladen und installiert werden.

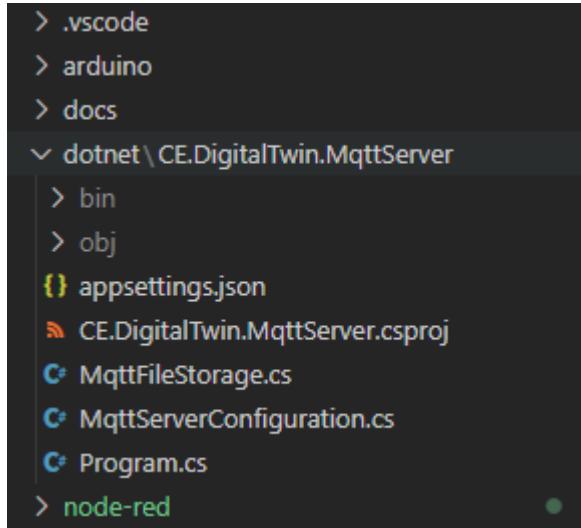


Abbildung 38: Terminal von MQTT-Server öffnen

Im Anschluss muss im „dotnet“-Ordner durch ein Rechtsklick auf „CE.DigitalTwin.MqttServer“ (Abbildung 38) ein integriertes Terminal (Abbildung 39) geöffnet werden.

```
PS C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red\dotnet\CE.DigitalTwin.MqttServer>
```

Abbildung 39: Terminal von MQTT-Server

```
PS C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red\dotnet\CE.DigitalTwin.MqttServer> dotnet run
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:1883
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Berkan\PJ_IT-Projekt\digital-twin-with-node-red\dotnet\CE.DigitalTwin.MqttServer\
```

Abbildung 40: Starten des MQTT-Servers

Der MQTT-Server, der den Schlüsselpunkt der Kommunikation des Projekts innehat, kann nun unter dem Kommando „dotnet run“ (Abbildung 40) gestartet werden.

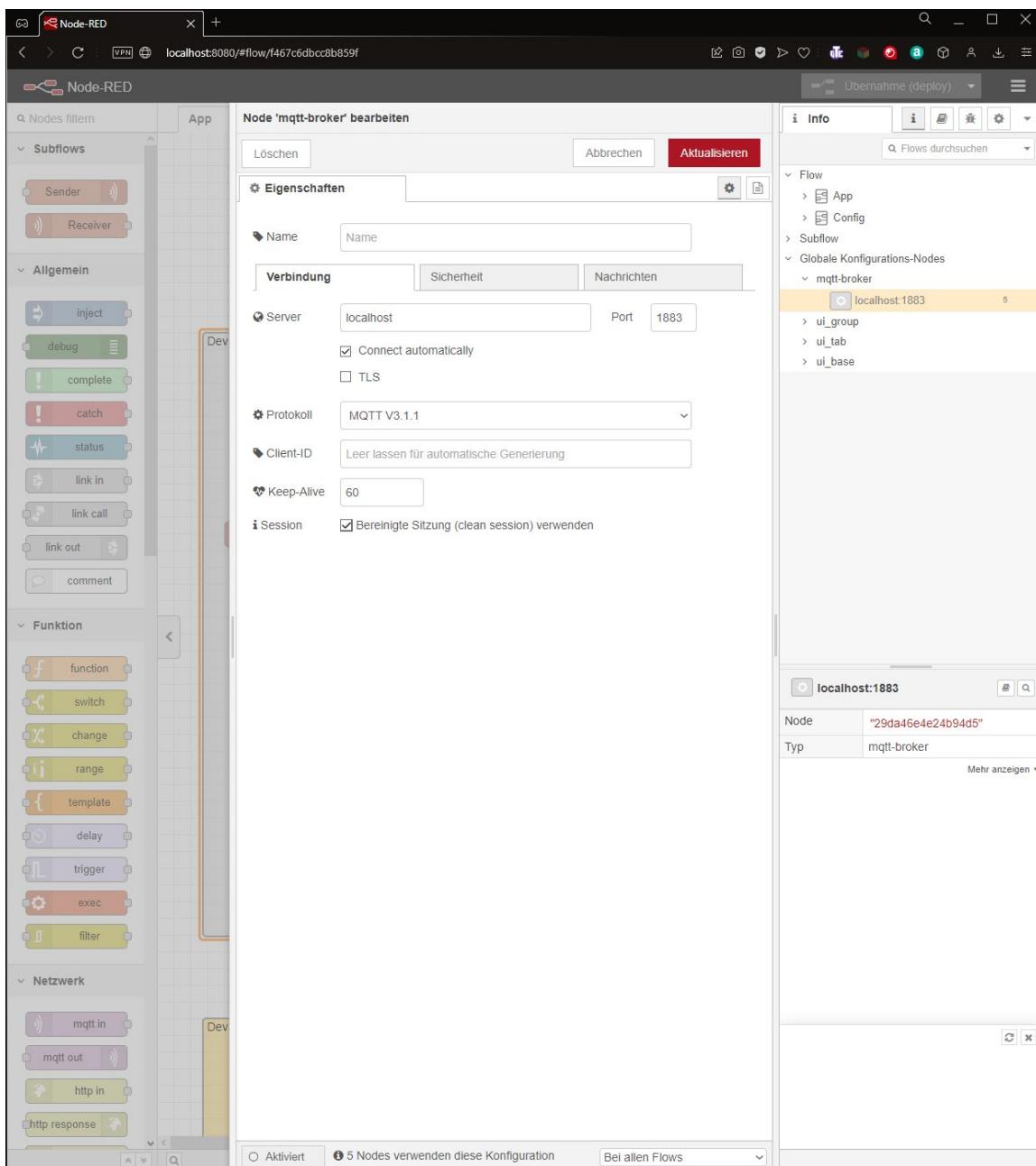


Abbildung 41: Ändern der Serveradresse innerhalb von Node-RED

Um die Verbindung herzustellen, muss innerhalb des Node-REDs, welche beim Starten des Projekts im Browser unter dem angegeben Port (Abbildung 37) läuft, die Adresse des MQTT-Servers geändert werden. Unter den Globalen Konfigurations-Nodes (Abbildung 41) muss durch Doppelklick auf „mqtt-broker“ das Bearbeitungsfenster dessen Nodes geöffnet werden. Im Anschluss muss unter „Verbindung“ der „Server“ auf die IP-Adresse des MQTT-Servers geändert werden. Da in unserem Fall der MQTT-Server zur Veranschaulichung auf dem gleichen Rechner läuft, genügt die Eingabe „localhost“. Nach der Eingabe muss die Bearbeitung aktualisiert und die Änderung übernommen (deploy) werden.

```
info: CE.DigitalTwin.MqttServer.MqttServer[0]
      Debugging client connected 'nodered_a078617d4b6d66ab' (endpoint: OHM087C821CE5)
```

Abbildung 42: Überprüfung der Verbindung zu MQTT 1.0

```
6 Feb 13:09:56 - [info] [mqtt-broker:29da46e4e24b94d5] Connected to broker: mqtt://localhost:1883
```

Abbildung 43: Überprüfung der Verbindung zu MQTT 2.0

Die Verbindung kann im Terminal des MQTT-Servers (Abbildung 39) unter der Ausgabe „Debugging client connected ‘nodered<...>‘ (endpoint:“ (Abbildung 42) überprüft und bestätigt werden. Zudem sollte im Terminal des Hauptordners (Abbildung 35) ebenfalls eine Ausgabe zur Bestätigung der Verbindung (Abbildung 43) zu finden sein. Während der Nutzung von Node-RED und des MQTT-Servers sollten die Eingabeaufforderungen nicht geschlossen werden.

Nun können in Node-RED erste Prozesse bzw. Flows erstellt, konfiguriert, durchgeführt und getestet werden. Falls der Entwicklungsprozess zu Ende ist, kann durch „STRG + C“ in den Eingabeaufforderungen von Node-RED und dem MQTT-Server gestoppt werden.

6.2 Installation für Linux

Die Installationsanleitung ist der Anleitung für Windows sehr ähnlich. Diese Anleitung folgt der README-Datei, die in dem Git-Repository des Projekts unter <https://gitlab.ce.jku.at/CE/digital-twin-with-node-red> dem Projektteam zur Verfügung gestellt wurde.

6.2.1 Installation Git

Wie auch in Windows, muss auch in Linux Git (Abbildung 45) installiert sein, um das Repository des externen Partners zu klonen. Das Projekt nutzt und empfiehlt Version 2.34.1. Dies kann in Linux unter folgendem Kommando erfolgen.

Kommando: sudo apt-get install git

```
berkanlinux@DESKTOP-RK3TA12:~$ git --version
git version 2.34.1
berkanlinux@DESKTOP-RK3TA12:~$
```

Abbildung 44: Versionsüberprüfung von Git

Die Git-Version (Abbildung 44) kann mittels „git --version“ ermittelt werden.

```
berkanlinux@DESKTOP-RK3TA12:~$ To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

berkanlinux@DESKTOP-RK3TA12:~$ sudo apt-get install git
[sudo] password for berkanlinux:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.34.1-1ubuntu1.5).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
berkanlinux@DESKTOP-RK3TA12:~$
```

Abbildung 45: Installationskommando für Git

„sudo“ ist ein Kommando, das den Befehl als Administrator ausführt und „apt“ ist eine Abkürzung für Advanced Package Tool.

6.2.2 Installation curl

```
berkanlinux@DESKTOP-RK3TA12:~$ sudo apt update
[sudo] password for berkanlinux:
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [601 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [127 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [8076 B]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [528 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [81.2 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [556 B]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [637 kB]
Get:13 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [87.7 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [11.3 kB]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [4960 B]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [996 B]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [240 B]
Get:18 http://archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:19 http://archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:20 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:21 http://archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:22 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:23 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [856 kB]
Get:24 http://archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [190 kB]
```

Abbildung 46: Ausführung von „sudo apt update“

„sudo apt update“ (Abbildung 46) ist ein Befehl in Linux, welches die Liste der verfügbaren Pakete auf dem System aktualisiert (Package Manager). Unter anderem werden auch

Abhängigkeiten und Konflikte zwischen Pakete überprüft. Der Befehl sollte regelmäßig verwendet werden, um sicherzustellen, dass das System auf dem neuesten Stand ist.

```
berkanlinux@DESKTOP-RK3TA12:~$ sudo apt install curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcurl4
The following packages will be upgraded:
  curl libcurl4
2 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
Need to get 483 kB of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 curl amd64 7.81.0-1ubuntu1.7 [193 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libcurl4 amd64 7.81.0-1ubuntu1.7 [289 kB]
Fetched 483 kB in 1s (901 kB/s)
(Reading database ... 24112 files and directories currently installed.)
Preparing to unpack .../curl_7.81.0-1ubuntu1.7_amd64.deb ...
Unpacking curl (7.81.0-1ubuntu1.7) over (7.81.0-1ubuntu1.6) ...
Preparing to unpack .../libcurl4_7.81.0-1ubuntu1.7_amd64.deb ...
Unpacking libcurl4:amd64 (7.81.0-1ubuntu1.7) over (7.81.0-1ubuntu1.6) ...
Setting up libcurl4:amd64 (7.81.0-1ubuntu1.7) ...
Setting up curl (7.81.0-1ubuntu1.7) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
berkanlinux@DESKTOP-RK3TA12:~$
```

Abbildung 47: Installationskommando von curl

Das Kommando (Abbildung 47) installiert das „curl“-Paket, welches Nutzern ermöglicht, Daten von einem Server herunterzuladen oder an einen Server hochzuladen. Das Projekt nutzt und empfiehlt Version 7.81.0.

6.2.3 Installation Visual Studio Code

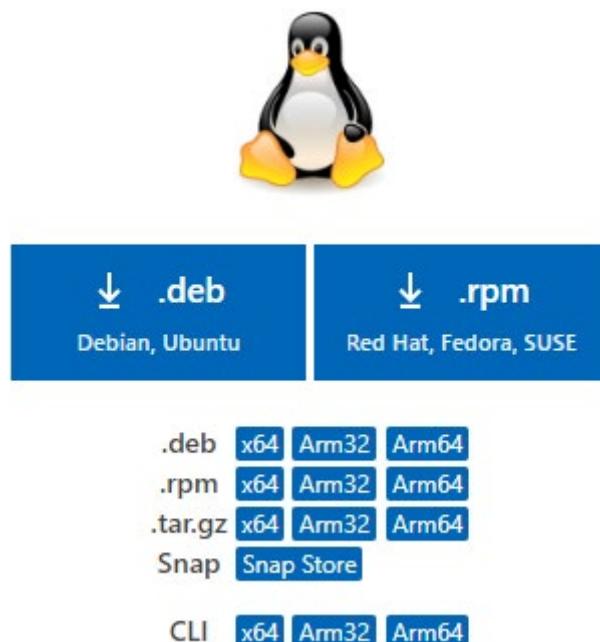


Abbildung 48: Datei zur Installation von VSC

Für die Installation von Visual Studio Code für Linux muss zuerst unter „<https://code.visualstudio.com/download#>“ die „.deb“-Datei (Abbildung 48) heruntergeladen werden. Im Anschluss muss das folgende Kommando ausgeführt werden.

Kommandos: „sudo apt install ~/Downloads/code \<version> \<platform>.deb“

→ Bsp: „sudo apt install ~/Downloads/code_1.75.0-1675266613_amd64.deb“

```
berkanlinux@DESKTOP-RK3TA12:~$ sudo apt install /mnt/c/Users/Berkan/Downloads/code_1.75.0-1675266613_amd64.deb
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'code' instead of '/mnt/c/Users/Berkan/Downloads/code_1.75.0-1675266613_amd64.deb'
The following additional packages will be installed:
adwaita-icon-theme als-a-topology-conf alsu-ucm-conf at-spi2-core cpp cpp-11 dconf-gsettings-backend dconf-service
fontconfig fontconfig-config fonts-dejavu-core gcc-11-base gsettings-desktop-schemas gtk-update-icon-cache
hicolor-icon-theme humanity-icon-theme libasound2 libasound2-data libatk-bridge2.0-0 libatk1.0-0 libatk1.0-data
libatspi2.0-0 libauthen-sasl-perl libavahi-client3 libavahi-common-data libavahi-common3 libcairo-gobject2 libcairo2
libclone-perl libcolor2 libcupsc2 libdata-dump-perl libdatr1 libdbconf1 libdeflate0 libdrm-amdgpu1 libdrm-intel1
libdrm-nouveau2 libdrm-radeon1 libencode-locale-perl libepoxy0 libfile-basedir-perl libfile-desktopentry-perl
libfile-listing-perl libfile-mimeinfo-perl libfont-afm-perl libfontconfig1 libfontfont1 libfreetype6 libgbm1
libgdk-pixbuf2.0-0 libgdk-pixbuf2.0-bin libgdk-pixbuf2.0-common libgl1 libgl1-amber-dri libgl1-mesa-dri
libglapi-mesa libglvnd0 libglx0 libgraphite2-3 libgtk-3-0 libgtk-3-bin libgtk-3-common libgtk3-0
libharfbuzz0b libhtml-form-perl libhtml-format-perl libhtml-parser-perl libhtml-tagset-perl libhtml-tree-perl
libhttp-cookies-perl libhttp-daemon-perl libhttp-date-perl libhttp-message-perl libhttp-negotiate-perl libice6
libio-html-perl libio-socket-ssl-perl libio-stringy-perl libipc-system-simple-perl libis123 libjbig0 libjpeg-turbo8
libjpeg8 liblcms2-2 libllvm11 libllvm15 liblwp-mediatypes-perl liblwp-protocol-https-perl libmailtools-perl libmpc3
libnet-dbus-perl libnet-htp-perl libnet-smtp-ssl-perl libnets-sleay-perl libnsp4 libnss3 libpango-1.0-0
libpangocairo-1.0-0 libpangoft2-1.0-0 libpciaccess0 libphobos2-ldc-shared98 libpixman-1-0 librsvg2-2 librsvg2-common
libsecret-1-0 libsecret-common libsensors-config libsensors5 libsthai libthai-data libthai0 libtie-ihash-perl
libtiff5 libtimageadperl libtry-tiny-perl liburi-perl libvte-2.91-0 libvte-2.91-common libvted-3-0 libvulkan1
libwayland-client0 libwayland-cursor0 libwayland-egl1 libwayland-server0 libwebp7 libwww-robotrules-perl
libx11-protocol-perl libx11-xcb1 libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0
libxcb-render0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libcomposite1 libxcursor1 libxdamage1
```

Abbildung 49: Ausführung des Installationskommandos von VSC

In Abbildung 49 ist die Ausführung des genannten Kommandos ersichtlich. Um die Installation von Visual Studio Code auszuführen, muss zum bis zum gewünschten Ordner, in der sich die heruntergeladene „.deb“-Datei befindet, navigiert werden. Das Navigieren zum gewünschten „Downloads“-Ordner musste mit einem „/mnt/“ begonnen werden, da man ansonsten nicht zum „C“- oder „D“-Datenträger gelangen konnte. Wofür und warum „/mnt/“ verwendet wird, kann unter <http://www.info.org/mnt.html> überprüft werden.

Am Anfang des Projekts wurde die Version 1.74.3, welche zur Umsetzung des Projekts problemlos funktionierte, verwendet. Im Laufe des Projekts wurde die Version von Visual Studio Code zur Sicherstellung, dass die Prozesse auch mit anderen Versionen funktionieren, aktualisiert. Zum Schluss wurde die Version 1.75.0 verwendet.

6.2.4 Installation Python

Die Installation von Python der Version 3 kann in Linux (Abbildung 50) mithilfe folgenden Kommandos erfolgen.

Kommando: sudo apt-get install python3-venv

Zur Überprüfung der Version können folgende Kommandos durchgeführt werden.

Kommandos: python --version

```
python3 --version
```

```
berkanlinux@DESKTOP-RK3TA12:~$ sudo apt-get install python3-venv
[sudo] password for berkanlinux:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3-venv is already the newest version (3.10.6-1~22.04).
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
berkanlinux@DESKTOP-RK3TA12:~$
```

Abbildung 50: Installationskommando für Python (Version 3)

Wie in Abbildung 50 ersichtlich ist, wurde für Linux die Version 3.10.6 installiert und verwendet.

6.2.5 Repository klonen

Da „Git“ bereits installiert ist kann das bereitgestellte Repository in den vorher erstellten und leeren Ordner „Projekt-IT“ (Abbildung 51) geklont werden. Auch hier muss zuerst zum gewünschten Ordner navigiert werden.

```
berkanlinux@DESKTOP-RK3TA12:/mnt/c/Users/Berkan/Projekt-IT$ git clone https://gitlab.ce.jku.at/CE/digital-twin-with-node-red.git
Cloning into 'digital-twin-with-node-red'...
Username for 'https://gitlab.ce.jku.at': k12018666
Password for 'https://k12018666@gitlab.ce.jku.at':
remote: Enumerating objects: 231, done.
remote: Total 231 (delta 0), reused 0 (delta 0), pack-reused 231
Receiving objects: 100% (231/231), 25.42 MiB | 3.65 MiB/s, done.
Resolving deltas: 100% (71/71), done.
berkanlinux@DESKTOP-RK3TA12:/mnt/c/Users/Berkan/Projekt-IT$
```

Abbildung 51: Klonen des Repositorys

6.2.6 Installation PlatformIO CLI

Das Projekt nutzt und empfiehlt Version 6.1.5. Als erster Schritt muss folgendes Kommando im Terminal ausgeführt werden:

```
python3 -c "$(curl -fsSL
https://raw.githubusercontent.com/platformio/platformio/master/scripts/get-platformio.py)"
```

Falls der Ordner /.local/bin nicht existiert, muss er mit folgendem Befehl erstellt werden:
`mkdir ~/.local/bin`

Als nächster Schritt muss \$PATH Variable angezeigt werden mit folgendem Kommando:
`echo $PATH`

Falls noch nicht vorhanden, muss die \$PATH Variable zu \$HOME/.local/bin exportiert werden:
`export PATH=$PATH:$HOME/.local/bin`

Als nächster Schritt müssen symbolische Verknüpfungen erstellt werden mit folgenden Kommandos:

`In -s ~/.platformio/pvenv/bin/platformio ~/.local/bin/platformio`

```
In -s ~/.platformio/penv/bin/pio ~/.local/bin/pio
In -s ~/.platformio/penv/bin/piodebuggdb ~/.local/bin/piodebuggdb
```

Daraufhin müssen die Regeln für die von den PlatformIO unterstützen Mikrocontroller installiert werden mit folgendem Kommando:

```
curl -fsSL https://raw.githubusercontent.com/platformio/platformio-core/master/scripts/99-
platformio-udev.rules | sudo tee /etc/udev/rules.d/99-platformio-udev.rules
```

Als nächstes ist es erforderlich den udev Service neuzustarten:

```
sudo service udev restart
```

Unter Ubuntu/Debian Installationen sind folgende Kommandos eventuell erforderlich:

```
sudo usermod -a -G dialout $USER
sudo usermod -a -G plugdev $USER
```

6.2.7 Konfiguration und Flashen eines Mikrocontrollers

Für die Konfiguration und das Flashen des Mikrocontrollers sind dieselben Schritte wie unter Punkt 6.1.8 durchzuführen.

6.2.8 Installation Node Version Manager

Im Gegensatz zur Installation von Node Version Manager für Windows, reicht die Ausführung (Abbildung 52) des folgenden Kommandos für die Installation von Node Version Manager der Version 0.39.2 aus.

Kommando: curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.2/install.sh | bash

```
berkanlinux@DESKTOP-RK3TA12:~$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.2/install.sh | bash
% Total    % Received % Xferd  Average Speed   Time     Time   Current
          Dload  Upload   Total   Spent    Left  Speed
100 15916  100 15916    0      0  67244      0 --:--:--:--:--:-- 67440
=> Downloading nvm from git to '/home/berkanlinux/.nvm'
=> Cloning into '/home/berkanlinux/.nvm'...
remote: Enumerating objects: 358, done.
remote: Counting objects: 100% (358/358), done.
remote: Compressing objects: 100% (304/304), done.
remote: Total 358 (delta 40), reused 164 (delta 28), pack-reused 0
Receiving objects: 100% (358/358), 219.04 KiB | 2.67 MiB/s, done.
Resolving deltas: 100% (40/40), done.
```

Abbildung 52: Installationskommando für Node Version Manager

6.2.9 Installation Node.js

```
berkanlinux@DESKTOP-RK3TA12:~$ nvm install 18
Downloading and installing node v18.14.0...
Downloaded https://nodejs.org/dist/v18.14.0/node-v18.14.0-linux-x64.tar.xz...
Computing checksum with sha256sum
Checksums matched!
Now using node v18.14.0 (npm v9.3.1)
Creating default alias: default -> 18 (-> v18.14.0)
berkanlinux@DESKTOP-RK3TA12:~$ node --version
v18.14.0
berkanlinux@DESKTOP-RK3TA12:~$
```

Abbildung 53: Installation Node.js

Die Installation sowie auch die Überprüfung der Version von Node.js (Abbildung 53) für Linux ist identisch mit der Anleitung für Windows (siehe Seite 75). Das Projekt nutzt und empfiehlt Version 18.

6.2.10 Installation Node-RED

Im nächsten Schritt muss in das geklonte Repository navigiert werden, um darin Node-RED zu klonen. Für dieses Projekt wurde Node-RED in der Version 3.0.2 verwendet und empfohlen. Dieser Schritt wird in Abbildung 54 veranschaulicht.

```
berkanlinux@DESKTOP-RK3TA12:/mnt/c/Users/Berkan/Projekt-IT$ cd digital-twin-with-node-red/
berkanlinux@DESKTOP-RK3TA12:/mnt/c/Users/Berkan/Projekt-IT/digital-twin-with-node-red$ git clone https://github.com/node-red/node-red.git
Cloning into 'node-red'...
remote: Enumerating objects: 80594, done.
remote: Counting objects: 100% (607/607), done.
remote: Compressing objects: 100% (344/344), done.
remote: Total 80594 (delta 286), reused 501 (delta 205), pack-reused 79987
Receiving objects: 100% (80594/80594), 39.51 MiB | 2.61 MiB/s, done.
Resolving deltas: 100% (53799/53799), done.
Updating files: 100% (1303/1303), done.
berkanlinux@DESKTOP-RK3TA12:/mnt/c/Users/Berkan/Projekt-IT/digital-twin-with-node-red$
```

Abbildung 54: zum geklonten Repository navigieren

Innerhalb des „node-red“-Ordners, welcher nun ein Unterordner des geklonten Repositorys ist, müssen Abhängigkeiten installiert werden. Durch Ausführen von „npm install“ (Abbildung 55) werden diese Abhängigkeiten heruntergeladen.

```
berkanlinux@DESKTOP-RK3TA12:/mnt/c/Users/Berkan/Projekt-IT/digital-twin-with-node-red/node-red$ npm install
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.

added 864 packages, and audited 865 packages in 2m
73 packages are looking for funding
  run `npm fund` for details
  9 vulnerabilities (5 moderate, 4 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New minor version of npm available! 9.3.1 -> 9.4.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.4.1
npm notice Run npm install -g npm@9.4.1 to update!
npm notice
berkanlinux@DESKTOP-RK3TA12:/mnt/c/Users/Berkan/Projekt-IT/digital-twin-with-node-red/node-red$
```

Abbildung 55: Ausführung von "npm install"

```

berkanlinux@DESKTOP-RK3TA12:/mnt/c/Users/Berkan/Projekt-IT/digital-twin-with-node-red$ npx grunt build
Running "clean:build" (clean) task
>> 0 paths cleaned.

Running "jsonlint:messages" (jsonlint) task
>> 35 files lint free.

Running "jsonlint:keymaps" (jsonlint) task
>> 1 file lint free.

Running "concat:build" (concat) task
Running "concat:vendor" (concat) task
Running "copy:build" (copy) task
Created 17 directories, copied 223 files

Running "uglify:build" (uglify) task
>> 4 files created 2.36 MB → 898 kB

Running "sass:build" (sass) task
Running "attachCopyright:js" (attachCopyright) task
Attached copyright to packages/node_modules/@node-red/editor-client/public/red/red.min.js
Attached copyright to packages/node_modules/@node-red/editor-client/public/red/main.min.js

Running "attachCopyright:css" (attachCopyright) task
Attached copyright to packages/node_modules/@node-red/editor-client/public/red/style.min.css

Done.
berkanlinux@DESKTOP-RK3TA12:/mnt/c/Users/Berkan/Projekt-IT/digital-twin-with-node-red$
```

Abbildung 56: Ausführung von "npx grunt build"

Unter „npx grunt build“ (Abbildung 56) versteht man das „Bauen“ (build-Funktion) von Node-RED und ist nur Nutzung von Node-RED eine Voraussetzung.

```

berkanlinux@DESKTOP-RK3TA12:/mnt/c/Users/Berkan/Projekt-IT/digital-twin-with-node-red$ ./create-project.sh dt-empty TestProjekt
  Create project 'TestProjekt' from template 'dt-empty'

added 54 packages, and audited 55 packages in 8s

found 0 vulnerabilities
berkanlinux@DESKTOP-RK3TA12:/mnt/c/Users/Berkan/Projekt-IT/digital-twin-with-node-red$
```

Abbildung 57: Erstellen eines Node-RED-Projekts

```

berkanlinux@DESKTOP-RK3TA12:/mnt/c/Users/Berkan/Projekt-IT/digital-twin-with-node-red$ ./start-project.sh TestProjekt 8080
Start project 'TestProjekt' on port '8080'

7 Feb 18:35:27 - [info]
Welcome to Node-RED
=====
7 Feb 18:35:27 - [info] Node-RED version: v3.0.2
7 Feb 18:35:27 - [info] Node.js version: v18.14.0
7 Feb 18:35:27 - [info] Linux 4.4.0-19041-Microsoft x64 LE
7 Feb 18:35:27 - [info] Loading palette nodes
7 Feb 18:35:28 - [info] Dashboard version 3.2.3 started at /ui
7 Feb 18:35:28 - [info] Settings file : /mnt/c/users/Berkan/Projekt-IT/digital-twin-with-node-red/projects/TestProjekt/settings.json
7 Feb 18:35:28 - [info] Context store : 'default' [module=memory]
7 Feb 18:35:28 - [info] User directory : /mnt/c/users/Berkan/Projekt-IT/digital-twin-with-node-red/projects/TestProjekt
7 Feb 18:35:28 - [warn] Projects disabled : editorTheme.projects.enabled=false
7 Feb 18:35:28 - [info] Flows file : /mnt/c/users/Berkan/Projekt-IT/digital-twin-with-node-red/projects/TestProjekt/flows.json
7 Feb 18:35:28 - [warn] Encrypted credentials not found
7 Feb 18:35:28 - [info] Server now running at http://127.0.0.1:8080/
7 Feb 18:35:28 - [info] Starting flows
7 Feb 18:35:28 - [info] Started flows
7 Feb 18:35:50 - [info] [mqqt-broker:29da46e4e24b94d5] Connection failed to broker: mqtt://192.168.155.9:1883
```

Abbildung 58: Starten des erstellten Projekts

Abbildung 57 und 58 bilden das Kreieren und Starten eines Node-RED-Projekts ab und sind der Ausführung für Windows ähnlich.

Kommandos: „./create-project.sh <template name> <project name>

„./start.project.sh <project name> <port>

Da die Parameter <template name>, <project name> und <port> bereits in der Installationsanleitung für Windows beschrieben wurden (siehe Seite 83/84), werden diese an dieser Stelle nicht erneut erklärt.

6.2.11 Verbindung mit MQTT

```
berkanlinux@DESKTOP-RK3TA12:~$ sudo apt-get install -y dotnet6
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  aspnetcore-runtime-6.0 aspnetcore-targeting-pack-6.0 dotnet-apphost-pack-6.0 dotnet-host dotnet-hostfxr-6.0
  dotnet-runtime-6.0 dotnet-sdk-6.0 dotnet-targeting-pack-6.0 dotnet-templates-6.0 liblttng-ust-common1
  liblttng-ust-ct15 liblttng-ust1 netstandard-targeting-pack-2.1
The following NEW packages will be installed:
  aspnetcore-runtime-6.0 aspnetcore-targeting-pack-6.0 dotnet-apphost-pack-6.0 dotnet-host dotnet-hostfxr-6.0
  dotnet-runtime-6.0 dotnet-sdk-6.0 dotnet-targeting-pack-6.0 dotnet-templates-6.0 dotnet6 liblttng-ust-common1
  liblttng-ust-ct15 liblttng-ust1 netstandard-targeting-pack-2.1
0 upgraded, 14 newly installed, 0 to remove and 39 not upgraded.
Need to get 124 MB of archives.
After this operation, 454 MB of additional disk space will be used.
```

Abbildung 59: Installieren von .NET6.0

Anders als für Windows, kann „.NET“ für Linux mit einem einfachen Kommando (Abbildung 59) in der Version 6.0 heruntergeladen.

```
berkanlinux@DESKTOP-RK3TA12:~$ cd /mnt/
berkanlinux@DESKTOP-RK3TA12:/mnt$ cd c
berkanlinux@DESKTOP-RK3TA12:/mnt/c$ cd users
berkanlinux@DESKTOP-RK3TA12:/mnt/c/users$ cd berkan
berkanlinux@DESKTOP-RK3TA12:/mnt/c/users/berkan$ cd Projekt-IT/
berkanlinux@DESKTOP-RK3TA12:/mnt/c/users/berkan/Projekt-IT$ cd digital-twin-with-node-red/
berkanlinux@DESKTOP-RK3TA12:/mnt/c/users/berkan/Projekt-IT/digital-twin-with-node-red$ cd dotnet/
berkanlinux@DESKTOP-RK3TA12:/mnt/c/users/berkan/Projekt-IT/digital-twin-with-node-red/dotnet$ cd CE.DigitalTwin.MqttServer
berkanlinux@DESKTOP-RK3TA12:/mnt/c/users/berkan/Projekt-IT/digital-twin-with-node-red/dotnet/CE.DigitalTwin.MqttServer$ dotnet run

Welcome to .NET 6.0!
-----
SDK Version: 6.0.113

-----
Installed an ASP.NET Core HTTPS development certificate.
To trust the certificate run 'dotnet dev-certs https --trust' (Windows and macOS only).
Learn about HTTPS: https://aka.ms/dotnet-https

-----
Write your first app: https://aka.ms/dotnet-hello-world
Find out what's new: https://aka.ms/dotnet-whats-new
Explore documentation: https://aka.ms/dotnet-docs
Report issues and find source on GitHub: https://github.com/dotnet/core
Use 'dotnet --help' to see available commands or visit: https://aka.ms/dotnet-cli
-----
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:1883
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /mnt/c/users/Berkan/Projekt-IT/digital-twin-with-node-red/dotnet/CE.DigitalTwin.MqttServer/
```

Abbildung 60: Starten des MQTT-Servers

Um den MQTT-Server zu starten, muss zu „CE.DigitalTwin.MqttServer“ (Abbildung 60) navigiert werden und anschließend das Kommando „dotnet run“ ausgeführt werden.

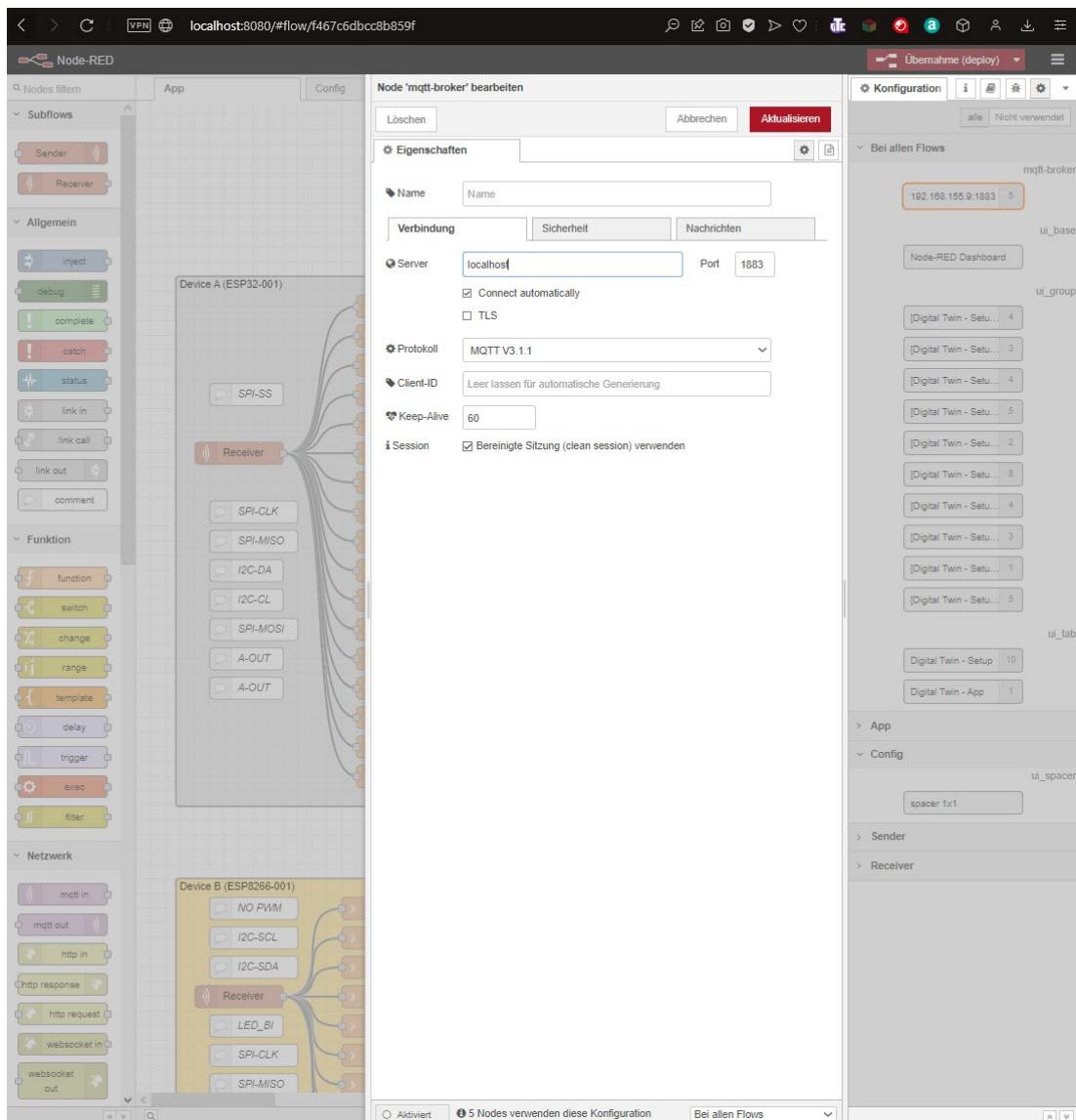


Abbildung 61: Ändern der Serveradresse innerhalb von Node-RED

Da nun sowohl ein Node-RED-Projekt erstellt und gestartet wurde und der MQTT-Server ebenfalls am Laufen ist, muss in Node-RED, welcher im Browser unter dem beim Starten des Projekts (Abbildung 58) angegebenen Port, „localhost:<port>“, läuft, der Server des „mqtt-brokers“ (Abbildung 61) auf die IP-Adresse des MQTT-Servers geändert werden. Da zur Veranschaulichung der Server auf gleichen Rechner läuft, wurde „localhost“ als Serveradresse angegeben. Nach der Eingabe der IP-Adresse des MQTT-Servers muss dies aktualisiert und die Änderung übernommen (deploy) werden.

Die Überprüfung der Verbindung von Node-RED mit dem MQTT-Server kann in den jeweiligen Terminals (Abbildung 42 und 43) erfolgen. Diese Terminals sollten nach dem Start eines Node-RED-Projekts und MQTT-Servers nicht geschlossen werden. Um das Projekt und den Server anzuhalten kann die Tastenkombination „STRG + C“ angewendet werden.

7. Betriebsanleitung



Abbildung 62: Gesamtbild des Node-RED-Prozesses

In diesem Kapitel wird der Ablauf der Node-RED Prozesse „Smart-shelf“ sowie „Smart-room“ (Abbildung 62) in ihren Einzelheiten näher beschrieben.

7.1 Betriebsanleitung für Smart-shelf



Abbildung 63: Node-RED Flows

„Connections“, „Config“ und „App“ (Abbildung 63) sind sogenannte Flows, die zur Entwicklung von „Smart-shelf“ und „Smart-room“ verwendet wurden. Der Flow „Config“ war bereits vom externen Partner konfiguriert gewesen, weshalb dieser Flow in diesem Projekt nicht näher betrachtet wird. Falls am Flow „App“ Änderungen vorgenommen werden, muss im Flow „Config“ nichts geändert werden.

7.1.1 MQTT-Anfrage

Zu Beginn des Prozesses erhalten wir vom Spot unter dem Topic „smart_shelf/request“ (Abbildung 65) eine MQTT-Anfrage, welches durch den „Receiver“-Knoten „MQTT-Request“ (Abbildung 64) gekennzeichnet ist.

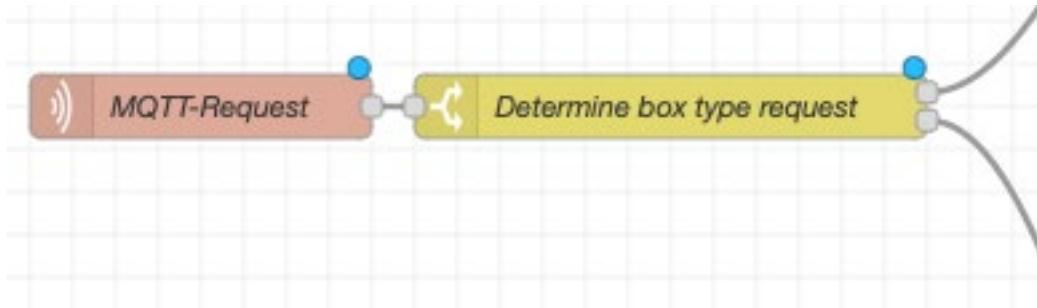


Abbildung 64: MQTT-Anfrageknoten (links), Knoten zum Ermitteln des Boxtyps (rechts)

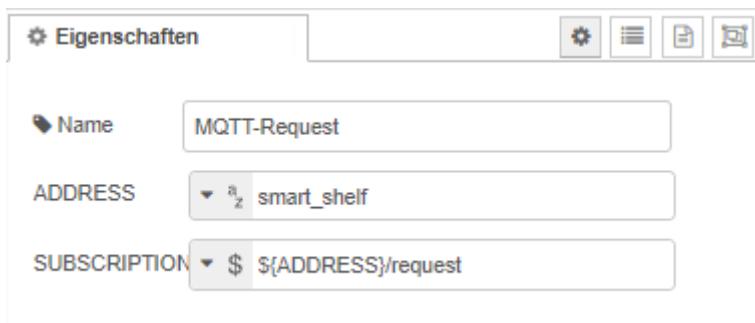


Abbildung 65: MQTT-Topic

In dieser Anfrage werden Nutzdaten mit den Parametern „processId“ und „status“ abgefangen. Unter „processId“ versteht man die Kennzeichnung (Identifier) der Anfrage des jeweiligen Camunda-Prozesses. „status“ bezeichnet den Typ der angefragten Box und sollte in diesem Fall nur „A“ oder „B“ sein, andere Boxtypen werden ignoriert und lösen keine Reaktion aus. Diese Struktur der Anfrage ist vom Projektteam der Gruppe B vorgegeben und einzuhalten. Wie in Abbildung 63 ersichtlich ist, besteht eine Verbindung zum Knoten „Determine box type request“, welcher Zugriff auf die Nutzdaten des Knotens „MQTT-Request“ hat.

7.1.2 Ermitteln des angefragten Boxtyps

Im „Switch“-Knoten „Determine box type request“ (Abbildung 64) wird je nach angefragtem Boxtyp, welcher in den Nutzdaten unter dem Parameter „status“ mitgegeben wurde, der Prozess umgeleitet. Durch die Eigenschaft „msg.payload.status“ (Abbildung 66) wird nur der Boxtyp ermittelt. Damit vergleichen wir den ermittelten Boxtyp mit den Werten „A“ oder „B“, die wir dem Knoten als Vergleichswerte zur Verfügung gestellt haben. Je nachdem, ob „status“ „A“ oder „B“ ist, wird der Prozess zum Knoten „Determine status“ (Abbildung 67) und der Gruppe, welche für den ermittelten Boxtyp zuständig ist, weitergeleitet.

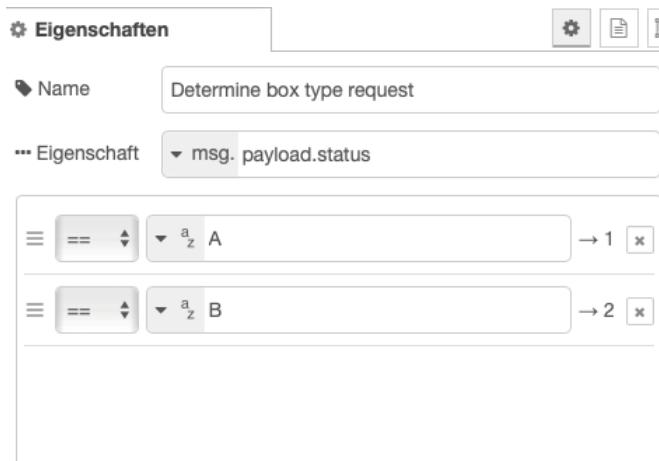


Abbildung 66: Eigenschaften des Switch-Knotens

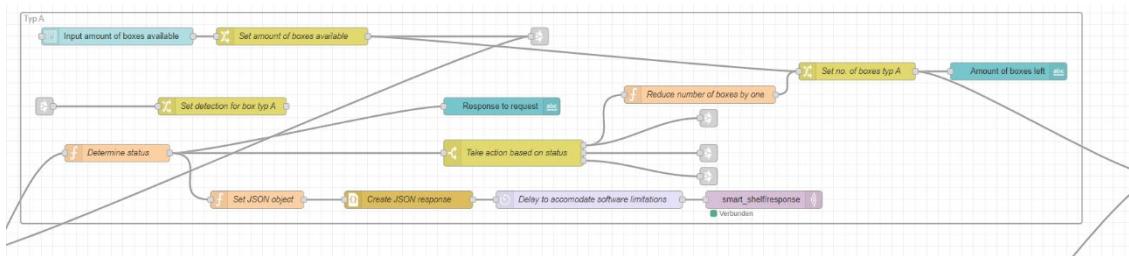


Abbildung 67: Prozessausschnitt für Boxtyp A

In den folgenden Kapiteln wird der Prozessablauf nur mit Boxtyp „A“ weitergeführt und beschrieben. Eine nähere Beschreibung des Prozessablaufs für Boxtyp „B“ ist nicht nötig, da der Ablauf des Prozesses für beide Boxtypen, bis auf die Variablennamen innerhalb der Knoten, identisch ist.

7.1.3 Zustand bestimmen

Um den Funktions-Knoten „Determine status“ (Abbildung 67) und vor allem ihren Inhalt zu verstehen, sind Informationen und Variablen anderer Knoten, die im folgendenden beschrieben werden, von großer Bedeutung.

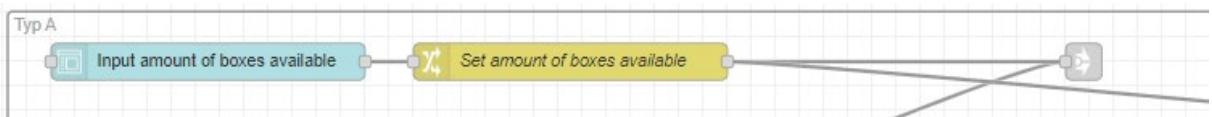


Abbildung 68: Form-Knoten (links), Change-Knoten (mitte), Link out-Knoten (rechts)

Der Form-Knoten „Input amount of boxes available“ (Abbildung 68) ist ein Knoten, der dem Node-RED Dashboard, User-Interface, ein Eingabeformular zur Bestimmung der Anzahl von Boxen des Typs „A“ hinzufügt.

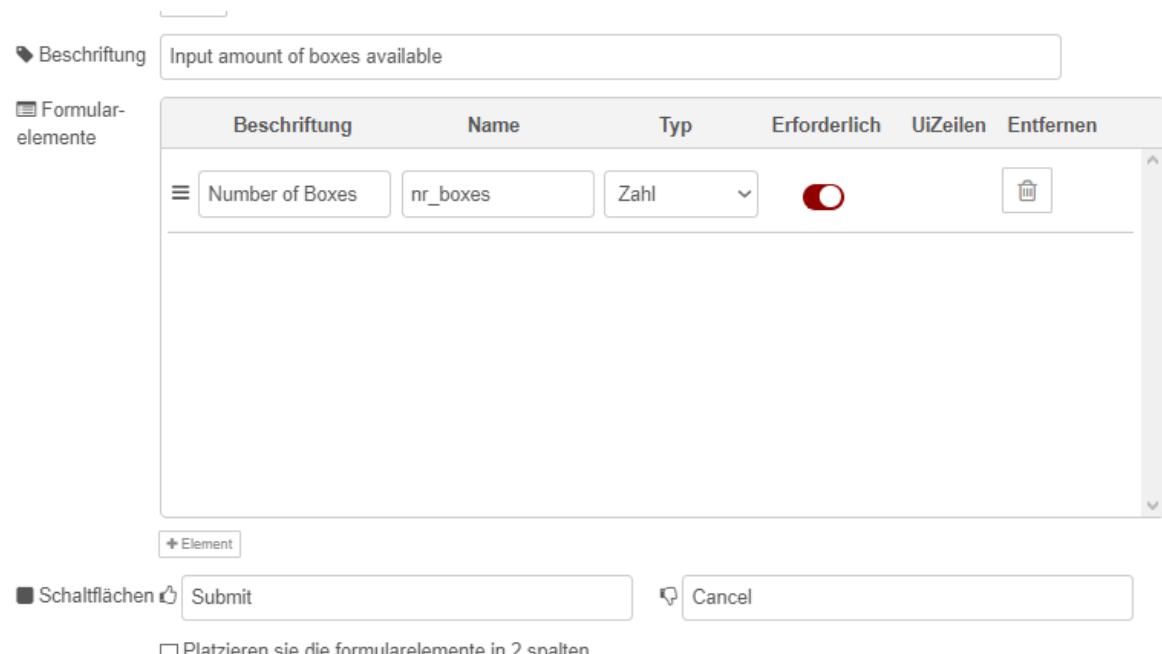


Abbildung 69: Eigenschaften des Knotens "Input amount of boxes available"

Die Anzahl der eingegebenen Boxen werden unter der Variable „nr_boxes“ und als Typ „Zahl“ (Abbildung 69) gespeichert.

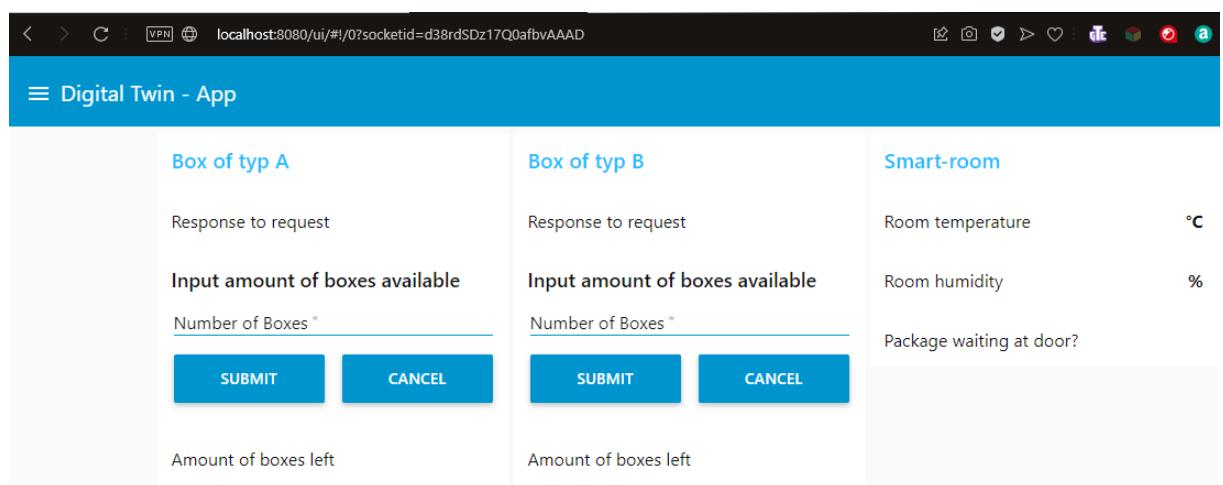


Abbildung 70: User-Interface

Das User-Interface (Abbildung 70) ist unter „localhost:<port>/ui“ zu finden. Wie in der Abbildung 70 zu sehen ist, befindet sich unter der Beschriftung „Input amount of boxes available“ eine Eingabe zur Bestimmung der „Number of Boxes“ für Boxtyp „A“. Durch „Submit“ wird die Anzahl der Boxen hinzugefügt. Diese werden im Feld „Amount of boxes left“ (Abbildung 70) angezeigt. Gleiche Semantik gilt auch für Boxtyp „B“.

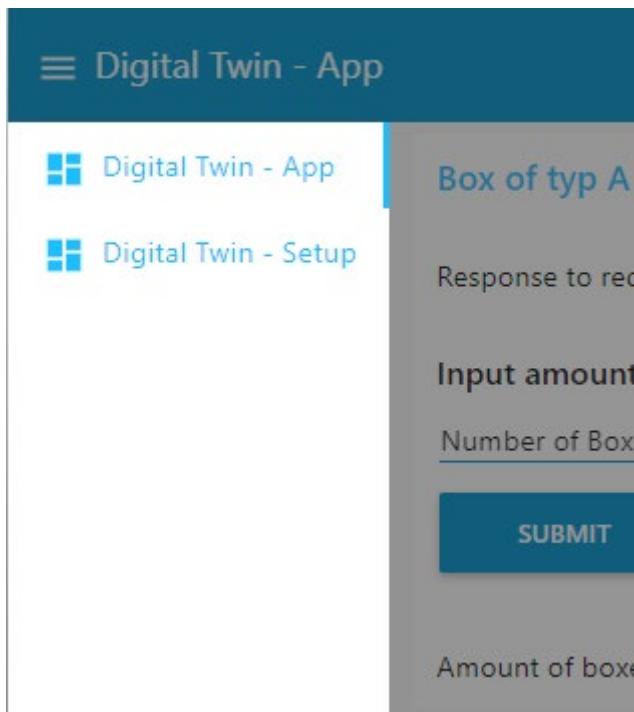


Abbildung 71: Seitenleiste am User-Interface

In der Seitenleiste (Abbildung 71) am User-Interface muss unter Setup das „Device“, Mikrocontroller, und der dazugehörige „Pin“ und „Mode“ (Abbildung 72) ausgewählt werden.

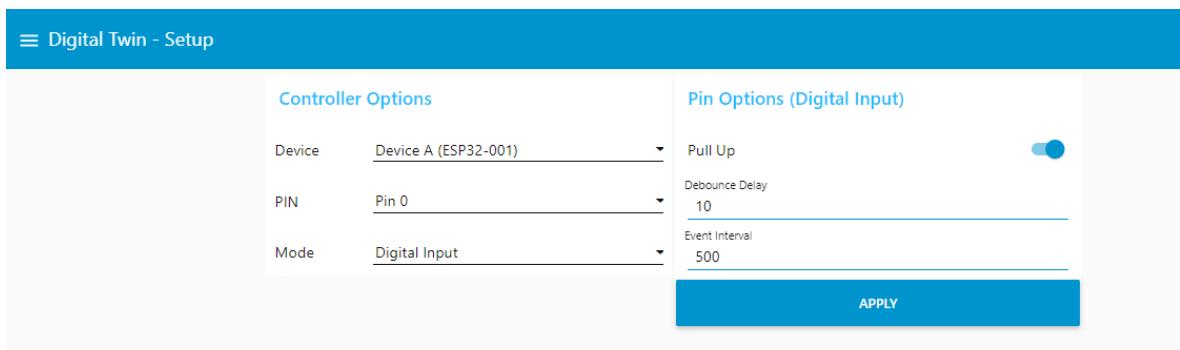


Abbildung 72: Auswahl des Geräts, Pins und Modus

Box of typ A

Response to request

Input amount of boxes available

Number of Boxes *

SUBMIT

CANCEL

Amount of boxes left

5

Abbildung 73: Ausschnitt des User-Interfaces

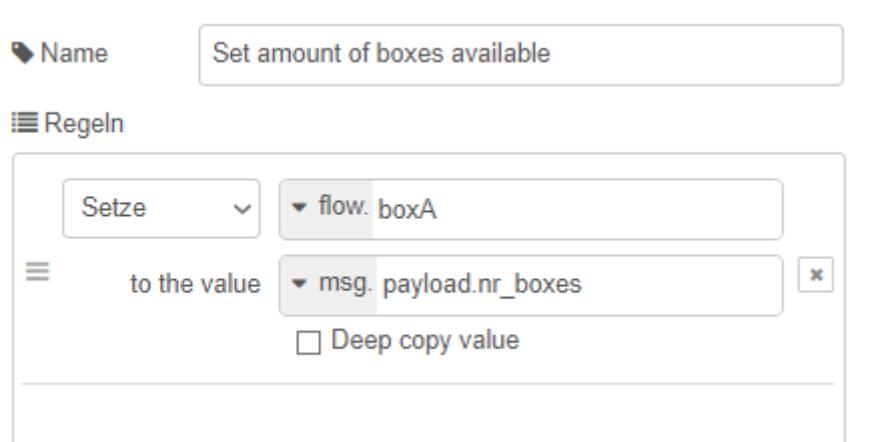


Abbildung 74: Eigenschaften des Knotens "Set amount of boxes available"

Im „change“-Knoten „Set amount of boxes available“ (Abbildung 68) wird unter „msg.payload.nr_boxes“ (Abbildung 74) auf die eingegebene Anzahl von Boxen zugegriffen und diese in der neuen Variable „boxA“ gespeichert. Das Präfix „flow.“ ermöglicht eine Nutzung der gesetzten Variable innerhalb des gesamten Flows. Der „Link out“-Knoten, welches sich am Ende der Abbildung 68 befindet, ist mit einem „Link in“-Knoten zur Steuerung der Lichter des LED-Ampelmoduls verbunden.



Abbildung 75: Verbindung zwischen zwei Knoten zur Erstellung und Setzung einer Variable

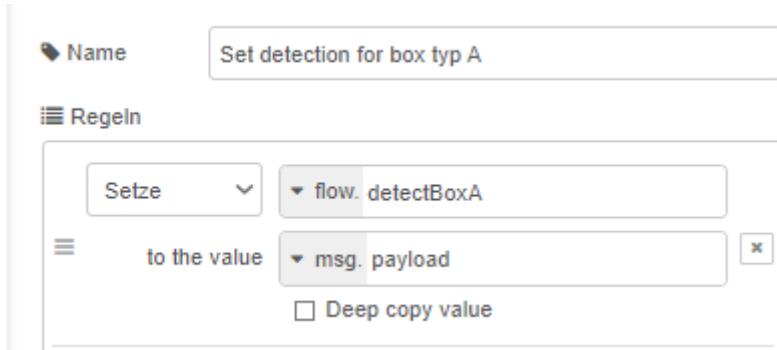


Abbildung 76: Eigenschaften des Knotens "Set detection for box typ A" (Erstellung und Setzung einer Variable)

In Abbildung 75 ist ein „Link in“-Knoten, der mit einem „change“-Knoten verbunden ist, ersichtlich. Der „Link in“-Knoten ist mit einem „Link out“-Knoten aus dem „Connections“-Flow verbunden, der die Nutzdaten vom Infrarotsensor erhält. Der „change“-Knoten „Set detection for box typ A“ (Abbildung 75) kann durch die Verbindung mit dem „Link in“-Knoten auf die Nutzdaten des Infrarotsensor zugreifen. Diese Nutzdaten, die nur aus 0 (Sensor hat ein Objekt nicht erkannt) und 1 (Sensor hat ein Objekt erkannt) bestehen und unter der Variable „payload“ gespeichert sind (Abbildung 76) werden der neuen Variable „detectBoxA“ zugewiesen.

Der Funktions-Knoten „Determine status“ (Abbildung 67) überprüft die vorhin genannten Variablen „boxA“ und „detectBoxA“, die nun im ganzen Flow zur Verfügung stehen. Je nach erfüllter Bedingung wird „msg.payload.status“ (Abbildung 77) geändert.

Die Funktion überprüft zunächst, ob der Wert der Variable „detectBoxA“ „1“ und „boxA“ größer als „0“ ist. Falls diese Bedingung erfüllt ist, wird das Attribut „msg.payload.status“ auf „Available“ gesetzt. Falls die erste Bedingung nicht erfüllt ist, überprüft die Funktion, ob „detectBoxA“ „0“ und „boxA“ größer als „0“ ist. Wenn diese Bedingung erfüllt ist, wird „msg.payload.status“ auf „Incomplete“ gesetzt. Falls keine der beiden Bedingungen erfüllt ist, wird „msg.payload.status“ auf „Empty“ gesetzt. Am Ende wird das modifizierte Nachrichtenobjekt „msg“ zurückgegeben.

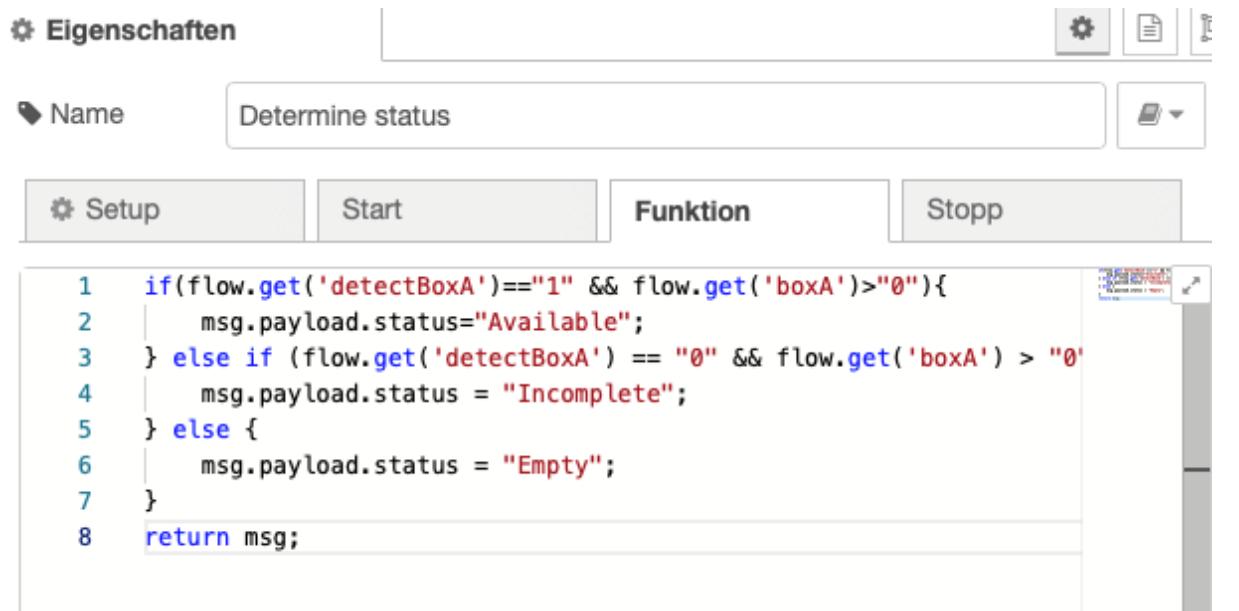


Abbildung 77: Eigenschaften des Knotens "Determine status"

7.1.4 Veranschaulichung des Zustands

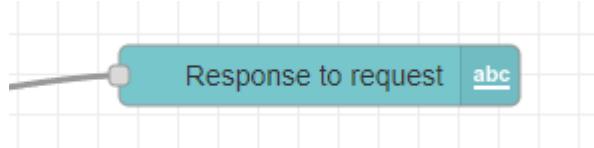


Abbildung 78: Knoten zur Veranschaulichung am User-Interface

Der in Abbildung 78 ersichtlicher Text-Knoten „Response to request“ ist ebenfalls ein Knoten, der zur Veranschaulichung von Texten bzw. Daten am User-Interface zuständig ist. In diesem Knoten wird die „msg.payload.status“-Variable (Abbildung 79), welche im „Determine status“-Knoten bestimmt wurde, abgespeichert, um diese im Anschluss an die Anfrage des Endnutzers als Antwort am User-Interface anzuzeigen.

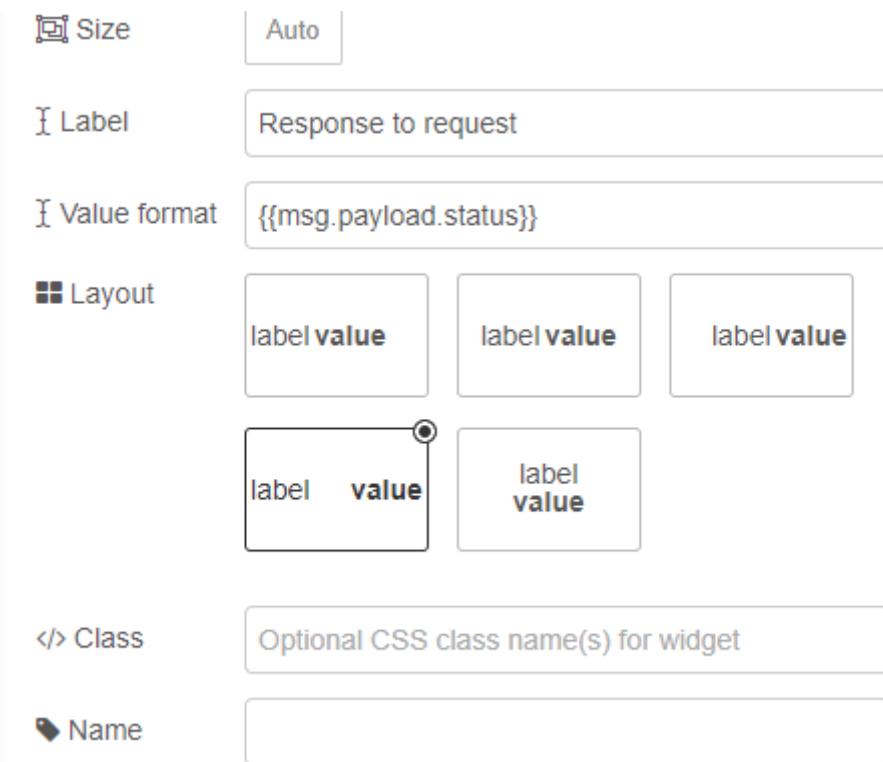


Abbildung 79: Eigenschaften des Knotens "Response to request"

Je nach „msg.payload.status“ kann auf die Anfrage ein „Empty“, „Incomplete“ oder „Available“ (Abbildung 80).

Response to request Empty

Abbildung 80: Veranschaulichung eines Zustands am User-Interface

„Empty“ bedeutet, dass sowohl keine Box vom Infrarotsensor erfasst wird und auch keine im Sortiment vorhanden sind. Bei „Incomplete“ wird keine Box erfasst, jedoch ist mindestens eine Box im Sortiment vorhanden. Unter „Available“ versteht man, dass sowohl eine Box erfasst wird und mindestens eine Box im Sortiment vorhanden ist.

7.1.5 Erweitern des JSON-Strings

Der Funktion-Knoten „Set JSON object“ (Abbildung 81) erweitert die bereits vorhandene Variable „payload“ (Abbildung 82), um einen neuen Parameter.

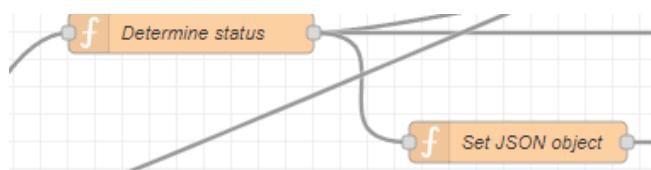


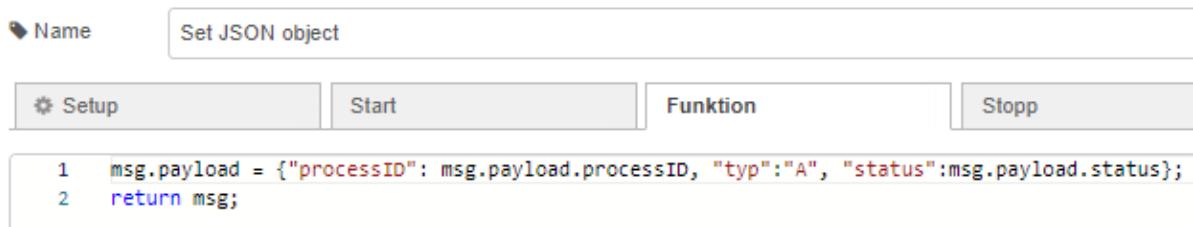
Abbildung 81: Funktionsknoten "Set JSON object" zur Erweiterung des vorhandenen JSON-Strings

Der Wert für den Parameter „processID“ ist bereits gegeben und wird dem „msg.payload“ durch „msg.payload.processID“ (Abbildung 82) entnommen.

Der Parameter „typ“ wurde neu hinzugefügt und auf „A“ festgelegt. Dieser neue Parameter wurde hinzugefügt, um beim Antworten der MQTT-Anfrage den angefragten Typ der Box darzustellen, da der Parameter „status“ nicht mehr „A“ ist.

Der Wert für den Parameter „status“ ist auch gegeben und wird ebenfalls dem „msg.payload“ durch „msg.payload.status“ entnommen.

Am Ende wird das modifizierte Nachrichtenobjekt „msg“ zurückgegeben.



```

Name: Set JSON object
Funktion:
1 msg.payload = {"processID": msg.payload.processID, "typ": "A", "status": msg.payload.status};
2 return msg;
  
```

Abbildung 82: Erweiterung des JSON-Strings um "typ"

Diesen Funktions-Knoten braucht man, weil Node-RED es nicht ermöglicht, das bestehende JSON-String, um den Typ der Box zu erweitern. Deswegen muss ein neuer JSON-String mit bestehenden Werten und dem neuen Parameter geschrieben werden.

7.1.6 Erstellen eines JSON-Objekts

„Create JSON response“ (Abbildung 83) ist ein Parser-Knoten, der Variablen bzw. Objekte in ein JSON-Objekt konvertieren. Bisher wurden die Daten des Objekts „msg.payload“ in einem JSON-String-Format gespeichert. In diesem Schritt wird „msg.payload“ (Abbildung 84) für die Antwort an die MQTT-Anfrage von einem JSON-String zu einem JSON-Objekt konvertiert.



Abbildung 83: Knoten zum Konvertieren des JSON-Strings zum JSON-Objekt



Optionen Objekt => JSON

JSON-Zeichenfolge formatieren

Abbildung 84: Eigenschaften des Knotens "Create JSON response"

7.1.7 Antwort auf MQTT-Anfrage

Der „mqtt out“-Knoten „smart_shelf/response“ (Abbildung 85) in Node-RED ist ein Ausgabe-Knoten, der verwendet wird, um Nachrichten an einen MQTT-Broker zu senden.



Abbildung 85: Knoten zum Beantworten der MQTT-Anfrage

Unter der Eigenschaft „Server“ des „smart_shelf/response“-Knotens (Abbildung 87) ist die IP-Adresse des MQTT-Brokers anzugeben, an den die Nachricht gesendet werden soll. In diesem Fall wird es an die Adresse „localhost:1883“ gesendet, da zur Veranschaulichung des Prozesses der MQTT-Broker am gleichen Rechner läuft. Der verschickten Nachricht an den MQTT-Broker wird ein „Topic“ zugewiesen, unter dessen Clients die Nachricht einsehen können. Der „delay“-Knoten „Delay to accomodate software limitations“ ist ein Knoten zur Verzögerung der Sendung der Antwort auf die MQTT-Anfrage um 2 Sekunden (Abbildung 86).

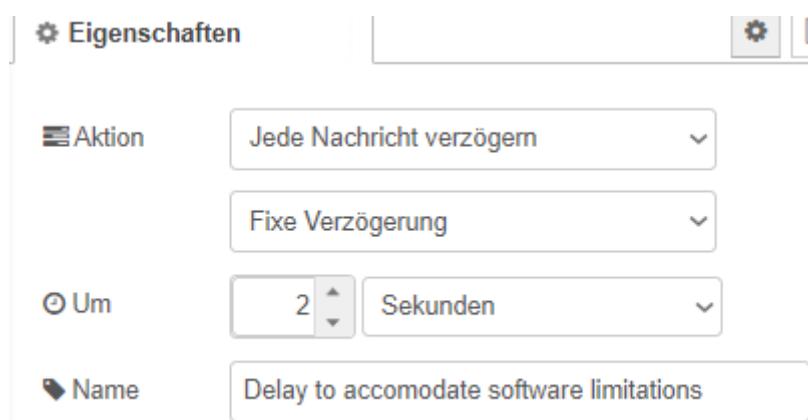


Abbildung 86: Eigenschaften des Knotens "Delay to accomodate software limitations"

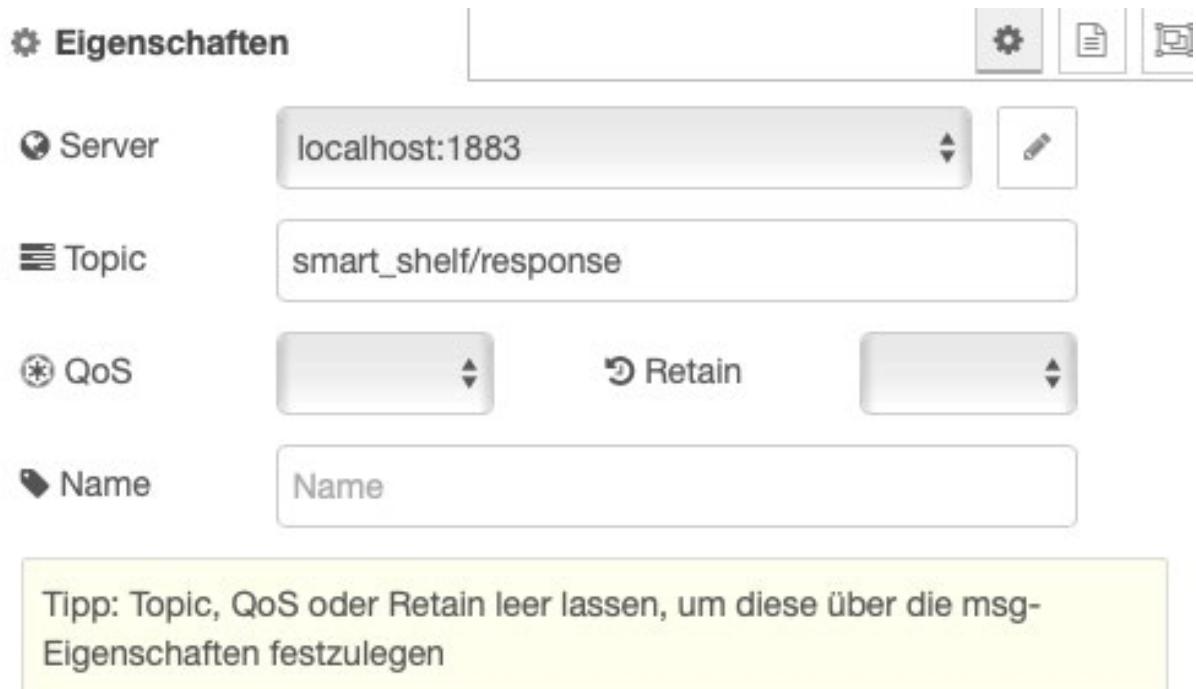


Abbildung 87: Eigenschaften des Knotens "smart_shelf/response"

7.1.8 Aktiveren eines Lichts basierend auf dem Zustand

In diesem Schritt wird anhand dem „status“ (Abbildung 88), welches im „Determine status“- Knoten bestimmt wurde, eines der drei Lichter Grün, Gelb oder Rot (Abbildung 89) des LED-Ampelmoduls aktiviert.

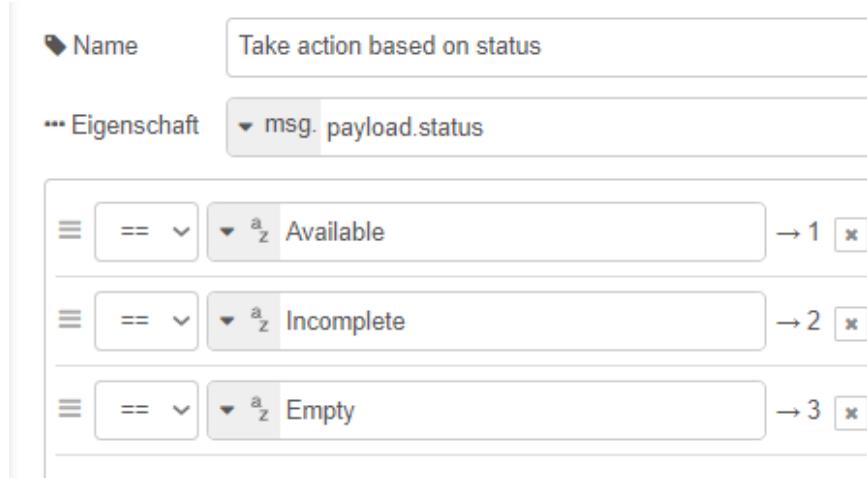


Abbildung 88: mögliche Zustände

Grünes Licht bedeutet nicht nur, dass eine Box des Typs „A“ verfügbar ist, sondern auch das eine aus dem Sortiment entnommen wird, weshalb bei grünem Licht der Prozess weitergeführt werden muss, um die Anzahl der noch verfügbaren Boxen in den nächsten Schritten zu korrigieren. Das gelbe Licht wird aktiviert, wenn der Status „Incomplete“ eintritt. Status „Empty“ aktiviert das rote Licht.

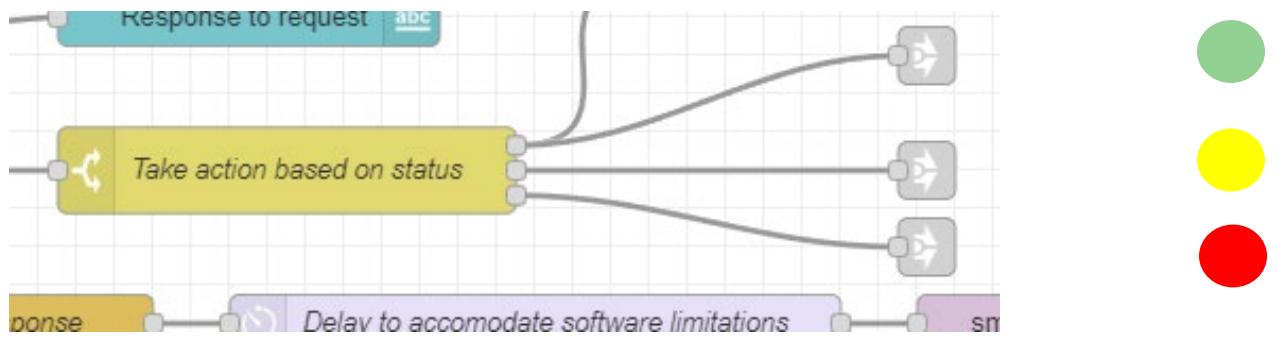


Abbildung 89: Knoten zur Aktivierung der Lichter des LED-Ampelmoduls

Die jeweiligen „Link out“-Knoten (Abbildung 89) sind mit ihren zuständigen „Link in“-Knoten (Abbildung 90) verbunden. Bei einem „status“ von „Available“ befinden wir uns in der Gruppe „Green light box A“, welches in einem „Switch“-Knoten „Activate light“ das grüne Licht aktiviert und durch einen anderen „Link out“-Knoten und „Switch“-Knoten „Deactivate light“ alle anderen Lichter ausschaltet. Der gleiche Prozess gilt auch bei Eintreten von gelbes und rotes Licht.

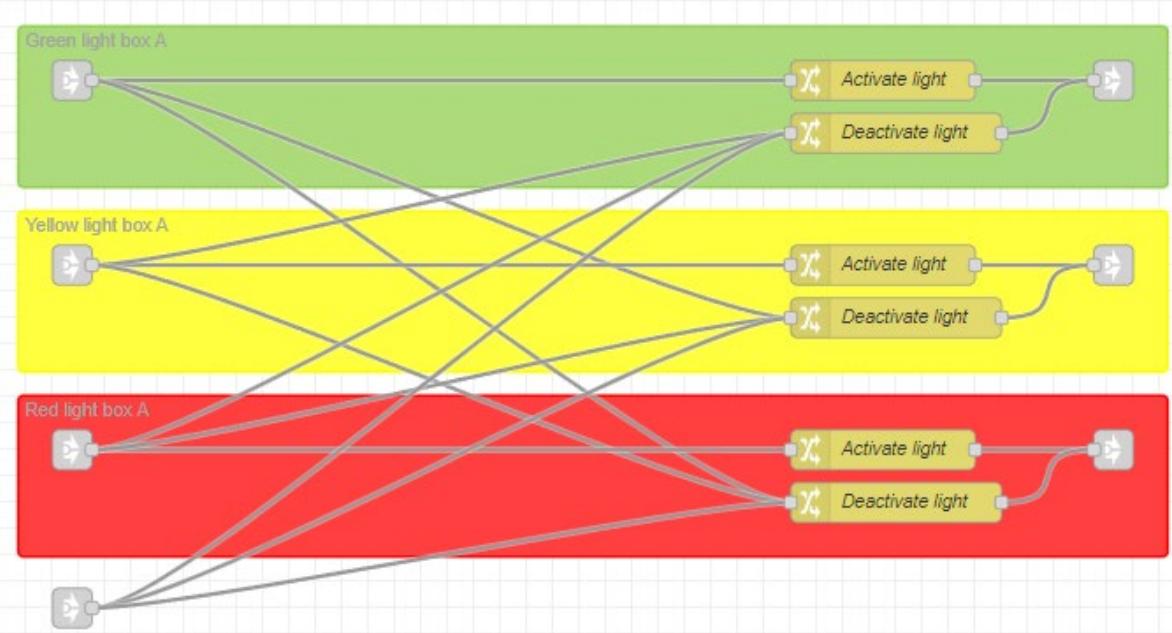


Abbildung 90: Zuständige Gruppen inklusive Knoten zur Aktivierung/Deaktivierung der Lichter

Der „Link in“-Knoten, welches sich außerhalb der Gruppen befindet, hat Verbindungen zu allen „Deactivate light“-Knoten der verschiedenen Lichter. Beim Bestimmen der Anzahl der Boxen im Dashboard wird der beschriebene „Link in“-Knoten ausgelöst und alle Lichter des LED-Ampelmoduls werden ausgeschaltet.

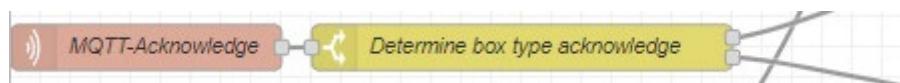


Abbildung 91: MQTT-Nachricht, dass Prozess abgeschlossen wurde

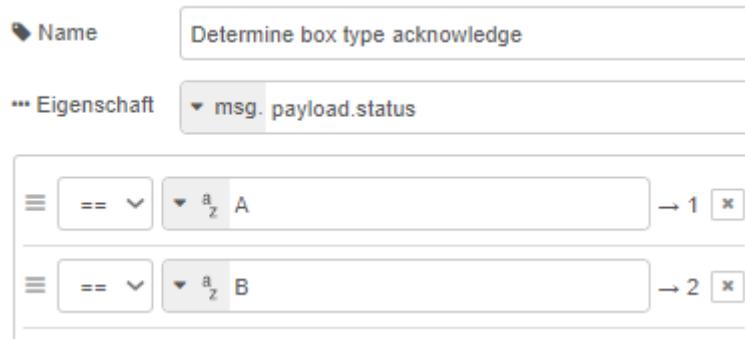


Abbildung 92: Eigenschaften des Knotens "Determine box type acknowledge"

Der „Receiver“-Knoten „MQTT-Acknowledge“ (Abbildung 91) kennzeichnet eine MQTT-Nachricht, die vom Spot gesendet wird, um Auskunft darüber zu geben, dass der Prozess abgeschlossen wurde. Im „switch“-Knoten „Determine box type acknowledge“ (Abbildung 91) wird je nach erfüllter Bedingung (Abbildung 92) der Prozess zum „Link out“-Knoten (Abbildung 68) weitergeleitet. Der genannte „Link out“-Knoten ist mit dem „Link out“-Knoten (Abbildung 90) der außerhalb der drei Lichtergruppen befindet und welcher alle Lichter des LED-Ampelmoduls ausschaltet, da der Prozess fertig ist.

7.1.9 Reduzieren der Boxanzahl

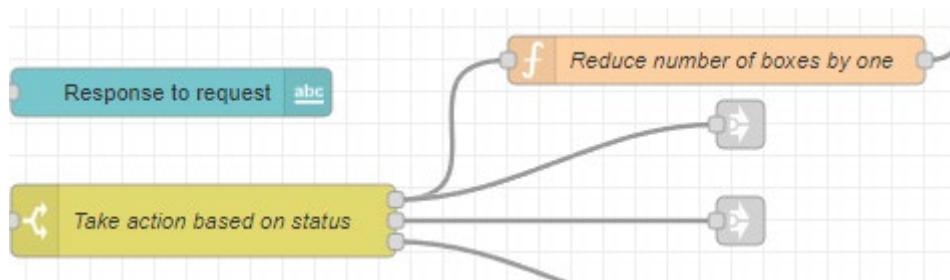


Abbildung 93: Knoten zur Reduzierung der Boxanzahl um 1

Der „Reduce number of boxes by one“-Knoten (Abbildung 93) ist die Fortsetzung des Prozesses bei grünem Licht. Im Knoten wird die „flow.set()“ Methode verwendet, um die Variable „boxA“ (Abbildung 94) neu zu definieren, da wie vorhin erwähnt, bei grünem Licht eine Box aus dem Sortiment entnommen wird und die Anzahl der Boxen korrigiert werden muss. Der Wert von „boxA“ wird durch Abziehen von „1“ von seinem aktuellen Wert, welches durch die Methode „flow.get('boxA')“ erhalten wird, neu definiert. Im Anschluss wird der eingetroffene Payload der Nachricht unverändert weitergeleitet.

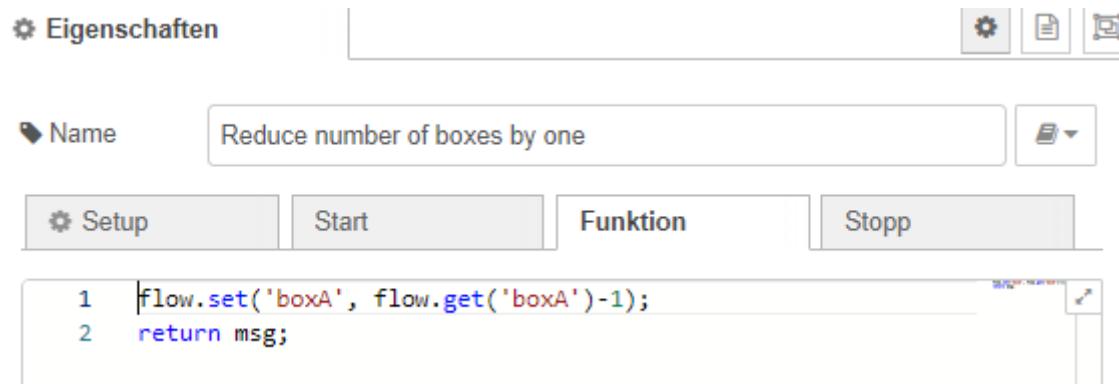


Abbildung 94: Eigenschaften des Knotens "Reduce number of boxes by one"

7.1.10 Setzen von Variablen zur Veranschaulichung am User-Interface und Display

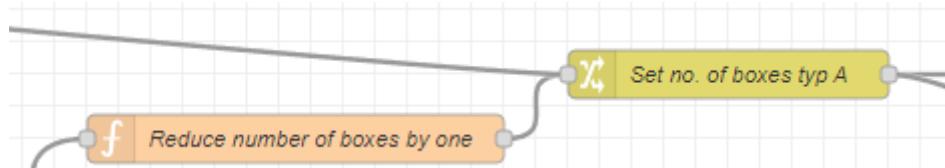


Abbildung 95: Knoten zur Setzung von Variablen

Im Funktions-Knoten „Set no. of boxes typ A“ (Abbildung 95) wird die vorhin veränderte Variable „boxA“ zur Veranschaulichung am Dashboard bzw. User-Interface verändert. Der Variable „msg.payload“ (Abbildung 96) wird der zuletzt gespeicherte Wert der Variable „flow.boxA“ zugewiesen, um diese im Anschluss am User-Interface anzeigen. Unter der Variable „msg.topic“ wird der selbe Wert in Form von einem String gespeichert, um diese am Display, welches sich am Mikrokontroller befindet, anzeigen.

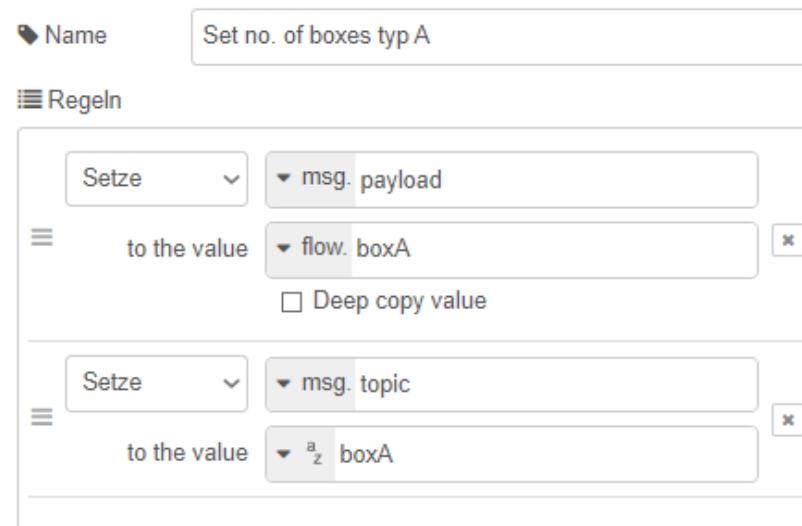


Abbildung 96: Eigenschaften des Knotens "Set no. of boxes typ A"

7.1.11 Anzeigen der vorhandenen Boxen

„Amount of boxes left“ (Abbildung 97) ist ein weiter User-Interface-Knoten, welche nur zum Anzeigen von Texten bzw. Daten verwendet wird.

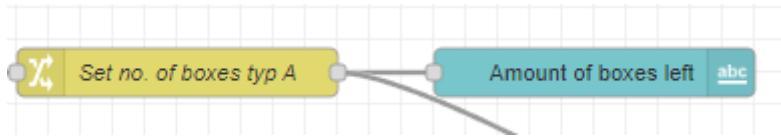


Abbildung 97: Knoten zur Anzeige am User-Interface

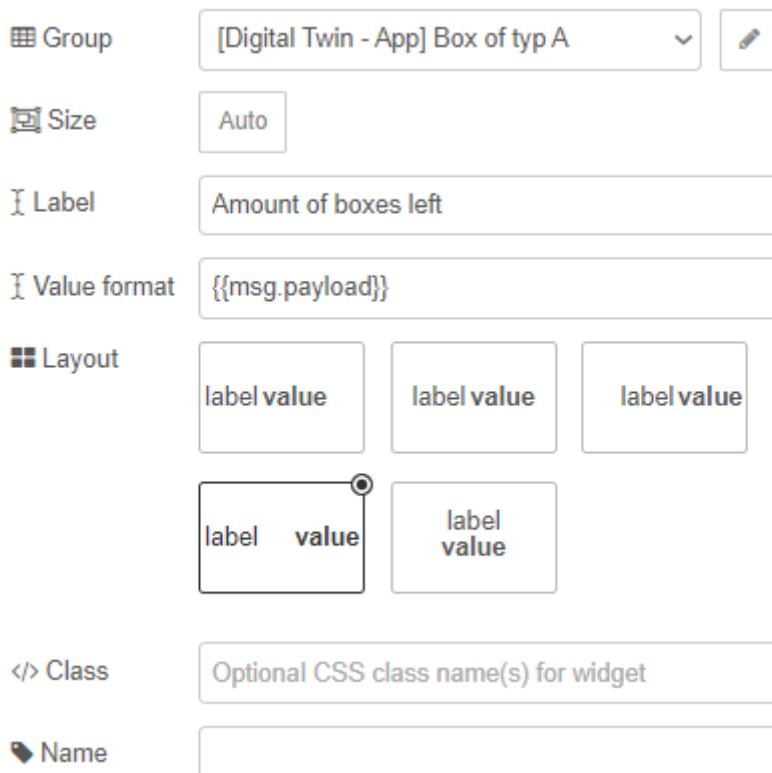


Abbildung 98: Eigenschaften des Knotens "Amount of boxes left"

Im genannten Text-Knoten wird unter „Value format“ (Abbildung 98) die vorhin gesetzte Variable „msg.payload“, die am User-Interface angezeigt werden soll, hinzugefügt. Unter „Layout“ wird die Ansichtsform des Knotens auf dem User-Interface ausgewählt. Abbildung 73 veranschaulicht die Ansicht des „Amount of boxes left“-Knotens auf dem Dashboard.

7.1.12 Zusammenführung der Anzahl der Boxen des Typs „A“ und „B“

Der Sequenz-Knoten "Join Box information" (Abbildung 99) ist dafür zuständig,

Nachrichten über die Anzahl von Boxen vom Typ „A“ und „B“ zu verbinden.

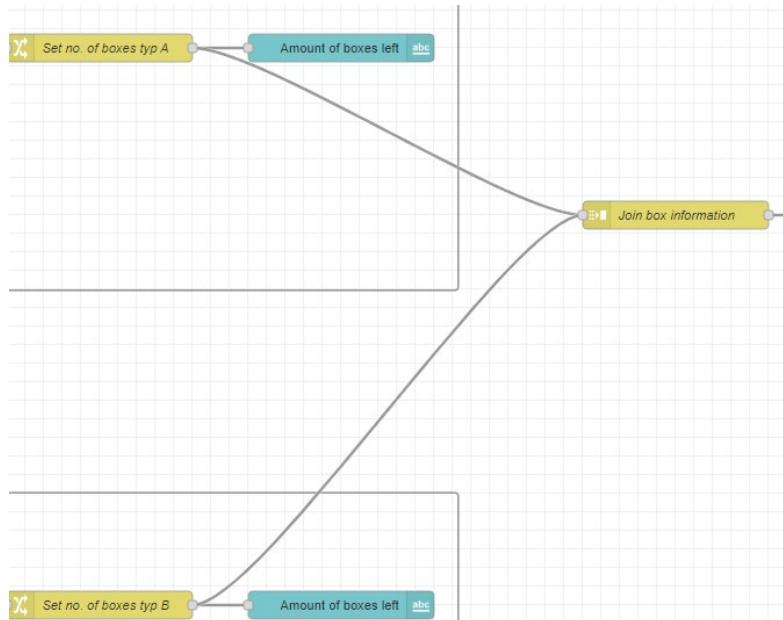


Abbildung 99: Knoten zur Zusammeführung von Variablen

Diese Information ist wichtig für die Darstellung am Display.

Modus	<input type="button" value="Manuell"/>	
Verbinde jede	<input type="button" value="▼ msg. payload"/>	
und erstelle	<input type="button" value="ein Schlüssel/Wert-Objekt"/>	
mit dem Wert von	<input type="button" value="msg. topic"/>	als Schlüssel
Senden der Nachricht:		
<ul style="list-style-type: none"> • Nach einer Anzahl von Nachrichtenteilen <input type="text" value="1"/> <input checked="" type="checkbox"/> und bei jeder nachfolgenden Nachricht • Bei Zeitablauf nach erster Nachricht von <input type="button" value="Sekunder"/> • Nach Nachricht mit <code>msg.complete</code>-Eigenschaft 		
	<input type="button" value="Join box information"/>	

Abbildung 100: Eigenschaften des Knotens "Join box information"

Da dieser Knoten mit dem Prozessablauf des Boxtyps „A“ und „B“ verbunden ist, wird in diesem Knoten die Variable „payload“ beider Prozesse mit ihrem jeweiligen Wert der Variable „topic“ als Schlüssel (Abbildung 100) verbunden, um anschließend diese am Display anzuzeigen.

Wenn eine neue Übernahme (deploy) durchgeführt wird, kann es vorkommen, dass die Informationen über die Anzahl von Boxen verloren gehen. In diesem Fall werden die Werte in der Benutzeroberfläche jedoch weiterhin angezeigt, da sie im User-Interface gespeichert sind. Bei einer neuen Anfrage werden die Werte zum angefragten Boxtyp dem User-Interface entnommen, andernfalls werden je nach Boxtyp Nullwerte angezeigt.

7.1.13 Erstellen der Nachricht zur Veranschaulichung am Display

„Create display“-Knoten (Abbildung 101) verwendet JSONata, um die Darstellung von Daten am Display zu erstellen.

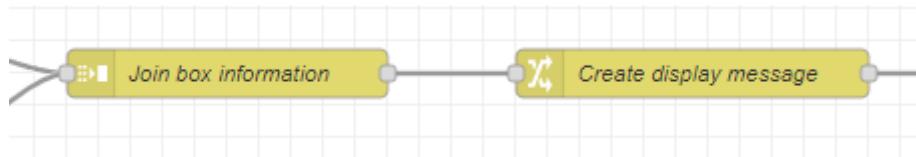


Abbildung 101: Knoten zum Setzen einer Variable zur Veranschaulichung am Display

Hierbei wird das msg.payload-Objekt (Abbildung 102) in ein JSONata-Format umgewandelt, um die Informationen in der gewünschten Form anzuzeigen und die Positionen der Elemente festzulegen. Die Bezeichnung „boxA-label“ sowie „boxB-label“ ist irrelevant und kann nach Bedarf umbenannt werden, da sie lediglich zum besseren Verständnis der Elemente beitragen.

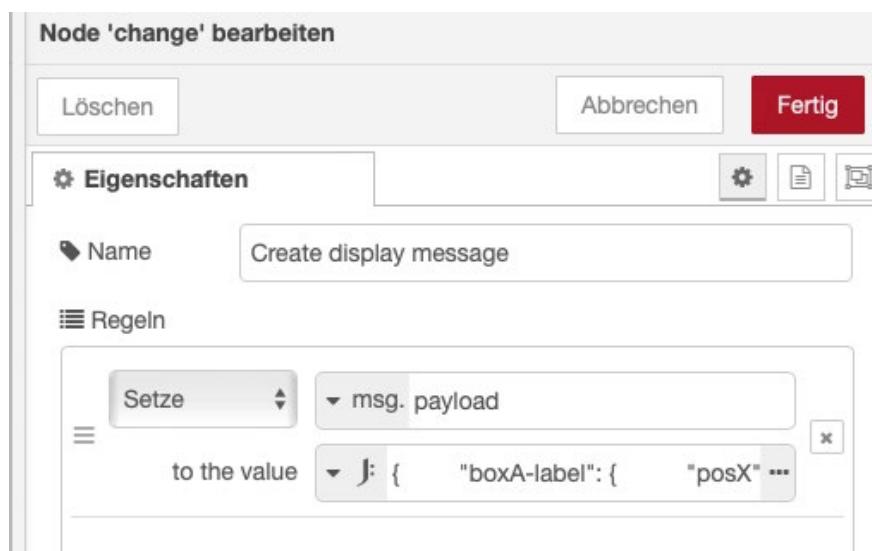


Abbildung 102: Eigenschaften des Knotens "Create display message"

7.1.14 Einfügen einer Verzögerung, um Mikrocontroller nicht zu überlasten

Der Delay-Knoten "Delay..." (Abbildung 103) verzögert den Nachrichtenfluss für die Darstellung um 0,5 Sekunden (Abbildung 104) um eine bessere Anpassung an die Anforderungen der Anwendung zu ermöglichen. Hierdurch wird sichergestellt, dass der Fluss der Nachrichten ordnungsgemäß zur Darstellung am Display weitergeleitet werden. Der Grund für die Verzögerung um 0,5 Sekunden ist eine Überlagerung der Signale an die Lichter mit dem Signal an das Display am Mikrocontroller ESP32. Die Nachricht kann sonst nicht ordnungsgemäß vom Display verarbeitet werden. Durch ein „Link out“-Knoten (Abbildung 103) werden die Nachrichten, die zuletzt zusammengeführt und in ein JSONata-Format gebracht wurden, an ein „Link in“-Knoten innerhalb des „Connections“-Flow geschickt, um diese im Anschluss auf dem Display des ESP32-001 anzuzeigen.

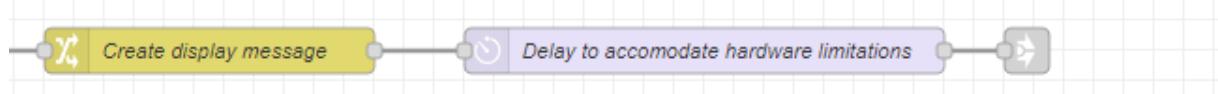


Abbildung 103: Knoten zum Einfügen einer Verzögerung

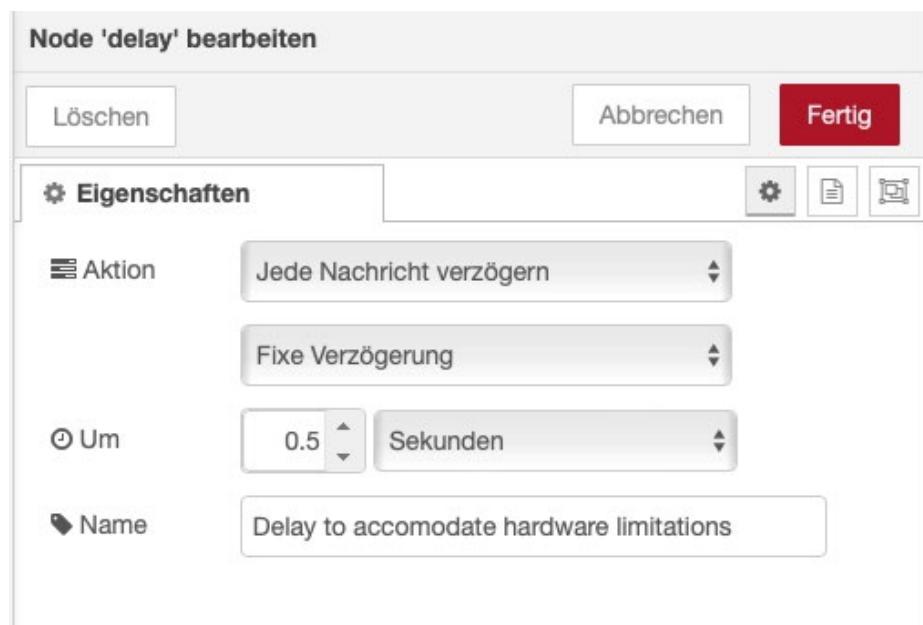


Abbildung 104: Eigenschaften des Knotens "Delay to accomodate hardware limitations"

7.2 Betriebsanleitung Smart-room

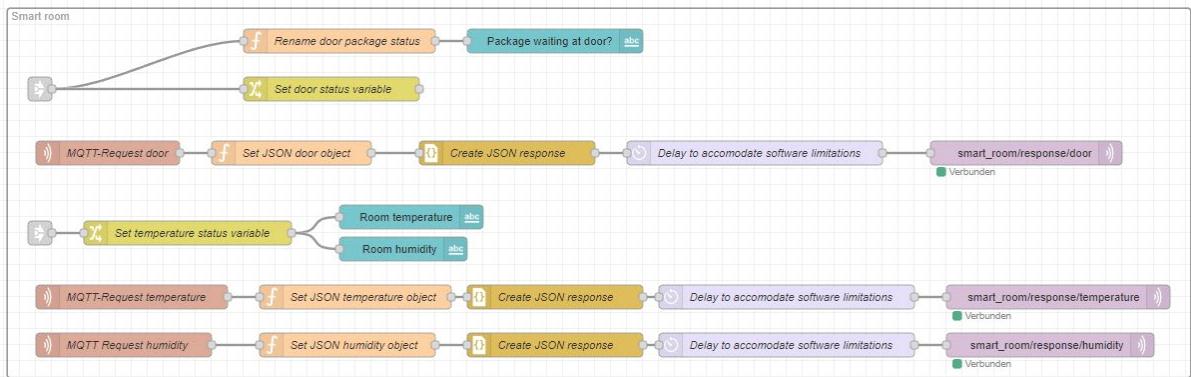


Abbildung 105: Gesamtbild des Prozesses Smart-room

„Smart-room“ ist ein Prozess (Abbildung 105) der innerhalb des gleichen Node-RED-Projekts und parallel zum „Smart-shelf“-Prozess läuft. Dieser Prozess läuft jedoch auf einem anderen Mikrokontroller, ESP8266-001, als der von „Smart-shelf“. In diesem Prozess werden Raumtemperatur und Luftfeuchtigkeit des Raums ermittelt und dem Nutzer auf dem Dashboard angezeigt. Zudem wird durch ein Infrarotsensor an der Tür des Raums erfasst, ob sich ein Objekt an der Tür befindet, um dies dem Spot zu übermitteln, damit dieser in Aktion tritt.

7.2.1 Setzen und Ändern von Variablen und Anzeige am User-Interface

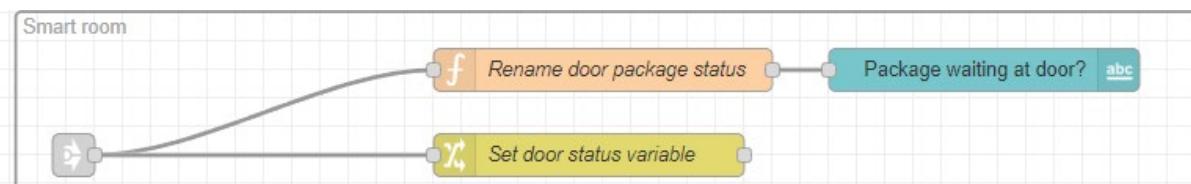


Abbildung 106: Knoten zum Setzen/Ändern von Variablen und Anzeigen von Information am User-Interface

Auf den „Link in“-Knoten (Abbildung 106), welches den Infrarotsensor an der Tür repräsentiert und die Nutzdaten dessen speichert, folgen zwei weitere Knoten, die die Nutzdaten zur Weiterverwendung und für das Verständnis modifizieren und erleichtern.



Abbildung 107: Eigenschaften des Knotens "Rename door package status"

Im Funktions-Knoten „Rename door package status“ (Abbildung 106) werden die Nutzdaten des Infrarotsensors umgewandelt. Falls „msg.payload“ „1“ entspricht wird „1“ in ein „Yes“ (Abbildung 107) umgewandelt. Bei einem Wert von „0“ wird ein „No“ gesetzt. Der Grund für diese Änderung kann durch den nächsten Knoten, „Package waiting at door?“ besser verstanden werden.

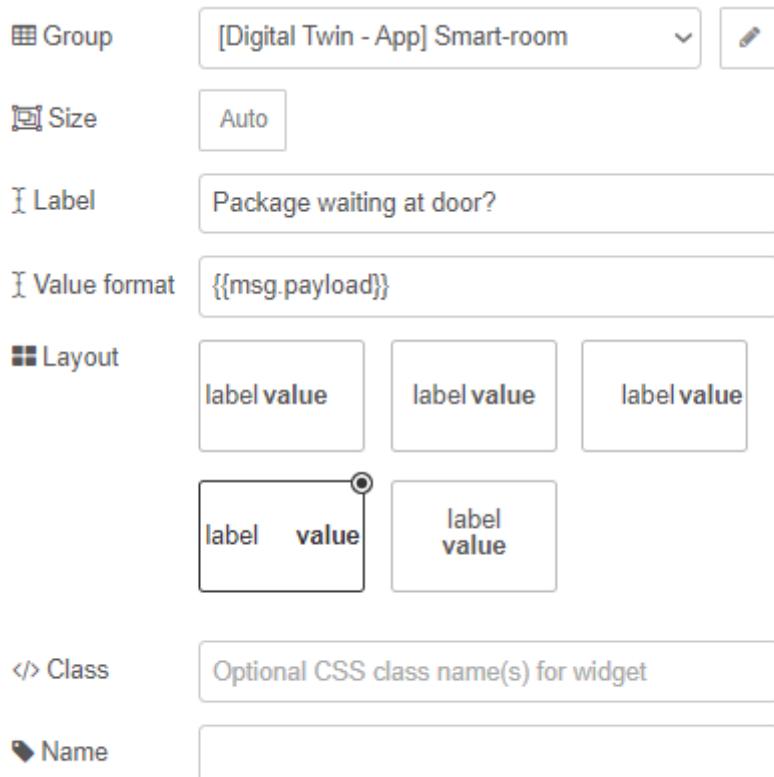


Abbildung 108: Eigenschaften des Knotens "Package waiting at door?"

Im Text-Knoten „Package waiting at door?“ (Abbildung 106) wird „msg.payload“ (Abbildung 108) auf dem Dashboard angezeigt.

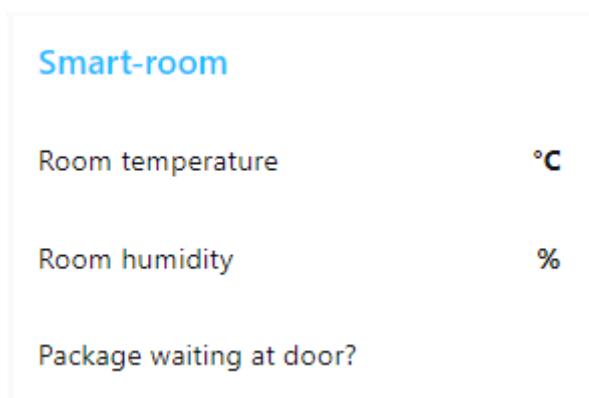


Abbildung 109: Ausschnitt des User-Interfaces

Je nachdem, ob sich eine Box an der Tür befindet oder nicht, wird dies am Dashboard unter „Package waiting at door?“ (Abbildung 109) durch ein „Yes“ oder „No“ gekennzeichnet.

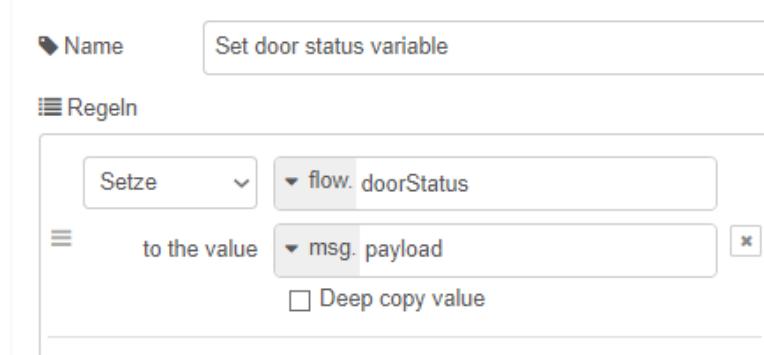


Abbildung 110: Eigenschaften des Knotens "Set door status variable"

Im Change-Knoten „Set door status variable“ (Abbildung 106) werden die Nutzdaten der Variable „msg.payload“ in der neuen Variable „doorStatus“ (Abbildung 110) gespeichert, um diese innerhalb des ganzen Flows nutzen zu können.

7.2.2 Erstellen einer MQTT-Antwort zu eingehender MQTT-Anfrage



Abbildung 111: Knoten zur Erstellung einer MQTT-Antwort auf eine MQTT-Anfrage

Unter dem „Receiver“-Knoten „MQTT-Request door“ (Abbildung 111) wird eine MQTT-Anfrage bezüglich des Status an der Tür abgefangen.

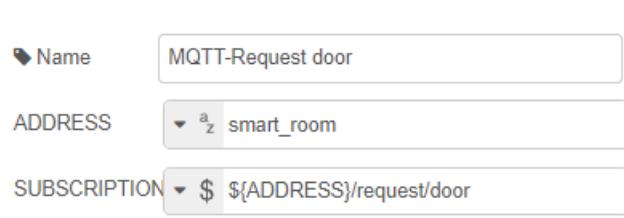


Abbildung 112: Eigenschaften des Knotens "MQTT-Request door"

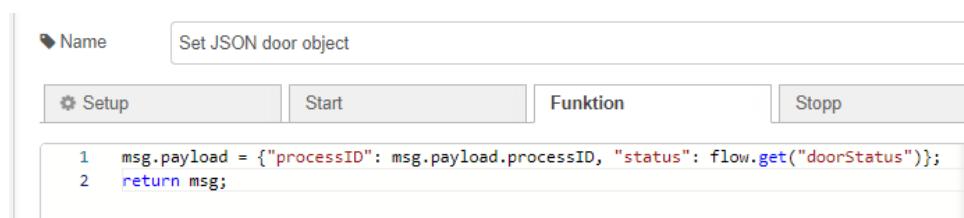
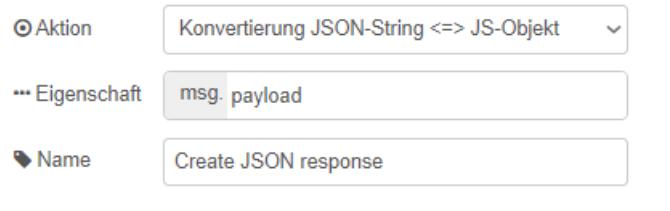


Abbildung 113: Eigenschaften des Knotens "Set JSON door object"

Unter dem Topic „smart_room/request/door“ (Abbildung 112) kann die Anfrage gestellt werden. Im Anschluss wird im Funktions-Knoten „Set JSON door object“ und „json“-Knoten

„Create JSON response“ (Abbildung 111) der Inhalt der Variable „doorStatus“ (Abbildung 113) dem Parameter „status“ zugewiesen und diese Nachricht zu einem JSON-Objekt (Abbildung 114) konvertiert.



Optionen Objekt => JSON

JSON-Zeichenfolge formatieren

Abbildung 114: Eigenschaften des Knotens "Create JSON response"

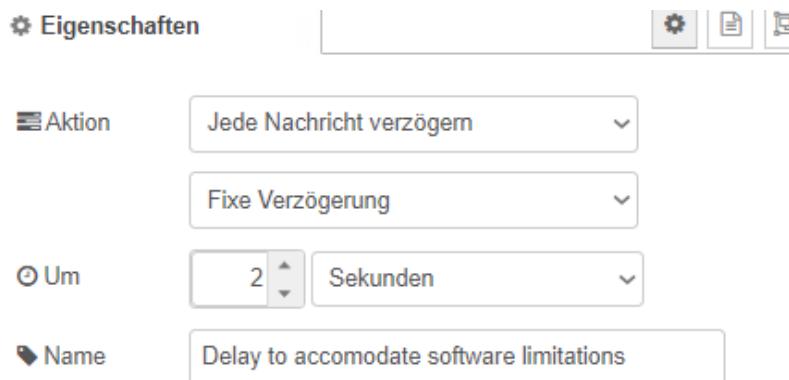


Abbildung 115: Eigenschaften des Knotens "Delay to accomodate software limitations"

Der „delay“-Knoten wurde eingefügt, um die Antwort auf die MQTT-Anfrage um 2 Sekunden (Abbildung 115) zu verzögern, da das für den Camunda Showcase Prozess verwendete Projekt der Gruppe B die Antworten ohne eine solche Verzögerung nicht ordnungsgemäß verarbeiten kann. Gleiches gilt für die „delay“-Knoten aus Abbildung 121.

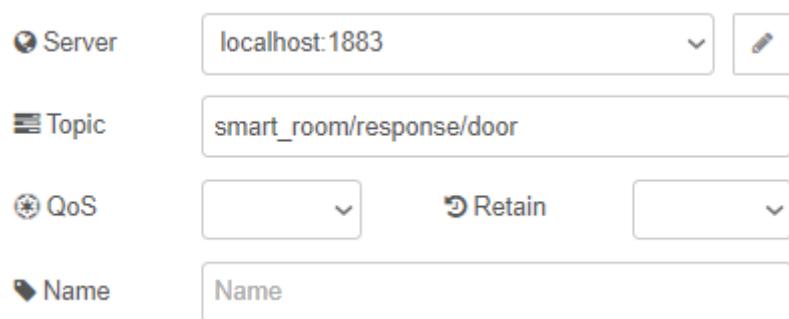


Abbildung 116: Eigenschaften des Knotens "smart_room/response/door"

Im „mqtt out“-Knoten „smart_room/response/door“ (Abbildung 111) wird unter dem Topic „smart_room/response/door“ (Abbildung 116) die vorhin erstellte Antwort als JSON-Objekt an die MQTT-Anfrage zurückgegeben.

7.2.3 Setzen von Variablen zum Anzeigen am User-Interface

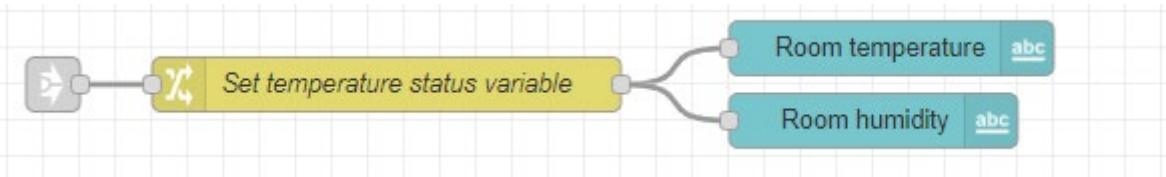


Abbildung 117: Knoten zur Anzeige am User-Interface

Der „Link in“-Knoten (Abbildung 117) repräsentiert einen Sensor, der die Temperatur und Luftfeuchtigkeit des Raums ermittelt.

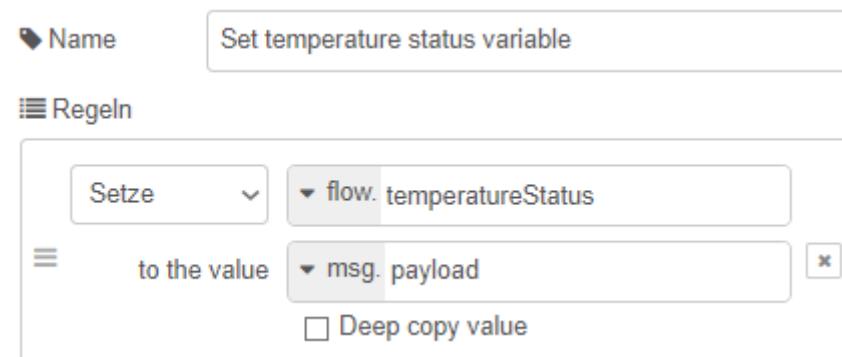


Abbildung 118: Eigenschaften des Knotens "Set temperature status variable"

Im „Change“-Knoten „Set temperature status variable“ werden die Nutzdaten der Variable „payload“ in der neuen Variable „temperatureStatus“ (Abbildung 118) gespeichert, um auf diese Daten in anderen Knoten zugreifen zu können.

Die „text“-Knoten „Room temperature“ und „Room humidity“ (Abbildung 117) sind wiederum Knoten zur Veranschaulichung der Nutzdaten am User-Interface (Abbildung 109). Die Temperatur wird in „°C“ (Abbildung 119) und Luftfeuchtigkeit in „%“ (Abbildung 120) angezeigt.

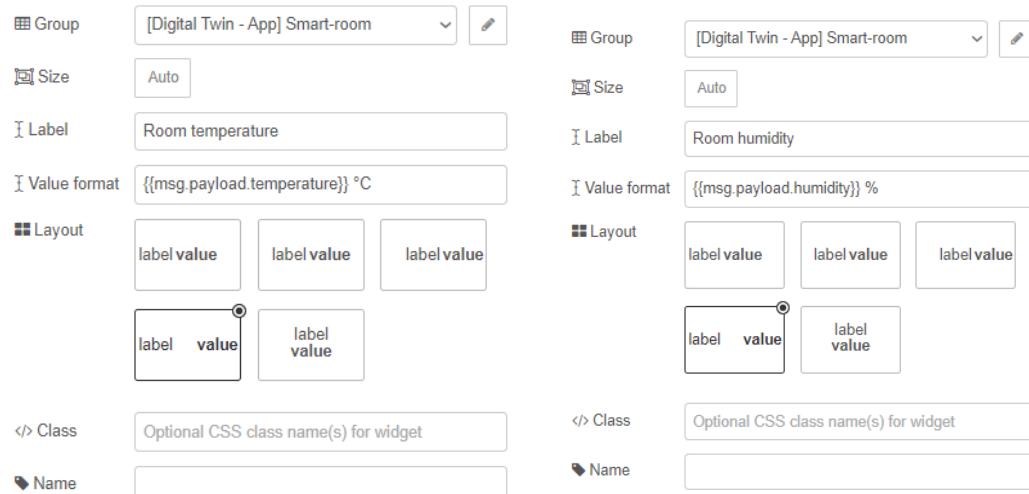


Abbildung 119: Eigenschaften des Knotens "Room temperature"
 Abbildung 120: Eigenschaften des Knotens "Room humidity"

7.2.4 Erstellen von MQTT-Antworten zu eingehenden MQTT-Anfrage

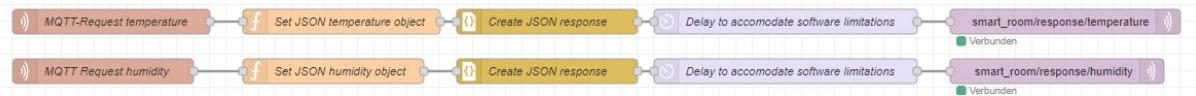


Abbildung 121: Knoten zur Erstellung einer MQTT-Antwort auf eine MQTT-Anfrage

Unter den „Receiver“-Knoten „MQTT-Request temperature“ und „MQTT Request humidity“ (Abbildung 121) wird eine MQTT-Anfrage bezüglich des Status der Temperatur und Luftfeuchtigkeit abgefangen. Unter dem Topic „smart_room/request/temperature“ (Abbildung 122) und „smart_room/request/humidity“ (Abbildung 123) kann die Temperatur und die Luftfeuchtigkeit abgefragt werden.

Name MQTT-Request temperature ADDRESS smart_room SUBSCRIPTION \${ADDRESS}/request/temperature	Name MQTT Request humidity ADDRESS smart_room SUBSCRIPTION \${ADDRESS}/request/humidity
---	---

Abbildung 122: Eigenschaften des Knotens „MQTT-Request temperature“
 Abbildung 123: Eigenschaften des Knotens „MQTT-Request humidity“

Auch hier werden der Nachricht in einem Funktions-Knoten „Set JSON...“ (Abbildung 121) die **Daten der Variable „temperatureStatus“ (Abbildung 124) dem Parameter „status“ zugewiesen**. Durch „flow.get(„temperatureStatus“).temperature“ wird explizit ausgedrückt, dass nur die Daten bezüglich der Temperatur zugewiesen werden, da in der Variable „temperatureStatus“ sowohl die Temperatur als auch die Luftfeuchtigkeit abgespeichert ist. Gleiches gilt für die Luftfeuchtigkeit (Abbildung 125).

Name Set JSON temperature object Setup Start Funktion Stopp <pre> 1 msg.payload = {"processID":msg.payload.processID, "status": flow.get("temperatureStatus").temperature} 2 return msg; </pre>

Abbildung 124: Eigenschaften des Knotens "Set JSON temperature object"

Name Set JSON humidity object Setup Start Funktion Stopp <pre> 1 msg.payload = {"processID":msg.payload.processID, "status": flow.get("temperatureStatus").humidity} 2 return msg; </pre>

Abbildung 125: Eigenschaften des Knotens "Set JSON humidity object"

Im Anschluss werden die Nachrichten von einem String-Format durch ein „json“-Knoten „Create JSON response“ (Abbildung 121) in ein JSON-Objekt konvertiert. Zuletzt werden diese erstellten Nachrichten in einem „mqtt out“-Knoten „smart_room/response/...“ (Abbildung 121) unter den Topics „smart_room/response/temperature“ (Abbildung 126) und „smart_room/response/humidity“ (Abbildung 127) zur Verfügung gestellt.

 Server	localhost:1883		 Server	localhost:1883			
 Topic	smart_room/response/temperature		 Topic	smart_room/response/humidity			
 QoS		 Retain		 QoS		 Retain	
 Name	Name		 Name	Name			

Abbildung 126: Eigenschaften des Knotens "smart_room/response/temperature"

Abbildung 127: Eigenschaften des Knotens "smart_room/response/humidity"

8. Reflexion

Als abschließendes Kapitel wird in den folgenden Unterkapiteln eine Reflexion aus Sicht der Projektgruppe hinsichtlich des Projekts durchgeführt.

8.1 Reflexion der Umsetzung

Zu Beginn der Lehrveranstaltung konnten wir uns unter dem Projekt noch wenig vorstellen, da die zu verwendenden Technologien für uns neu waren und wir den Use-Case noch nicht ganz verstanden hatten. Von Termin zu Termin haben wir besseres Verständnis entwickelt und uns bei der Orientierung gegenseitig unterstützt. Die Teamarbeit hat sehr gut funktioniert, da wir uns stets gegenseitig geholfen haben und über Discord und WhatsApp miteinander kommuniziert haben. Regelmäßige Gruppenbesprechungen waren notwendig, da nur ein Gruppenmitglied die Hardware zuhause hatte. Bei der Bedienung der Hardware haben uns die Lehrveranstaltungsleiter und der Techniker Markus Jahn geholfen. Bei Fragen und Problemen konnten wir uns immer an die Leiter der Lehrveranstaltung wenden, welche uns bei der Durchführung des Projekts unterstützt haben. Auch wenn uns die Umsetzung des Projekts nicht immer einfach gefallen ist, konnten wir mit etwas Hilfe das Projekt erfolgreich umsetzen.

8.2 Reflexion des Projektverlaufs

Im Verlauf des Projekts Smart-shelf gab es verschiedene Herausforderungen, die es zu bewältigen galt. Ein Stolperstein war beispielsweise, dass es nicht genügend Lichtschranken gab, um das Regal wie ursprünglich geplant auszustatten. Das Team musste daraufhin kreativ werden und alternative Lösungen finden, wie zum Beispiel den Einsatz von Infrarotsensoren. Hieraus wurde gelernt, dass ein lösungsorientiertes Arbeiten und eine gewisse Flexibilität bei der Umsetzung von Projekten von großer Bedeutung sind. Ein weiterer Stolperstein war die Terminfindung, da das Projektteam aus fünf Personen bestand. Es mussten Teiltermine gefunden werden, um alle unter einen Hut zu bringen. Dies erforderte, eine gute Kommunikation und Kollaboration innerhalb des Teams, um asymmetrisches gemeinsames Arbeiten zu ermöglichen. Auch hieraus wurde gelernt, dass eine offene und transparente Kommunikation unerlässlich ist, um ein Projekt erfolgreich abzuschließen. Schließlich gab es auch eine Herausforderung im Hinblick auf das Wissen des Teams im Bereich der Hardware. Da das Wirtschaftsinformatik-Studium kaum Berührungspunkte mit Hardware hatte, musste das Team umfangreiche Recherchen durchführen, um die Grundzüge zum Konfigurieren der Hardware zu verstehen. Diese Erfahrung hat gezeigt, wie wichtig es ist, sich auch über Themenbereiche hinaus zu bilden und über den eigenen Tellerrand hinauszuschauen. Insgesamt hat das Projektteam aus diesen Herausforderungen gelernt, dass ein lösungsorientiertes Arbeiten, eine offene Kommunikation

und eine gewisse Flexibilität unerlässlich sind, um ein Projekt erfolgreich abzuschließen. Zudem wurde deutlich, wie wichtig es ist, sich auch über das eigene Fachgebiet hinaus Wissen anzueignen und eine gewisse Kreativität an den Tag zu legen, um unerwartete Probleme zu lösen.

8.3 Reflexion der Ideen

Für die Umsetzung des Projekts mussten aufgrund bestimmter Gründe einige Ideen verworfen werden. Die Reservation von Boxen in einer lokalen Datenbank ist eine der Ideen, die verworfen werden, musste. Aufgrund unpassender Hardware konnte die Idee nicht vollzogen werden, weshalb Alternativen zum Reservieren von Boxen (ohne Implementation aufgrund fehlender Hardware) angeführt wurden (siehe Seite 19). Um die Verwerfung dieser Idee vorzubeugen, sollte im Vorhinein die vorhandene Hardware getestet werden und dementsprechend gehandelt werden, denn am Anfang des Projekts wurde der Flash-Speicher des Mikrocontroller als lokale Datenbank vorgeschlagen, welche jedoch nach einer gewissen Zeit aufgrund von vielen Anfragen funktionsunfähig werden würde. Eine weitere Idee, die verworfen werden musste, ist die Implementation eines Roboterarms in den Gesamtprozess. Aufgrund fehlender Verfügbarkeit eines Roboterarms des Instituts für Communications Engineering für die Nutzung dessen am Prozess des „Smart-shelfs“ konnte diese Ideen nicht umgesetzt werden. Falls die Implementation einer Idee bis zum Start des Projekts noch ungewiss ist, sollte der Prozess bzw. das Use-case ohne jegliche Berücksichtigung der ungewissen Ideen begonnen werden. Das Design vieler Knoten sowie Flows in Node-RED mussten entsprechend dem Design des Projekts der Gruppe B umgesetzt werden. Darunter sind Knoten wie „Receiver“-Knoten und „Sender“-Knoten zum Abfangen und Senden von MQTT-Nachrichten sowie „delay“-Knoten zur Verzögerung von MQTT-Antworten, da das Projekt der Gruppe B die Antworten ohne Verzögerung nicht ordnungsgemäß verarbeiten kann. Das Design der Antworten mit der „processId“ sowie dem „status“ sind ebenfalls Vorgaben des Projekts der Gruppe B, weshalb auf diese Rücksicht genommen und das Design beibehalten werden musste. Die Absprache zu etwaigen Designentscheidungen sollte zwischen den Gruppen möglichst am Anfang des Projekts erfolgen, um inkonsistente Verknüpfungen und Stress am Ende des Projekts vorzubeugen.

8.4 Reflexion der Hilfestellungen

Während der Planung und der Umsetzung des Projekts war es notwendig, uns Wissen über die verwendeten Technologien und Werkzeuge anzusehen. Dazu bezogen wir Hinweise und Informationen aus verschiedensten Quellen.

Während der gesamten Umsetzung, aber besonders in der Planungsphase des Projekts erhielten wir Input und Unterstützung durch die Leitung der Lehrveranstaltung. Da uns in der Planungsphase noch einige grundlegende Dinge zum Projektziel unklar waren kamen besonders anfänglich viele Fragen auf. Zwischen den zweiwöchigen Terminen konnten wir uns mit Fragen und Unklarheiten allerdings immer an die Leitung der Lehrveranstaltung wenden. Für diese Betreuung möchten wir uns besonders bei Herrn Mag. Richard Heininger und Herrn Thomas Jost bedanken. Neben der LVA- Leitung möchten wir auch Herrn Dr. Markus Jahn danken, der sich Zeit nahm, uns die Grundlagen zu den Themen Microcontroller und Node-RED näher zu bringen und uns einige Templates zur Verfügung stellte. Dieser Crashkurs erleichterte uns den Einstieg in das Themengebiet ungemein.

Informationen zu der verwendeten Hardware bezogen wir vor allem aus der jeweiligen Dokumentation des Herstellers (AZ- Delivery) und aus verschiedenen Foren und Blogs, die sich mit den einschlägigen Themengebieten befassen. Die Informationen zur Software und den verwendeten Technologien bezogen wir ebenfalls hauptsächlich aus den offiziellen Dokumentationen. Besonders hilfreich waren dabei die Dokumentationen zu Node-RED, PlatformIO CLI und MQTT. Außerdem hatten wir Kontakt zu den anderen Projektgruppen der Lehrveranstaltung und konnten uns so vor allem über die Umsetzung der MQTT- Schnittstelle und des Camunda Showcase Prozess austauschen.

8.5 Reflexion der Projektinhalte

Als wichtigster Inhalt des Projekts ist das Erlernen und die Nutzung des Entwicklungswerkzeugs Node-RED zu kategorisieren. Node-RED ist nicht mit anderen bis dato im Studium angelernten Technologien im Bereich des Entwickelns vergleichbar. Der Grund für diese Differenz liegt im Schwerpunkt von Node-RED auf das grafische Entwickeln mit Hilfe von Flows und Nodes und der daraus resultierenden fehlenden Notwendigkeit für Programmierkenntnisse. Die stärkste Ähnlichkeit zu einem bereits bekannten Werkzeug besitzt die Camunda-Plattform durch deren Nutzung von ähnlichen Konzepten wie Flows und Nodes zur Darstellung von Prozessen. Die Stärke von Node-RED liegt aus Sicht der Projektgruppe in der schnell anlernbaren und intuitiven Nutzung der graphischen Benutzeroberfläche, welche wie bereits erwähnt keine Programmierkenntnisse für die Umsetzung von Projekten voraussetzt. Node-RED eignet sich durch das nutzerfreundliche Verändern von Nodes und Flows hervorragend für das schnelle Erreichen und Anpassen von Projektenergebnissen. Diese Fähigkeit erwies sich besonders im Rahmen des Prototyping von verschiedenen Lösungsszenarien in der Anfangsphase des Projekts als sehr hilfreich. Eine Schwäche von Node-RED bestand darin, dass bei komplexeren Projekten viele Flows zu zeichnen sind, welche sich überlagern und dadurch ein komplexes Gesamtbild

schaffen. Die Beibehaltung des Überblicks über das Projekt war für das Projektteam im Projektverlauf stark erschwert. Ebenso sind grundlegende Funktionalitäten wie das Verändern von JSON-Objekten nicht vorhanden und erzwingen dadurch die Nutzung von nicht nutzerfreundlichen Umwegen wie der Schaffung von neuen Objekten. Herausforderungen bei der Nutzung von Node-RED für das Projektteam haben sich insgesamt jedoch durch eine gute Dokumentation von Seiten des Node-RED-Projekts, einer vorherigen Konfiguration durch Dr. Markus Jahn und einem großen Community-Support in Grenzen gehalten. Zusammenfassend ist Node-RED daher als ein sehr hilfreiches Tool zu sehen, welches seine Schwächen durch sehr große Stärken ausgleichen kann. Als ebenso bedeutsamen Projektinhalt ist das Arbeiten mit Hardware anzusehen. Für die Projektgruppe hat dieser Teil des Projekts eine große Abweichung von bisherig bekannten Inhalten dargestellt. Der Wechsel von bisherig softwarenahem Entwickeln zu hardwarenahem Entwickeln hat das Erlernen der Grundzüge von bis dato als gegeben angenommener Hardware notwendig gemacht. Es war entsprechend notwendig die Dokumentationen der Mikrocontroller und Module und der damit verbundenen Möglichkeiten zu deren Konfiguration zu studieren, welche bis dato im Rahmen des Studiums in keiner bedeutenden Kapazität angesprochen wurde. Für das Projektteam hat das eine doch größere Herausforderung dargestellt, welche jedoch entsprechend gemeistert werden konnte. Die Hardware selbst hat seine Stärken durch eine hohe Resistenz gegenüber verschiedenen Konfigurationen und damit verbundenen Fehlern gezeigt. Dem Projektteam war hingegen die Beschränkungen der Hardware negativ aufgefallen, welche in eine Reduzierung der anvisierten Funktionalitäten resultierte. Als Beispiel dazu ist auf die Problematik des Flash-Speichers im Rahmen der Datenbank-Funktionalität zu verweisen.

9. Abbildungsverzeichnis

Abbildung 1: Projektschema	8
Abbildung 2: Projektstrukturplan	15
Abbildung 3: Gantt-Diagramm.....	16
Abbildung 4: ESP32 Pinbelegung	52
Abbildung 5: Schaltplan Smart-shelf	58
Abbildung 6: Steckplatine Smart-shelf	59
Abbildung 7: ESP8266 Pinbelegung	60
Abbildung 8: Schaltplan Smart-shelf	65
Abbildung 9: Steckplatine Smart-room.....	66
Abbildung 10: Versorgungspins der Sensoren	67
Abbildung 11: Gesamtarchitektur Smart-shelf & Smart-room.....	68
Abbildung 12: Git-Versionsüberprüfung	70
Abbildung 13: Erstellung eines Ordners	71
Abbildung 14: Navigieren zu erstelltem Ordner	71
Abbildung 15: geklontes Repository.....	71
Abbildung 16: NVM-Installationsschritt 1.....	72
Abbildung 17: NVM-Installationsschritt 2.....	72
Abbildung 18: NVM-Installationsschritt 3.....	73
Abbildung 19: NVM-Installationsschritt 4.....	73
Abbildung 20: NVM-Installationsschritt 5.....	74
Abbildung 21: NVM-Installationsschritt 6.....	74
Abbildung 22: Versionsüberprüfung	75
Abbildung 23: Installationskommando Node.js.....	75
Abbildung 24: Versionsüberprüfung Node.js	75
Abbildung 25: Ausführung von PlatformIO CLI Skript.....	76
Abbildung 26: Setzen der Systemvariablen für PlatfotmIO CLI	77
Abbildung 27: Bestimmung des Ports eines Mikrocontrollers	78
Abbildung 28: Setzen der Ports für PlatformIO.....	78
Abbildung 29: Konfigurieren des Mikrocontrollers vor dem Flashen	79
Abbildung 30: Navigieren zum geklonten Repository	80
Abbildung 31: Navigieren zum geklonten Node-RED-Ordner	80
Abbildung 32: Ausführung von "npm install"	80
Abbildung 33: Ausführung von "npx grunt build"	81
Abbildung 34: Geklontes Repository innerhalb VSC	82
Abbildung 35: Terminal innerhalb VSC	83
Abbildung 36: Erstellung eines Node-RED-Projekts.....	83

Abbildung 37: Starten des erstellten Projekts.....	84
Abbildung 38: Terminal von MQTT-Server öffnen.....	84
Abbildung 39: Terminal von MQTT-Server.....	85
Abbildung 40: Starten des MQTT-Servers	85
Abbildung 41: Ändern der Serveradresse innerhalb von Node-RED	85
Abbildung 42: Überprüfung der Verbindung zu MQTT 1.0	86
Abbildung 43: Überprüfung der Verbindung zu MQTT 2.0	86
Abbildung 44: Versionsüberprüfung von Git.....	87
Abbildung 45: Installationskommando für Git	87
Abbildung 46: Ausführung von "sudo apt update"	87
Abbildung 47: Installationskommando von curl	88
Abbildung 48: Datei zur Installation von VSC	88
Abbildung 49: Ausführung des Installationskommandos von VSC	89
Abbildung 50: Installationskommando für Python (Version 3)	90
Abbildung 51: Klonen des Repositorys	90
Abbildung 52: Installationskommando für Node Version Manager	91
Abbildung 53: Installation Node.js	92
Abbildung 54: zum geklonten Repository navigieren.....	92
Abbildung 55: Ausführung von "npm install"	92
Abbildung 56: Ausführung von "npx grunt build"	93
Abbildung 57: Erstellen eines Node-RED-Projekts.....	93
Abbildung 58: Starten des erstellten Projekts.....	93
Abbildung 59: Installieren von .NET6.0	94
Abbildung 60: Starten des MQTT-Servers	94
Abbildung 61: Ändern der Serveradresse innerhalb von Node-RED	95
Abbildung 62: Gesamtbild des Node-RED-Prozesses.....	96
Abbildung 63: Node-RED Flows	96
Abbildung 64: MQTT-Anfrageknoten (links), Knoten zum Ermitteln des Boxtyps (rechts)	97
Abbildung 65: MQTT-Topic	97
Abbildung 66: Eigenschaften des Switch-Knotens	98
Abbildung 67: Prozessausschnitt für Boxtyp A	98
Abbildung 68: Form-Knoten (links), Change-Knoten (mitte), Link out-Knoten (rechts).....	98
Abbildung 69: Eigenschaften des Knotens "Input amount of boxes available".....	99
Abbildung 70: User-Interface	99
Abbildung 71: Seitenleiste am User-Interface	100
Abbildung 72: Auswahl des Geräts, Pins und Modus	100
Abbildung 73: Ausschnitt des User-Interfaces.....	101

Abbildung 74: Eigenschaften des Knotens "Set amount of boxes available"	101
Abbildung 75: Verbindung zwischen zwei Knoten zur Erstellung und Setzung einer Variable ..	101
Abbildung 76: Eigenschaften des Knotens "Set detection for box typ A" (Erstellung und Setzung einer Variable)	102
Abbildung 77: Eigenschaften des Knotens "Determine status"	103
Abbildung 78: Knoten zur Veranschaulichung am User-Interface	103
Abbildung 79: Eigenschaften des Knotens "Response to request"	104
Abbildung 80: Veranschaulichung eines Zustands am User-Interface.....	104
Abbildung 81: Funktionsknoten "Set JSON object" zur Erweiterung des vorhandenen JSON-Strings	104
Abbildung 82: Erweiterung des JSON-Strings um "typ".....	105
Abbildung 83: Knoten zum Konvertieren des JSON-Strings zum JSON-Objekt	105
Abbildung 84: Eigenschaften des Knotens "Create JSON response"	106
Abbildung 85: Knoten zum Beantworten der MQTT-Anfrage.....	106
Abbildung 86: Eigenschaften des Knotens "Delay to accomodate software limitations"	106
Abbildung 87: Eigenschaften des Knotens "smart_shelf/response".....	107
Abbildung 88: mögliche Zustände	107
Abbildung 89: Knoten zur Aktivierung der Lichter des LED-Ampelmoduls.....	108
Abbildung 90: Zuständige Gruppen inklusive Knoten zur Aktivierung/Deaktivierung der Lichter	108
Abbildung 91: MQTT-Nachricht, dass Prozess abgeschlossen wurde	108
Abbildung 92: Eigenschaften des Knotens "Determine box type acknowledge"	109
Abbildung 93: Knoten zur Reduzierung der Boxanzahl um 1	109
Abbildung 94: Eigenschaften des Knotens "Reduce number of boxes by one"	110
Abbildung 95: Knoten zur Setzung von Variablen	110
Abbildung 96: Eigenschaften des Knotens "Set no. of boxes typ A".....	110
Abbildung 97: Knoten zur Anzeige am User-Interface.....	111
Abbildung 98: Eigenschaften des Knotens "Amount of boxes left"	111
Abbildung 99: Knoten zur Zusammeführung von Variablen.....	112
Abbildung 100: Eigenschaften des Knotens "Join box information"	112
Abbildung 101: Knoten zum Setzen einer Variable zur Veranschaulichung am Display	113
Abbildung 102: Eigenschaften des Knotens "Create display message"	113
Abbildung 103: Knoten zum Einfügen einer Verzögerung	114
Abbildung 104: Eigenschaften des Knotens "Delay to accomodate hardware limitations"	114
Abbildung 105: Gesamtbild des Prozesses Smart-room	115
Abbildung 106: Knoten zum Setzen/Ändern von Variablen und Anzeigen von Information am User-Interface	115

Abbildung 107: Eigenschaften des Knotens "Rename door package status"	115
Abbildung 108: Eigenschaften des Knotens "Package waiting at door?"	116
Abbildung 109: Ausschnitt des User-Interfaces.....	116
Abbildung 110: Eigenschaften des Knotens "Set door status variable"	117
Abbildung 111: Knoten zur Erstellung einer MQTT-Antwort auf eine MQTT-Anfrage	117
Abbildung 112: Eigenschaften des Knotens "MQTT-Request door"	117
Abbildung 113: Eigenschaften des Knotens "Set JSON door object"	117
Abbildung 114: Eigenschaften des Knotens "Create JSON response"	118
Abbildung 115: Eigenschaften des Knotens "Delay to accomodate software limitations"	118
Abbildung 116: Eigenschaften des Knotens "smart_room/response/door"	118
Abbildung 117: Knoten zur Anzeige am User-Interface.....	119
Abbildung 118: Eigenschaften des Knotens "Set temperature status variable"	119
Abbildung 119: Eigenschaften des Knotens "Room temperature"	119
Abbildung 120: Eigenschaften des Knotens "Room humidity"	119
Abbildung 121: Knoten zur Erstellung einer MQTT-Antwort auf eine MQTT-Anfrage	120
Abbildung 122: Eigenschaften des Knotens „MQTT-Request temperature“	120
Abbildung 123: Eigenschaften des Knotens „MQTT-Request humidity“	120
Abbildung 124: Eigenschaften des Knotens "Set JSON temperature object.....	120
Abbildung 125: Eigenschaften des Knotens "Set JSON humidity object"	120
Abbildung 126: Eigenschaften des Knotens "smart_room/response/temperature"	121
Abbildung 127: Eigenschaften des Knotens "smart_room/response/humidity"	121

10. Tabellenverzeichnis

Tabelle 1: LED- Ampel Definition	7
Tabelle 2: Meilensteine	9
Tabelle 3: Arbeitspaket P1	10
Tabelle 4: Arbeitspaket P2	10
Tabelle 5: Arbeitspaket P3	10
Tabelle 6: Arbeitspaket P4	11
Tabelle 7: Arbeitspaket I1	11
Tabelle 8: Arbeitspaket I2	11
Tabelle 9: Arbeitspaket I3	12
Tabelle 10: Arbeitspaket I4	12
Tabelle 11: Arbeitspaket I5	12
Tabelle 12: Arbeitspaket I6	12
Tabelle 13: Arbeitspaket I7	13
Tabelle 14: Arbeitspaket I8	13
Tabelle 15: Arbeitspaket T1	13
Tabelle 16: Arbeitspaket T2	13
Tabelle 17: Arbeitspaket E1	14
Tabelle 18: Arbeitspaket E2	14
Tabelle 19: Arbeitspaket E3	14
Tabelle 20: Arbeitspaket E4	14
Tabelle 21: Test 1.1	22
Tabelle 22: Test 1.2	23
Tabelle 23: Test 1.3	24
Tabelle 24: Test 1.4	25
Tabelle 25: Test 1.5	26
Tabelle 26: Test 2.1	27
Tabelle 27: Test 2.2	28
Tabelle 28: Test 2.3	29
Tabelle 29: Test 2.4	30
Tabelle 30: Test 2.5	31
Tabelle 31: Test 3.1	32
Tabelle 32: Test 3.2	33
Tabelle 33: Test 3.3	34
Tabelle 34: Test 3.4	35
Tabelle 35: Test 3.5	36
Tabelle 36: Test 3.6	37

Tabelle 37: Test 3.7	38
Tabelle 38: Test 4.1	39
Tabelle 39: Test 4.2	40
Tabelle 40: Test 4.3	41
Tabelle 41: Test 4.4	42
Tabelle 42: Test 5.1	43
Tabelle 43: Test 5.2	44
Tabelle 44: Test 5.3	45
Tabelle 45: Test 6.1	46
Tabelle 46: Test 6.2	47
Tabelle 47: Test 6.3	48
Tabelle 48: Test 6.4	49
Tabelle 49: Test 7.1	50
Tabelle 50: Test 7.2	51
Tabelle 51: Verwendung der Pins am ESP32	52
Tabelle 52: Abbildungen ESP32	53
Tabelle 53: Abbildungen 9V- Batterie/ Akku	53
Tabelle 54: Abbildungen Spannungswandler	54
Tabelle 55: Beschaltung des Spannungswandlers	54
Tabelle 56: Abbildungen Display	55
Tabelle 57: Beschaltung des I2C-Displays	55
Tabelle 58: Abbildungen Infrarotsensor	56
Tabelle 59: Beschaltung Infrarotsensoren	56
Tabelle 60: Abbildungen LED-Ampel	57
Tabelle 61: Beschaltung LED-Ampeln	57
Tabelle 62: Verwendung der Pins am ESP32	60
Tabelle 63: Abbildungen ESP8266	61
Tabelle 64: Abbildungen 9V- Batterie/ Akku	61
Tabelle 65: Abbildungen Spannungswandler	62
Tabelle 66: Beschaltung des Spannungswandlers	62
Tabelle 67: Abbildungen Infrarotsensor	63
Tabelle 68: Beschaltung Infrarotsensoren	63
Tabelle 69: Abbildungen Temperatur- und Feuchtigkeitssensor	64
Tabelle 70: Beschaltung Temperatur- und Feuchtigkeitssensor	64
Tabelle 71: Schnittstellen Smart-shelf & Smart-room	69

11. Anhang

Auf den folgenden Seiten ist die Übersicht über die Zeitaufzeichnung aller Projektteammitglieder als auch deren individuellen Zeitaufzeichnungen zu finden.

Zeitaufzeichnung - Gruppe C

*Tabelle wird automatisch befüllt!

Meilenstein	Matthias Schwarz	Ozan Akkoyun	Berkan Kalkan	Betim Sulejmani	Andrea Wieser	Summe
Projektplanung	51,50	39,33	53,75	48,17	54,25	247,00
Node-RED aufsetzen	17,25	21,67	31,42	34,25	6,50	111,08
Hardware- Prototyp	22,15	16,75	0,00	0,00	2,00	40,90
finale Hardware	1,25	10,45	0,00	0,00	3,00	14,70
Node-RED UI	3,42	7,33	3,50	0,00	6,50	20,75
MQTT Anbindung	13,67	9,17	2,00	2,00	17,00	43,83
Systemtest	12,00	11,25	14,50	14,50	12,50	64,75
Projektbericht	28,17	39,42	45,75	49,08	51,25	213,67
Summe	149,40	155,37	150,92	148,00	153,00	756,68

Name: Matthias Schwarz
Matrikelnummer: 12022973

Summe: 149,40

Datum	Start	Ende	Tätigkeit	Meilenstein	Zeit	Stunden
13.10.2022	09:00	10:00	Vorbesprechung	Projektplanung	01:00	1,00
17.10.2022	15:00	19:30	Erstellen des grundsätzlichen BPMN Modells	Projektplanung	04:30	4,50
31.10.2022	14:30	18:30	Erweitern des BPMN Modells um Ausführbarkeit	Projektplanung	04:00	4,00
02.11.2022	08:30	10:00	LVA-Einheit: Besprechung der Prozessmodelle	Projektplanung	01:30	1,50
17.11.2022	13:45	15:00	LVA-Einheit: Besprechung des Projektplans und kurze Einführung in Node-RED	Projektplanung	01:15	1,25
21.11.2022	09:00	14:15	Projektplanung durchführen	Projektplanung	05:15	5,25
24.11.2022	11:00	17:20	Ziele/ Nicht- Ziele definieren, Projektplanung, ...	Projektplanung	06:20	6,33
25.11.2022	13:00	18:30	Projektplanung weiter durchführen (Definieren von Arbeitspaketen und Meilensteinen)	Projektplanung	05:30	5,50
26.11.2022	18:00	22:30	Formulierung der Einführung, Problemstellung und Schaffung von Jira	Projektplanung	04:30	4,50
28.11.2022	18:30	21:45	Projektplanung abschließen	Projektplanung	03:15	3,25
01.12.2022	13:45	16:00	LVA-Einheit: Präsentation der Projektpläne, PC- Konfiguration für Projekt	Projektplanung	02:15	2,25
05.12.2022	09:00	13:30	Einarbeiten in Themen: MicroController, Sensorik, NodeRED	Node-RED aufsetzen	04:30	4,50
13.12.2022	18:15	22:00	Hardware Kompetenz erweitern, Sensorliste, ...	Hardware- Prototyp	03:45	3,75
15.12.2022	13:45	14:30	LVA-Einheit: Besprechung des Fortschritts	Hardware- Prototyp	00:45	0,75
18.12.2022	10:00	18:00	Node- Red Dokumentation + Einarbeitung	Node-RED aufsetzen	08:00	8,00
20.12.2022	09:00	11:15	Workshop: Mikrocontroller, Node-RED	Projektbericht	02:15	2,25
22.12.2022	12:15	17:00	Aufsetzen der notwendigen Software (Node-RED, PlatformIO und etc.) in Windows	Node-RED aufsetzen	04:45	4,75
23.12.2022	10:15	15:30	erhaltene Hardware testen, ...	Hardware- Prototyp	05:15	5,25
30.12.2022	14:40	17:30	Hardware Prototyp erstellen	Hardware- Prototyp	02:50	2,83
31.12.2022	14:40	17:30	Hardware Prototyp erweitern	Hardware- Prototyp	02:50	2,83
12.01.2023	13:45	15:00	LVA-Einheit: Besprechung des Fortschritts	Projektplanung	01:15	1,25
13.01.2023	15:15	16:55	Teambesprechung für weitere Vorgehen und Fortschritt	Projektplanung	01:40	1,67
13.01.2023	19:00	23:59	projektspezifische Anpassung der Hardware	Hardware- Prototyp	04:59	4,98
14.01.2023	00:00	01:45	Erweiterungen Hardware	Hardware- Prototyp	01:45	1,75
14.01.2023	19:05	22:30	Node-RED Funktionalität für smart-room implementieren	Node-RED UI	03:25	3,42
26.01.2023	13:45	14:30	LVA-Einheit: Update	Projektplanung	00:45	0,75
29.01.2023	14:00	15:30	Teambesprechung für weiteres Vorgehen	Projektplanung	01:30	1,50
29.01.2023	19:05	23:10	Camunda Showcase Prozess beginnen und Dokumentation PlatformIO	MQTT Anbindung	04:05	4,08
01.02.2023	09:50	13:15	Vorbereitung und Besprechung mit Gruppe Infrastruktur	MQTT Anbindung	03:25	3,42
04.02.2023	11:30	14:00	Teambesprechung	Projektplanung	02:30	2,50
04.02.2023	14:15	20:25	Camunda Showcase Prozess erstellen	MQTT Anbindung	06:10	6,17
05.02.2023	13:00	14:30	Teambesprechung	Projektplanung	01:30	1,50
07.02.2023	10:00	16:00	Testen der Teilsysteme mit CamundaProzess	Systemtest	06:00	6,00
08.02.2023	11:00	17:00	Testen der Teilsysteme mit CamundaProzess	Systemtest	06:00	6,00
09.02.2023	13:45	15:25	Schaltplan erstellen (techn. Handbuch)	Projektbericht	01:40	1,67
10.02.2023	13:45	17:30	technisches Handbuch schreiben	Projektbericht	03:45	3,75
11.02.2023	09:00	13:00	Steckplatine erstellen (techn. Handbuch)	Projektbericht	04:00	4,00
12.02.2023	13:45	15:00	Abgabe Hardware & Camunda Showase Prozess bearbeiten	finale Hardware	01:15	1,25
13.02.2023	18:00	21:30	technisches Handbuch überarbeiten	Projektbericht	03:30	3,50
14.02.2023	10:00	18:00	technisches Handbuch überarbeiten	Projektbericht	08:00	8,00
24.02.2023	14:30	17:30	Teambesprechung - Abgabe, Überarbeitung, ...	Projektplanung	03:00	3,00
26.02.2023	13:00	18:00	Projektabchluss und Fertigstellen des Abschlussberichts	Projektbericht	05:00	5,00

Name: Berkan Kalkan
Matrikelnummer: 12018666

Summe: 150,92

Datum	Start	Ende	Tätigkeit	Meilenstein	Zeit	Stunden
13.10.2022	09:00	10:00	Vorbesprechung	Projektplanung	01:00	1,00
17.10.2022	15:00	19:30	Erstellen des grundsätzlichen BPMN Modells	Projektplanung	04:30	4,50
31.10.2022	14:30	18:30	Erweitern des BPMN Modells um Ausführbarkeit	Projektplanung	04:00	4,00
02.11.2022	08:30	10:00	LVA-Einheit: Besprechung der Prozessmodelle	Projektplanung	01:30	1,50
17.11.2022	13:45	15:00	LVA-Einheit: Besprechung des Projektplans und kurze Einführung in Node-RED	Projektplanung	01:15	1,25
21.11.2022	09:00	14:15	Projektplanung durchführen	Projektplanung	05:15	5,25
24.11.2022	11:00	17:20	Projektplanung (Ziele definieren)	Projektplanung	06:20	6,33
25.11.2022	13:00	18:30	Projektplanung weiter durchführen (Definieren von Arbeitspaketen und Meilensteinen)	Projektplanung	05:30	5,50
26.11.2022	18:00	22:30	Formulierung der Einführung, Problemstellung und Schaffung von Jira	Projektplanung	04:30	4,50
28.11.2022	18:30	21:45	Projektplanung abschließen	Projektplanung	03:15	3,25
01.12.2022	13:45	16:00	LVA-Einheit: Präsentation der Projektpläne, Konfiguration des Rechners für das Projekt	Projektplanung	02:15	2,25
03.12.2022	16:00	22:45	Node-RED Dokumentation erarbeiten	Node-RED aufsetzen	06:45	6,75
05.12.2022	18:00	23:45	Hardware-Kompetenzen aneignen	Projektbericht	05:45	5,75
10.12.2022	14:00	19:00	MQTT-Dokumentation erarbeiten	Node-RED aufsetzen	05:00	5,00
15.12.2022	13:45	14:15	LVA-Einheit: Besprechung des Fortschritts	Node-RED aufsetzen	00:30	0,50
20.12.2022	09:00	11:15	Workshop in Mikrocontroller, Node-RED	Node-RED aufsetzen	02:15	2,25
25.12.2022	16:15	22:45	Aufsetzen der notwendigen Software (Node-RED, Platformio und etc.) in Windows	Node-RED aufsetzen	06:30	6,50
27.12.2022	18:35	22:30	Einarbeiten in Node-RED (Grundverständnisse umsetzen und vorgefertigtes Beispiel probieren)	Node-RED aufsetzen	03:55	3,92
29.12.2022	13:45	14:30	LVA-Einheit: Besprechung des Fortschritts	Node-RED aufsetzen	00:45	0,75
02.01.2023	18:00	23:45	Node-RED Anbindung Teil 1 (Hardwareanbindung)	Node-RED aufsetzen	05:45	5,75
03.01.2023	17:30	21:00	Node-RED Anbindung Teil 2 (User-Interface)	Node-RED UI	03:30	3,50
04.01.2023	19:00	21:00	MQTT-Anbindung	MQTT Anbindung	02:00	2,00
07.01.2023	12:00	13:15	Recherche zu Datenbank Teil 1	Projektbericht	01:15	1,25
08.01.2023	10:30	12:45	Recherche zu Datenbank Teil 2	Projektbericht	02:15	2,25
12.01.2023	13:45	14:00	LVA-Einheit: Besprechung des Fortschritts	Projektplanung	00:15	0,25
13.01.2023	15:15	16:55	Teambesprechung für weitere Vorgehen	Projektplanung	01:40	1,67
26.01.2023	13:45	14:30	LVA-Einheit: Besprechung des Fortschritts	Projektplanung	00:45	0,75
29.01.2023	14:00	14:45	Teambesprechung für weiteres Vorgehen	Projektplanung	00:45	0,75
01.02.2023	09:50	11:15	Vorbereitung und Besprechung mit Gruppe Infrastruktur	Projektplanung	01:25	1,42
04.02.2023	11:30	14:00	Teambesprechung	Projektplanung	02:30	2,50
04.02.2023	15:00	21:45	Funktionstest (Testen einzelner Komponenten und Kommunikation)	Systemtest	06:45	6,75
05.02.2023	13:00	14:30	Teambesprechung	Projektplanung	04:05	4,08
05.02.2023	14:00	21:45	Funktionstest (Testen einzelner Komponenten und Kommunikation + Behebung der Fehler)	Systemtest	07:45	7,75
09.02.2023	16:00	17:45	Node-RED Prozess Dokumentation beginnen	Projektbericht	01:45	1,75
10.02.2023	14:00	21:30	Installationsdokumentation schreiben	Projektbericht	07:30	7,50
11.02.2023	16:00	23:15	Installationsdokumentation fertigstellen	Projektbericht	07:15	7,25
15.02.2023	13:00	18:45	Node-RED Prozess Dokumentation weiterführen	Projektbericht	05:45	5,75
16.02.2023	14:30	22:15	Node-RED Prozess Dokumentation fertigstellen	Projektbericht	07:45	7,75
18.02.2023	12:00	13:30	Reflexion (Ideen)	Projektbericht	01:30	1,50
24.02.2023	14:30	17:30	Teambesprechung + Abgabe	Projektplanung	03:00	3,00
26.02.2023	13:00	18:00	Projektabchluss (Abschlussbericht vollenden)	Projektbericht	05:00	00:00

Name:	Ozan Akkoyun		Summe:	155,37
Matrikelnummer:	12007358			
Datum	Start	Ende	Tätigkeit	Meilenstein
13.10.2022	09:00	10:00	Vorbesprechung	Projektplanung
17.10.2022	15:00	19:30	Erstellen des grundsätzlichen BPMN Modells	Projektplanung
31.10.2022	14:30	18:30	Erweitern des BPMN Modells um Ausführbarkeit	Projektplanung
02.11.2022	08:30	10:00	LVA-Einheit: Besprechung der Prozessmodelle	Projektplanung
17.11.2022	13:45	15:00	LVA-Einheit: Besprechung des Projektplans und kurze Einführung in Node-RED	Projektplanung
21.11.2022	09:00	14:15	Projektplanung durchführen	Projektplanung
24.11.2022	11:00	17:20	Projektplanung (Ziele definieren)	Projektplanung
25.11.2022	13:00	18:30	Projektplanung weiter durchführen (Definieren von Arbeitspaketen und Meilensteinen)	Projektplanung
26.11.2022	18:00	22:30	Formulierung der Einführung, Problemstellung und Schaffung von Jira	Projektplanung
28.11.2022	18:30	21:45	Projektplanung abschließen	Projektplanung
01.12.2022	13:45	16:00	LVA-Einheit: Präsentation der Projektpläne, Konfiguration des Rechners für das Projekt	Projektplanung
01.12.2022	19:15	22:05	Installation GitLab-Repo und notwendige Software in Linux	Node-RED aufsetzen
03.12.2022	12:10	18:40	Node-RED Dokumentation erarbeiten	Node-RED aufsetzen
04.12.2022	09:45	15:25	Hardware-Kompetenzen aneignen	Hardware- Prototyp
08.12.2022	16:00	19:40	MQTT in Kombination mit Projekt als auch mit anderen Gruppen ausarbeiten	MQTT Anbindung
15.12.2022	13:45	14:15	LVA-Einheit: Besprechung des Fortschritts	Node-RED aufsetzen
20.12.2022	09:00	11:15	Workshop in Mikrocontroller, Node-RED	Hardware- Prototyp
22.12.2022	11:20	18:00	Aufsetzen der notwendigen Software (Node-RED, Platformio und etc.) in Windows	Node-RED aufsetzen
23.12.2022	10:15	15:30	Testen der erhaltenen Hardware	Hardware- Prototyp
25.12.2022	13:05	17:00	Einarbeiten in Node-RED (Grundverständnisse umsetzen und vorgefertigtes Beispiel probieren)	Node-RED aufsetzen
29.12.2022	13:45	14:30	LVA-Einheit: Besprechung des Fortschritts	Hardware- Prototyp
30.12.2022	14:40	17:30	Notwendige Hardware prototypisch verkabeln	Hardware- Prototyp
01.01.2023	12:35	16:30	Funktionalität in Node-RED umsetzen	Node-RED UI
12.01.2023	13:45	14:00	LVA-Einheit: Besprechung des Fortschritts	finale Hardware
13.01.2023	12:15	13:30	Individuelles Meeting mit Auftraggeber zum Abstimmen des weiteres Vorgehens	Node-RED aufsetzen
13.01.2023	15:15	16:55	Teambesprechung für weitere Vorgehen	finale Hardware
13.01.2023	19:00	23:59	Erweiterung Hardware um smart-room Funktionalität	finale Hardware
14.01.2023	00:00	02:03	Erweiterung Hardware um smart-room Funktionalität	finale Hardware
14.01.2023	19:05	22:30	Node-RED Funktionalität für smart-room implementieren	Node-RED UI
26.01.2023	13:45	14:30	LVA-Einheit: Besprechung des Fortschritts	finale Hardware
29.01.2023	14:00	14:45	Teambesprechung für weiteres Vorgehen	finale Hardware
29.01.2023	19:05	23:10	Camunda Showcase Prozess beginnen und Dokumentation PlatformIO	MQTT Anbindung
01.02.2023	09:50	11:15	Vorbereitung und Besprechung mit Gruppe B	MQTT Anbindung
04.02.2023	11:30	14:00	Teambesprechung	Projektbericht
04.02.2023	14:15	20:25	Camunda Showcase Prozess bearbeiten	Projektbericht
05.02.2023	13:00	14:30	Teambesprechung	Projektbericht
07.02.2023	10:00	16:00	Testen Teilsysteme mit Camunda Showcase Gruppe B	Systemtest
08.02.2023	11:00	16:15	Testen Teilsysteme mit Camunda Showcase Gruppe B	Systemtest
08.02.2023	17:00	21:15	Camunda Showcase Prozess bearbeiten	Projektbericht
09.02.2023	13:45	15:25	Abgabe Hardware & Camunda Showase Prozess bearbeiten	Projektbericht
14.02.2023	12:20	17:35	Dokumentation der Projektergebnisse	Projektbericht
18.02.2023	12:15	19:05	Dokumentation der Projektergebnisse	Projektbericht
24.02.2023	14:30	17:20	Teambesprechung - Abgabe	Projektbericht
25.02.2023	15:20	18:45	Verfassung der Reflexion des Projekts	Projektbericht
26.02.2023	13:00	18:00	Teambesprechung - Fertigstellen des Abschlussberichts	Projektbericht

Name:	Betim Sulejmani		Summe:	148,00
Matrikelnummer:	11920789			
Datum	Start	Ende	Tätigkeit	Meilenstein
13.10.2022	09:00	10:00	Vorbesprechung	Projektplanung
17.10.2022	15:00	19:30	Erstellen des grundsätzlichen BPMN Modells	Projektplanung
31.10.2022	14:30	18:30	Erweitern des BPMN Modells um Ausführbarkeit	Projektplanung
02.11.2022	08:30	10:00	LVA-Einheit: Besprechung der Prozessmodelle	Projektplanung
17.11.2022	13:45	15:00	LVA-Einheit: Besprechung des Projektplans und kurze Einführung in Node-RED	Projektplanung
21.11.2022	09:00	14:15	Projektplanung durchführen	Projektplanung
24.11.2022	11:00	17:20	Projektplanung (Ziele definieren)	Projektplanung
25.11.2022	13:00	18:30	Projektplanung weiter durchführen (Definieren von Arbeitspaketen und Meilensteinen)	Projektplanung
26.11.2022	18:00	22:30	Formulierung der Einführung, Problemstellung und Schaffung von Jira	Projektplanung
28.11.2022	18:30	21:45	Projektplanung abschließen	Projektplanung
01.12.2022	13:45	16:00	LVA-Einheit: Präsentation der Projektpläne, Konfiguration des Rechners für das Projekt	Projektplanung
04.12.2022	16:30	20:45	Node-RED Dokumentation erarbeiten	Node-RED aufsetzen
05.12.2022	18:00	20:00	Hardware-Kompetenz aneignen	Node-RED aufsetzen
08.12.2022	14:00	17:30	MQTT -Dokumentation erarbeiten	Node-RED aufsetzen
15.12.2022	13:45	14:15	LVA-Einheit: Besprechung des Fortschritts	Node-RED aufsetzen
20.12.2022	09:00	11:15	Workshop in Mikrocontroller, Node-RED	Node-RED aufsetzen
24.12.2022	12:30	19:00	Aufsetzen der notwendigen Software (Node-RED, Platformio und etc.) in Windows	Node-RED aufsetzen
26.12.2022	16:15	21:00	Einarbeitung in Node-RED	Node-RED aufsetzen
29.12.2022	13:45	14:30	LVA-Einheit: Besprechung des Fortschritts	Node-RED aufsetzen
04.01.2023	14:00	19:30	Node-RED Anbindung	Node-RED aufsetzen
06.01.2023	17:15	21:30	Node-RED Anbindung	Node-RED aufsetzen
08.01.2023	17:30	19:30	MQTT-Anbindung	MQTT Anbindung
10.01.2023	12:15	16:30	Datenbank recherche	Projektbericht
11.01.2023	17:15	21:30	Datenbank recherche	Projektbericht
12.01.2023	13:45	14:00	LVA-Einheit: Besprechung des Fortschritts	Projektplanung
13.01.2023	15:15	16:55	Teambesprechung für weitere Vorgehen	Projektplanung
26.01.2023	13:45	14:30	LVA-Einheit: Besprechung des Fortschritts	Projektplanung
29.01.2023	14:00	14:45	Teambesprechung für weiteres Vorgehen	Projektplanung
01.02.2023	09:50	11:15	Vorbereitung und Besprechung mit Gruppe Infrastruktur	Projektplanung
04.02.2023	11:30	14:00	Teambesprechung	Projektplanung
04.05.2023	15:00	21:45	Funktionstest (Testen einzelner Komponenten und Kommunikation)	Systemtest
05.02.2023	13:00	14:30	Teambesprechung	Projektplanung
05.02.2023	14:00	21:45	Funktionstest (Testen einzelner Komponenten und Kommunikation + Behebung der Fehler)	Systemtest
09.02.2023	16:00	17:45	Node-RED Prozess dokumentieren	Projektbericht
10.02.2023	14:00	19:00	Node-RED Prozess Dokumentation weiterführen	Projektbericht
10.02.2023	14:00	21:30	Installationsdokumentation schreiben	Projektbericht
11.02.2023	16:00	23:15	Installationsdokumentation fertigstellen	Projektbericht
12.01.2023	17:30	21:00	Reflexion	Projektbericht
16.02.2023	14:30	22:15	Node-RED Prozess Dokumentation fertigstellen	Projektbericht
24.02.2023	14:30	17:20	Teambesprechung - Abgabe	Projektbericht
26.02.2023	13:00	18:00	fertigstellung Abschlussbericht	Projektbericht

Name: Andrea Wieser
Matrikelnummer: 12005223

Summe: 153,00

Datum	Start	Ende	Tätigkeit	Meilenstein	Zeit	Stunden
13.10.2022	09:00	10:00	Vorbesprechung	Projektplanung	01:00	1,00
13.10.2022	10:00	11:00	Gruppenbesprechung	Projektplanung	01:00	1,00
14.10.2022	08:00	10:00	Einlesen in Szenario	Projektplanung	02:00	2,00
15.10.2022	15:00	18:00	Gruppenbesprechung	Projektplanung	03:00	3,00
21.10.2022	12:00	18:00	Erstellung Modell	Projektplanung	06:00	6,00
22.10.2022	12:00	14:00	Gruppenbesprechung	Projektplanung	02:00	2,00
22.10.2022	14:00	17:00	Überarbeitung Modell	Projektplanung	03:00	3,00
29.10.2022	14:00	15:00	Gruppenbesprechung	Projektplanung	01:00	1,00
01.11.2022	16:00	18:00	Vorbereitung Präsentation	Projektplanung	02:00	2,00
02.11.2022	08:30	10:00	Präsentation Modell	Projektplanung	01:30	1,50
14.11.2022	16:00	20:00	Gruppenbesprechung	Projektplanung	04:00	4,00
16.11.2022	10:00	12:00	Ideen sammeln für Funktionalitäten Smart Room	Projektplanung	02:00	2,00
17.11.2022	13:45	15:15	Abstimmungstreffen	Projektplanung	01:30	1,50
17.11.2022	15:45	18:00	Gedanken zur Umsetzung des Smart Room	Projektplanung	02:15	2,25
20.11.2022	16:00	17:00	Gruppenbesprechung	Projektplanung	01:00	1,00
21.11.2022	14:00	18:00	Erstellung Projektplan Gruppe D	Projektplanung	04:00	4,00
22.11.2022	15:00	18:00	Gruppenbesprechung	Projektplanung	03:00	3,00
22.11.2022	18:00	21:00	Projektplan Gruppe D	Projektplanung	03:00	3,00
26.11.2022	Wechsel von Gruppe D zu Gruppe C					
27.11.2022	14:00	19:00	Projektplan Gruppe C	Projektplanung	05:00	5,00
28.11.2022	17:30	20:30	Gruppenbesprechung	Projektplanung	03:00	3,00
28.11.2022	18:30	21:30	Bearbeitung & Formatierung Projektplan Gruppe C	Projektplanung	03:00	3,00
01.12.2022	13:45	15:15	Abstimmungstreffen	Node-RED aufsetzen	01:30	1,50
03.12.2022	15:00	20:00	Recherche Node RED	Node-RED aufsetzen	05:00	5,00
13.12.2022	17:00	20:00	Vorbereitungen Node RED	Node-RED UI	03:00	3,00
15.12.2022	13:45	15:15	Abstimmungstreffen	Node-RED UI	01:30	1,50
18.12.2022	13:00	15:00	GitLab Projekt installieren	Node-RED UI	02:00	2,00
20.12.2022	08:00	10:00	MQTT Besprechung mit Techniker	Hardware- Prototyp	02:00	2,00
21.12.2022	14:00	17:00	Gruppenbesprechung	MQTT Anbindung	03:00	3,00
23.12.2022	12:00	14:00	Vorgehen dokumentieren	MQTT Anbindung	02:00	2,00
27.12.2022	13:45	14:15	Abstimmungstreffen	MQTT Anbindung	00:30	0,50
29.12.2022	13:00	15:00	GitLab Dokumentationen lesen	MQTT Anbindung	02:00	2,00
12.01.2023	13:45	15:15	Abstimmungstreffen	MQTT Anbindung	01:30	1,50
13.01.2023	13:00	17:00	Gruppenbesprechung	MQTT Anbindung	04:00	4,00
16.01.2023	12:00	18:00	Abschlussbericht	Projektbericht	06:00	6,00
20.01.2023	18:00	22:00	GitHub Gruppe B	MQTT Anbindung	04:00	4,00
26.01.2023	08:00	12:00	Tests	Systemtest	04:00	4,00
29.01.2023	14:00	17:00	Gruppenbesprechung	finale Hardware	03:00	3,00
30.01.2023	11:00	17:00	Beschreibung Durchführung	Projektbericht	06:00	6,00
31.01.2023	14:00	17:00	Beschreibung Teststrategie	Systemtest	03:00	3,00
31.01.2023	17:30	21:00	Reflexion	Projektbericht	03:30	3,50
01.02.2023	16:00	20:00	Korrekturlesen	Projektbericht	04:00	4,00
02.02.2023	16:00	17:00	Formatierung Abschlussbericht	Projektbericht	01:00	1,00
03.02.2023	15:00	17:00	Gruppenbesprechung	Projektbericht	02:00	2,00
04.02.2023	13:00	18:00	Überarbeitung Abschlussbericht	Projektbericht	05:00	5,00
05.02.2023	15:00	17:00	Gruppenbesprechung	Projektbericht	02:00	2,00
10.02.2023	15:00	20:00	Überarbeitung Durchführung	Projektbericht	05:00	5,00
15.02.2023	16:00	17:00	Überarbeitung Reflexion	Projektbericht	01:00	1,00
20.02.2023	12:00	17:45	Überarbeitung Abschlussbericht	Projektbericht	05:45	5,75
24.02.2023	14:30	18:30	Gruppenbesprechung	Projektbericht	04:00	4,00
25.02.2023	10:00	15:30	Überarbeitung Tests	Systemtest	05:30	5,50
26.02.2023	13:00	17:00	Gruppenbesprechung	Projektbericht	04:00	4,00
26.02.2023	18:00	20:00	Korrekturlesen	Projektbericht	02:00	2,00