Carl von Ossietzky
Universität
Oldenburg

Soft Skills

PROJEKTDOKUMENTATION SoSe 2022

SMART MAILBOX

Universelles Posteingangserkennungsgerät

Abgabetermin: 20.07.2022

Gruppenteilnehmer (Gruppe 15)
Martin Diedrich (Node-RED Dashboard),
Nikolas Kaminsky (Gehäuse),
Nicholas Schaper (Hardware),
Sebastian Vittinghoff (Software)

Inhaltsverzeichnis

Inhalt

Einleitung

Die folgende Projektdokumentation erläutert den Aufbau des Projekts, das im Rahmen des Moduls Soft Skills im Sommersemester 2022 durchgeführt wurde.

Projektbeschreibung

Die grundlegende Idee des Projekts ist ein Gerät, dass es erlaubt von überall und zu jedem Zeitpunkt einzusehen, ob man Post erhalten hat. Dazu wird sich überlegt welche Hardware und welche Sensorik für das Projekt zum Einsatz kommt und wie diese miteinander verbaut werden. Außerdem sollte das Gerät für eventuelle Reparaturen leicht aus dem Briefkasten zu entnehmen sein. Zudem sollte die Lösung universell für die meisten Briefkästen funktionieren und sehr simpel zu bedienen sowie einzubauen sein (plug - and -play Prinzip).

Sobald Post in den Briefkasten geworfen wird, kann der Benutzer sich dies auf einer Webapp anzeigen lassen und den Posteingang bestätigen.

Planung der Hardwarekomponenten

Zunächst wird sich überlegt, welche Sensoren sinnvoll für das Registrieren des Einwurfs des Briefes sind. Dort kamen Lichtsensoren und Ultraschallsensoren (Abstandsmesser) in Frage. Der Lichtsensor weist allerdings das Problem auf, dass er jedes Mal anschlagen würde, wenn die Klappe des Briefkastens geöffnet würde. Das würde wiederum unnötige Meldungen an den Nutzer bedeuten, da beim bloßen reinschauen in den Briefkasten der Sensor ausgelöst werden würde.
Dadurch bleibt der Ultraschallsensor als Werkzeug zur Ermittlung, ob Post eingeworfen wurde.

Die Entscheidung für den Mikrocontroller ergibt sich aus der bisherigen Erfahrung der Projektteilnehmenden mit dem ESPP8266 D1 Mini. Dieser wurde bereits in einem anderen Projekt des gleichen Moduls genutzt und erscheint daher günstig.
Außerdem erfüllt er die Minimalanforderungen, die durch das Projekt entstehen. Er verfügt über einen WLAN-Chip, genügend Anschlüsse, um den Ultraschallsensor und weitere Hardwarekomponenten zu verbinden, er stand kostenlos für dieses Projekt zur Verfügung und er ist physisch nicht so groß, sodass er gut in den Briefkasten und in Gehäuse passt.

Zur Statusanzeige des Geräts werden zwei LEDs eingesetzt: eine grüne LED und eine rote LED. Die rote LED zeigt an, dass das Gerät eingeschaltet aber noch nicht bereit ist und die grüne LED gibt an, ob das Gerät einwandfrei funktioniert und aktiv ist.

Außerdem einen Schalter, um das Gerät schnell und einfach ein – und ausschalten zu können. Um das Gerät einfach installieren zu können, wird eine Batterie benutzt, damit nicht zusätzlich zum Einbau ein Kabel in den Briefkasten gelegt werden muss.

Zuletzt wird ein Gehäuse benötigt, das die Komponenten enthält und sich leicht in den Briefkasten integrieren lässt.

Für nähere Erläuterungen zu dem Gehäuse wird hier auf das dafür vorgesehene Kapitel in der Dokumentation verwiesen.

Planung der Softwarekomponenten

Softwareseitig wird für das Projekt ein Programm benötigt, dass über folgende Fähigkeiten verfügt: Auslesung des Inputs des Ultraschallsensors und interpretieren des Wertes, eine Statusanzeige zur Informierung des Benutzers über die einwandfreie Funktionsweise des Geräts und eine Funktion zur Kalibrierung des Sensors.

Für die leichtere Bedienung des Programms wird eine graphische Benutzeroberfläche benötigt, die durch Node-Red realisiert wird.

Hardware

Die Hardware für die Smarte Mailbox ist im Folgenden in die Komponentenblöcke LEDs, Spannungsversorgung, Ultraschallsensor und Mikrocontroller unterteilt. In der Abbildung 5 sieht man eine Liste mit allen Komponenten der Hardware.

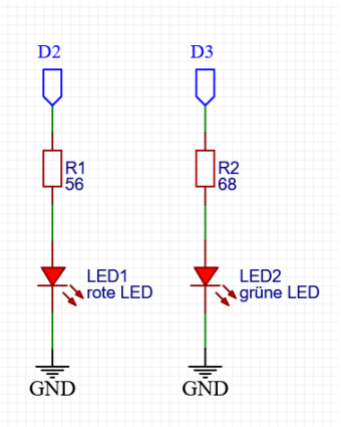

Abbildung 1: LEDs

In Abbildung 1 ist der Schaltplan der LEDs mit den entsprechenden Vorwiderständen R1 und R2 dargestellt. Die LEDs dienen als Statusanzeige, wobei die rote LED symbolisiert, dass die Smarte Mailbox angeschaltet jedoch noch nicht initialisiert ist. Sobald die Mailbox initialisiert ist, wird die rote LED ausgeschaltet und die grüne LED angeschaltet.

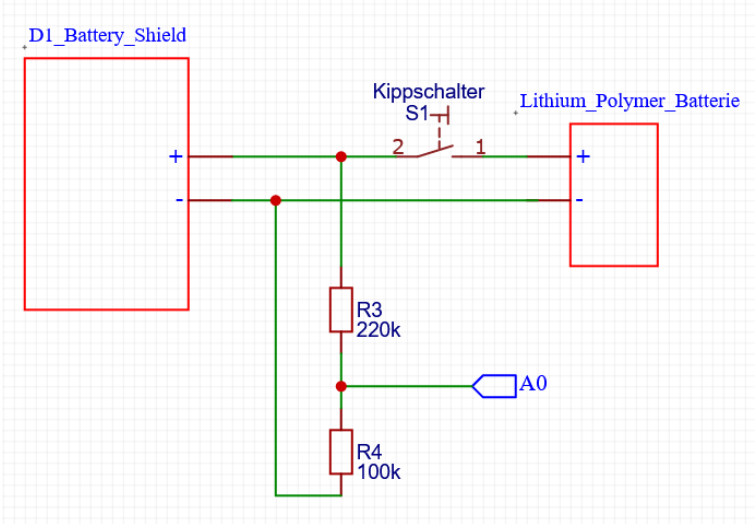

Abbildung 2: Spannungsversorgung

Die Abbildung 2 zeigt einen Spannungsteiler aus den Widerständen R3 und R4. Der Spannungsteiler ist an der Eingangsspannung des Battery Shieldes und dem Pin A0 des Mikrocontrollers angeschlossen. Auf diese Weise lässt sich an dem Pin A0 die Spannung der Lithium Polymer Batterie überprüfen. Die maximale Eingangsspannung des Pins A0 beträgt 3,3V. Da die Eingangsspannung der Batterie einen Maximalwert von 4,2V besitzt, wird der Spannungsteiler benötigt, damit es zu keinen Schäden an dem Mikrocontroller kommt.
Mit dem Kippschalter S1 lässt sich die Batterie von dem Battery Shield trennen, wodurch der Mikrocontroller an und aus geschaltet werden kann.
Die Lithium Polymer Batterie lässt sich über einen Micro USB-Anschluss auf dem Battery Shield aufladen.

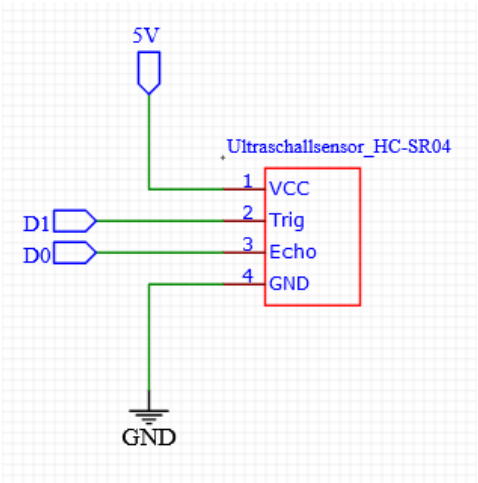

Abbildung 3: Ultraschallsensor HC-SR04

Der Ultraschallsensor aus der Abbildung 3 ist unter dem Einwurfsschlitz des Briefkastens angebracht und misst den Abstand zu der Briefkastenwand. Wenn zum Beispiel ein Brief in den Briefkasten geworfen wird, so verändert sich der gemessene Wert. Damit wird erkannt, dass Post eingegangen ist.

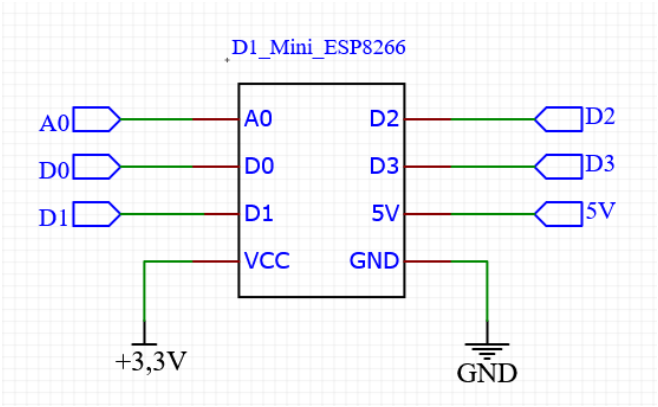

Abbildung 4: Microcontroller

In der Abbildung 4 ist die Belegung an dem Mikrocontroller D1 Mini ESP8266 der im Voraus aufgeführten und beschriebenen Komponenten der Smarten Mailbox dargestellt. Auf dem Mikrocontroller ist das Battery Shield aufgesteckt, welches den Mikrocontroller mit Strom versorgt.

Bauteil:	**Beschreibung:**
D1 Mini ESP8266	Mikrocontroller
D1 Battery Shield	Battery Shield
Debo Sen Ultra HC-SR04	Ultraschallsensor
EEMB 103454	Lithium Polymere Batterie
R1: Widerstand 56Ohm	Vorwiderstand rote LED
R2: Widerstand 68Ohm	Vorwiderstand grüne LED
LED1: 5-16000 RT	Rote LED
LED2: 5mm 5V Ge	Grüne LED
R3: Widerstand 220kOhm	Spannungsteiler Widerstand
R4: Widerstand 100kOhm	Spannungsteiler Widerstand
Bolatus Kippschalter	Kippschalter

Abbildung 5: Komponentenliste

Gehäuse

Für die Hardware des Projekts (Batterie, Mikrocontroller, Ultraschallsensor) wird ein Gehäuse benötigt, das groß genug ist, um die Materialien aufzubewahren. Es soll außerdem leicht in den Briefkasten ein und ausbaubar sein. Zusätzlich dazu soll auch die Hardware leicht aus dem Gehäuse entnehmbar und wieder einbaubar sein zwecks eventueller Reparaturen.

Zunächst wird sich ein System überlegt, bei dem eine Schiene in den Briefkasten eingesetzt wird auf die man das Gehäuse leicht stecken kann. So werden keine Schrauben oder ähnliches zum Ein-und Ausbau benötigt. Außerdem muss das Gehäuse Öffnungen für den Ultraschallsensor aufweisen, damit dieser korrekt die Entfernung misst und erkennt, ob ein Brief eingeworfen wird oder nicht.

Die Idee des ersten Designs ist auf folgender Grafik zu sehen:

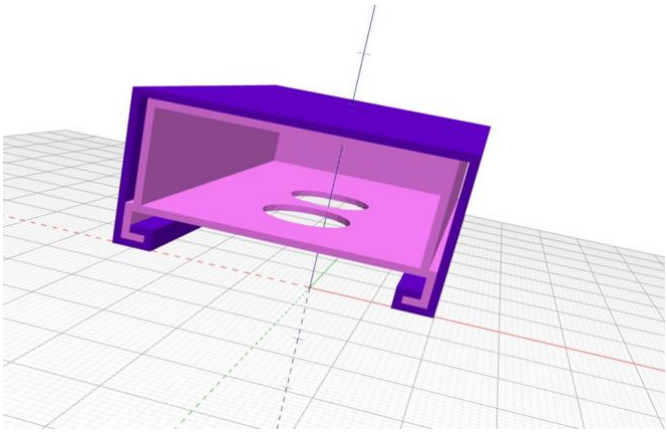

Abbildung 6: Gehäuse Version 1, Vorderseite

Der blaue Körper repräsentiert die in dem Gehäuse anzubringende Schienenvorrichtung und der magentafarbene Körper repräsentiert das Gehäuse selbst. Durch die angewinkelte Schiene soll erreicht werden, dass Gehäuse fest in der Schiene sitzt. In der nachfolgenden Grafik ist die Rückseite der Konstruktion zu sehen.

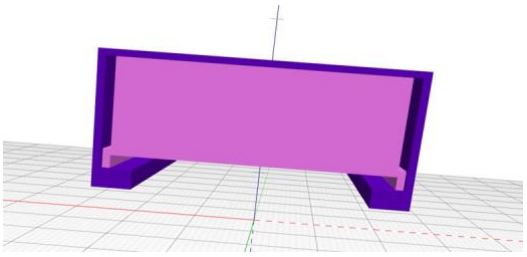

Abbildung 7: Gehäuse Version 1, Rückseite

Zu beachten ist hier, dass die Schienenvorrichtung eine Blockade aufweist, damit das Gehäuse festgehalten wird und nur in einer Richtung herausnehmbar ist. Die Rückseite auf der letzten Grafik bildet letztendlich die Unterseite des Gehäuses ab, die Seite mit den zwei Öffnungen die rechte Seite, die in den Briefkasten zeigt und die Oberseite ist die Kontaktseite die an dem Briefkasten befestigt wird.

Dies war die erste Idee und es wurde gleich ein Problem entdeckt, das schlussendlich zu einem anderen Design geführt hat.

Das Problem ist, dass sich das Gehäuse durch die Ausrichtung der Schienenvorrichtung nur von oben einbauen ließe. Als Folge müsste die Schienenvorrichtung sehr tief im Briefkasten eingebaut werden, um weiterhin ein leichtes Ein - und Ausbauen zu gewährleisten. Dies würde wiederum bedeuten, dass der Ultraschallsensor tiefer sitzt und so eventuell schon eingeworfene Briefe als neu eingeworfen interpretiert und dem Benutzer eine falsche Meldung für neu angekommene Post übermittelt.

Somit muss das Gehäuse eine Möglichkeit bieten, weiter oben im Briefkasten zu sitzen und gleichzeitig leicht ein – und ausbaubar sein.

Es wird eine neue Idee entwickelt, mit der das Gehäuse nicht von oben sondern seitlich eingebaut werden kannte, sodass praktisch keine Limitierung mehr für die Höhe des Gehäuses im Briefkasten vorlag.

Grob gesehen ist diese Idee die Gleiche, lediglich das Schienensystem hat sich verändert.

In der nachfolgenden Grafik ist die neue Schienenvorrichtung zu sehen:

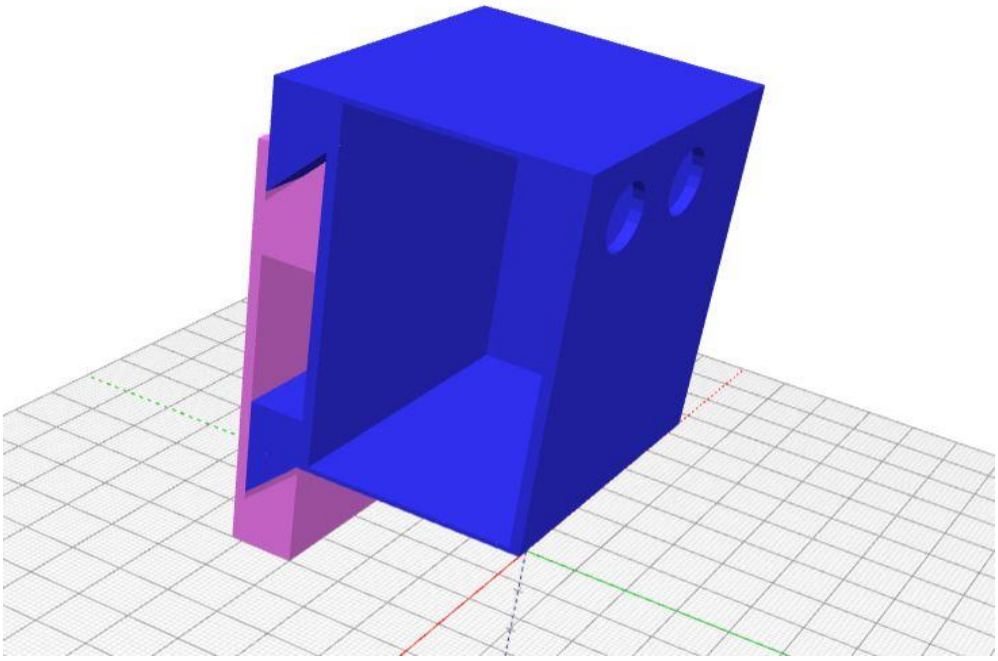

Abbildung 8: Gehäuse Version 2, Vorderseite

Die magentafarbene Schiene wird am linken Rand des Briefkastens angebracht und das Gehäuse kann leicht von der Seite oder auch von vorne auf die Schienenvorrichtung gesetzt werden. Der Ultraschallsensor zeigt immer noch in die korrekte Richtung im Briefkasten. Die Öffnungen für den Ultraschallsensor sind oben angebracht, um zu verhindern, dass bereits eingeworfene Briefe fälschlicherweise als neu eingeworfene erkannt werden.

Doch auch diese Idee bedarf einer Überarbeitung auf Grund folgender gefundener Probleme: Die hier zu erkennbaren für den Ultraschallsensor sind zu klein und müssen vergrößert werden. Außerdem ist das Gehäuse zu leicht nach links und rechts entnehmbar und könnte aus der Schiene

herausfallen. Daher werden links und rechts Blockaden benötigt, sodass das Gehäuse nur von einer Seite eingebaut werden kann und fester sitzt. Zusätzlich sind in dem Hardwareumfang noch ein an und Ausschalter, zwei Status-LEDs und ein Lüfter vorhanden.

Für den Schalter und die LEDs müssen weitere Öffnungen angebracht werden und ein paar Öffnungen für bessere Luftzufuhr.

Mit all diesen Optimierungen sieht das Gehäuse wie folgt aus:

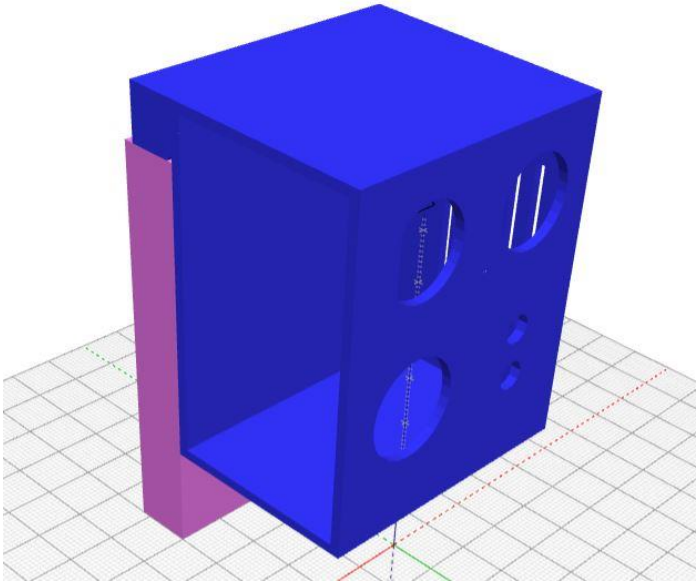

Abbildung 9: Gehäuse Version 3, Vorderseite

Die oberen beiden Öffnungen wurden nun vergrößert und auf den Ultraschallsensor angepasst. Außerdem sieht man für die weiteren Teile auch die entsprechenden.

Auf dieser Abbildung sind die Lüfterschlitze noch einmal deutlich besser zu erkennen.

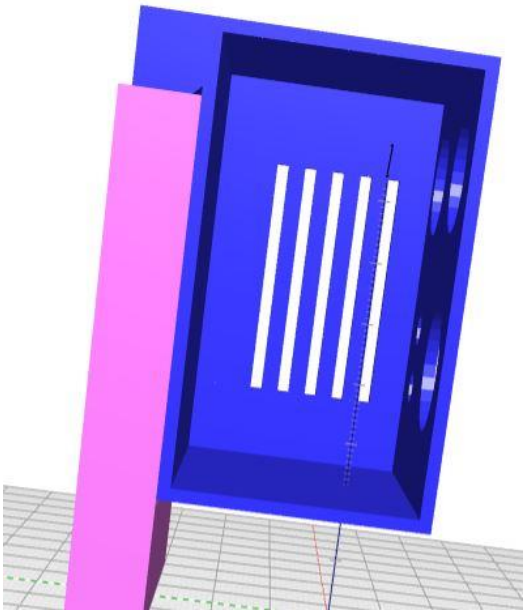

Abbildung 10: Gehäuse Version 3, Luftschlitze

Die auf der letzten Seite dem Betrachter zugewandte Seite ist offen damit die Hardware leichter eingebaut werden kann. Allerdings sollte es möglich sein, das Gehäuse vollständig zu schließen. Zudem sollte die schließende Seite ebenfalls Schlitze zur besseren Belüftung. So liegen sich zwei Seiten mit Öffnungen direkt gegenüber, was einen guten Airflow für die Hardwarekomponenten ermöglicht.

Diese Seite wurde wie folgt modelliert:

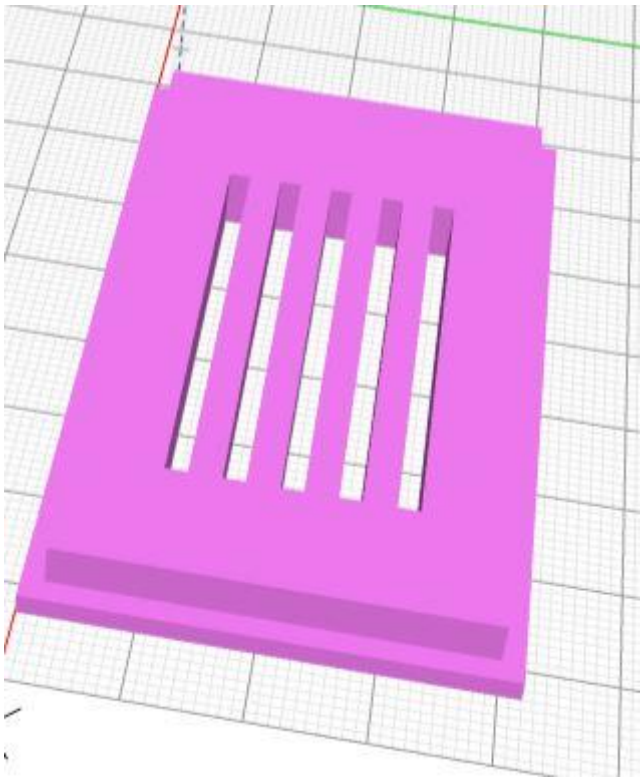

Abbildung 11: Gehäuse Version 3, Verschluss

Diese Tür wird einfach in die offene Seite hineingesteckt. Das innere Rechtecht hat exakt die gleichen Dimensionen wie die Öffnung am Gehäuse und kann so mit leicht ein – und ausgebaut werden.

Node-RED

Für das Projekt, insbesondere der Software, braucht man noch eine Schnittstelle, mit der man den Mikrocontroller mit einer graphischen Oberfläche kommunizieren lässt, um sich so anzeigen zu lassen, ob man Post bekommen hat und um den Sensor richtig zu kalibrieren. Für dieses Problem wird sich für Node-Red entschieden, mithilfe dessen man diese Oberfläche entwerfen kann.

Zum Aufbau der graphischen Oberfläche wird das Node-Red Dashboard verwendet. Nun wird der erste Aufbau entworfen und designed. Das Ziel ist es, eine Oberfläche zu schaffen, auf der man sieht, ob man Post bekommen hat, eine Statusanzeige, die einem den Status des Mikrocontrollers anzeigt, ob dieser Funktionstüchtig ist oder nicht und eine Möglichkeit bietet, den Sensor auf die richtige Entfernung zu kalibrieren. Um diese Statusanzeige zu realisieren, benutzen wir ein LED Add-on.

Der erste Entwurf des Dashboards sieht nun wie in der nächsten Abbildung zu sehen aus.

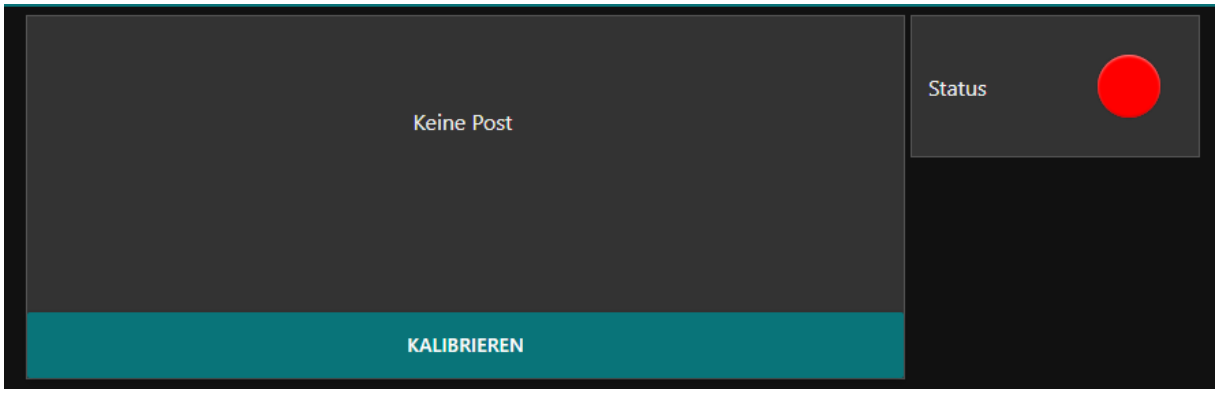

Abbildung 12: Dashboard Version 1, Initialer Zustand

Hier kann man nun alle Funktionen sehen, die man benötigt werden. Allerdings ist nicht von Vorteil, wenn man die Anzeige für die Post schon sehen kann, ohne dass der Mikrocontroller überhaupt funktioniert. Aus diesem Grund wird sich dafür entschieden, eine „Startseite“ zu machen, die angezeigt wird, solange der Controller noch nicht bereit ist.

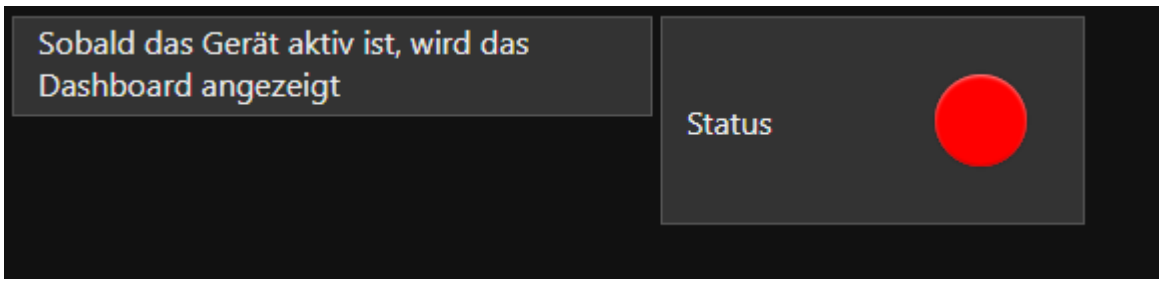

Abbildung 13: Dashboard Version 1, Gerät offline

Sobald das Gerät aktiv ist, wird man auf die Seite, die man in Abb. 1 sieht weitergeleitet und bekommt eine Meldung, wenn Post da ist, einersetis eine ganz normale Meldung und andererseits ein Pop-Up.

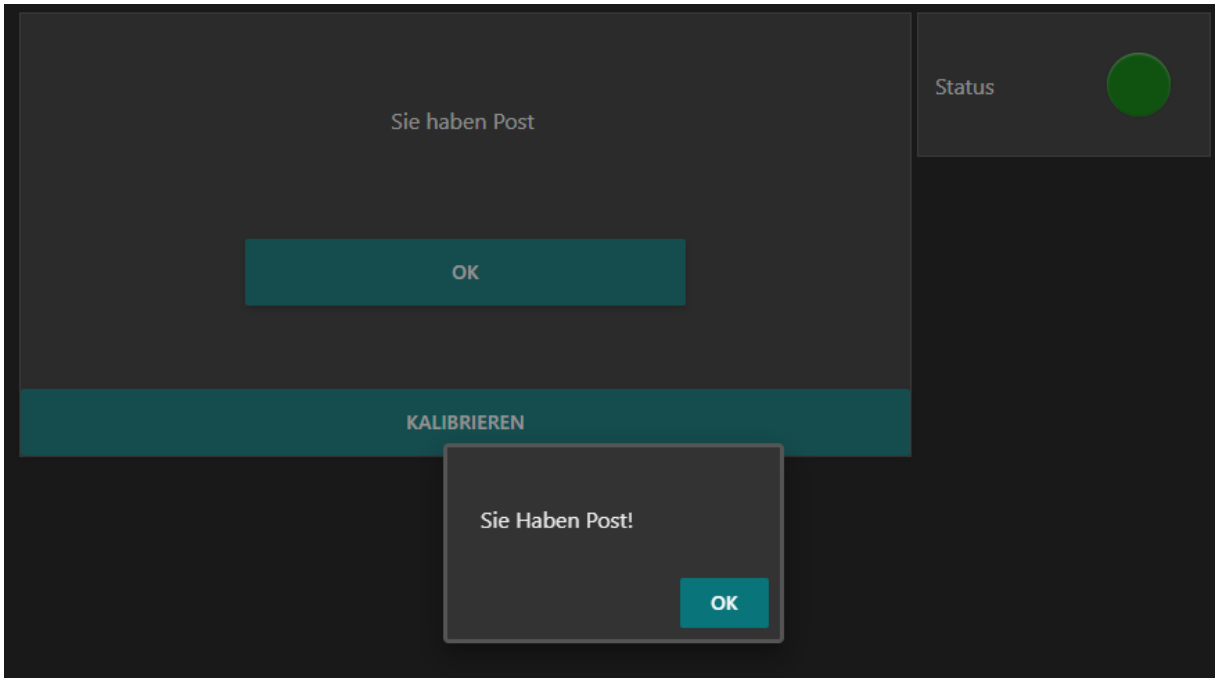

Abbildung 14: Dashbaord Version 1, Posteingang

Allerdings geht es jetzt nur um die graphische Oberfläche. Demnach wird es ebenfalls nochmal um den tatsächlichen Aufbau in Node-Red gehen. Die Oberfläche in Abb. 2 wird wie in der nächsten Abb. gezeigt wird, umgesetzt.

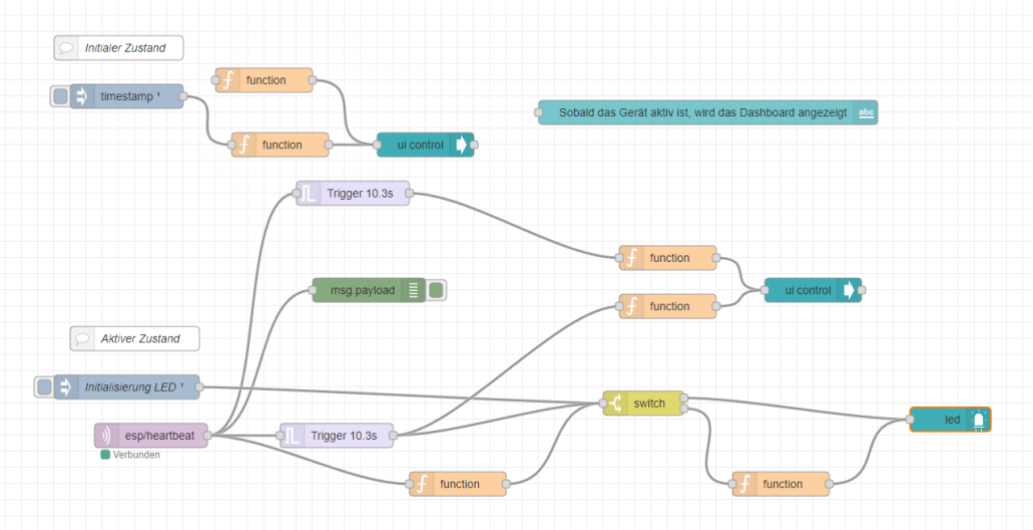

Abbildung 15: Flow, Initialer Zustand

Die Schaltung, zwischen den Oberflächen wird mithilfe des UI-Control-Blocks umgesetzt, der mit Hilfe einer JSON Datei (Abb. 5), zwischen den Definierten Gruppen umschaltet. Des Weiteren, musste die LED erstmalig initialisiert werden, wie man in dem unteren Abschnitt in Abb. 4 sieht. Um zwischen dem Mikrocontroller und Node-Red zu kommunizieren, wird der Broker MQTT benutzt, der in Abb. 4 einen Heartbeat sendet, um zu signalisieren, dass dieser noch funktioniert. Dementsprechend wird dann ebenfalls zwischen den Gruppen für die Oberfläche gewechselt. Entweder wenn der Mikrocontroller online ist, wie in Abb. 1 zusehen, oder wie in Abb. 2, wenn dieser offline ist.

```
msg.payload = {
    "group": {
        "hide": [
            "SmartPostbox_Standard",
            "SmartPostbox_PostDa"
        ],
        "show": [
            "SmartPostbox_Group3",
            "SmartPostbox_LED anzeige"
        ]
    }
}

return msg;
```

Abbildung 16: Funktionsweise Gruppenwechsel

Mit Hilfe dieses Payloads, in Form eines JSON Objektes, kann man bestimmte Gruppen ausblenden (hide) oder einblenden (show). Diese Funktionsweise, wird sich zu Nutze gemacht, um die verschiedenen Zustände darstellen zu können.

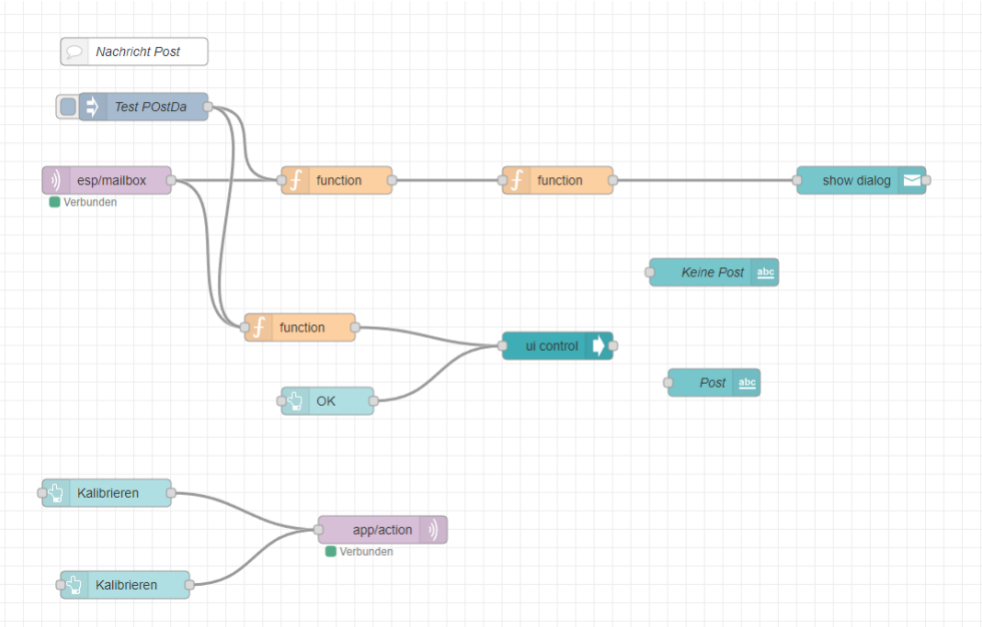

Abbildung 17: Flow, Posteingang

In Abb. 6 ist das Flow-Chart zu sehen, das die Funktionalität für das Erhalten der Post abbildet. Dieses funktioniert ebenfalls über den Broker MQTT, über den eine Nachricht gesendet wird, falls Post eintrifft. Die Funktion, die in der Grafik mit dem UI-Controller verbunden ist, sendet ein JSON-Objekt an diesen. Dieses stößt wiederum das Dashboard an, sodass dort die Nachricht erscheint, dass Post eingetroffen ist. Außerdem sind unten in der Abbildung die Kalibrierungs-Knöpfe zu erkennen, die ein JSON-Objekt an den MQTT senden.

Der eigentliche Aufbau und die Funktionsweise, sind also fertig. Beim Testen haben wir feststellen müssen, dass Fehler aufgetreten sind. Beispielsweise ist die Meldung über den Erhalt von Post nicht gespeichert und ist wieder gelöscht, sobald der Mikrocontroller offline ist oder man die Seite des Dashboards neu geladen hat. Es muss also eine andere Funktionsweise gebaut werden, die nicht über dieses Problem verfügt. Aus den bisher beschriebenen Problemen sind die entscheidenden Schlüsse gezogen worden und es konnten die logischen Optimierungen daraus abgeleitet werden.

So sieht der Flow für die gleiche Funktion, wie in Abb. 4 nun noch wie folgt aus.

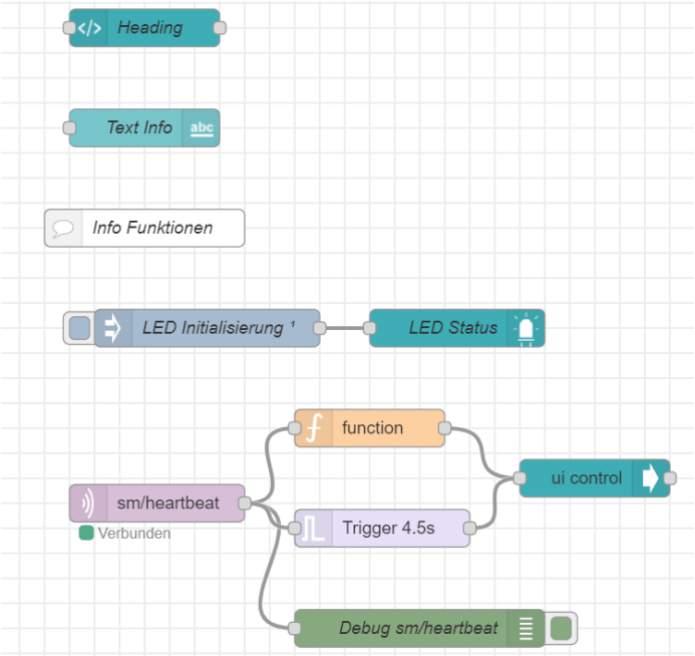

Abbildung 18: Flow Version 2, Initialer Zustand

Die Funktionsweise wird komprimiert und vereinfacht. Ebenfalls ist das Dashboard im Design angepasst, wie in Abb. 8, Abb. 9 und Abb. 10 zu sehen.

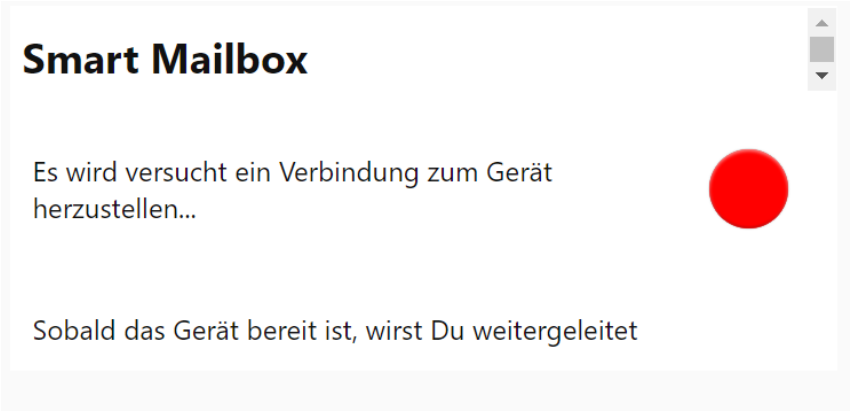

Abbildung 19: Dashbboard Version 2, gerät Offline

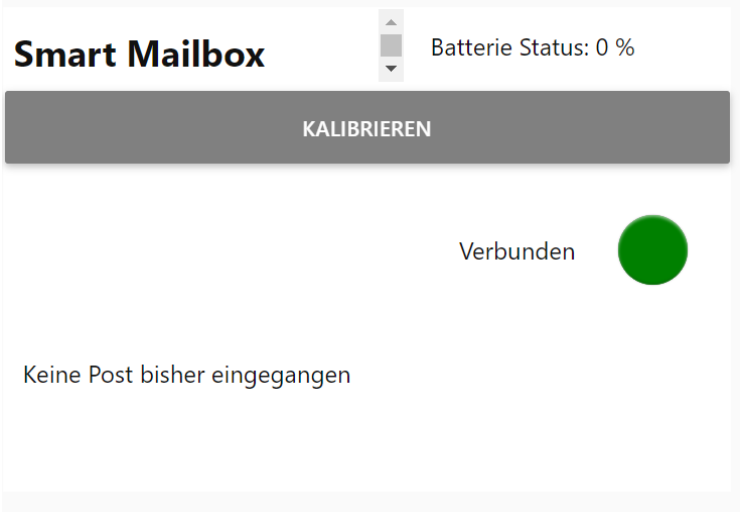

Abbildung 20: Dashboard Version 2, Initialer Zustand

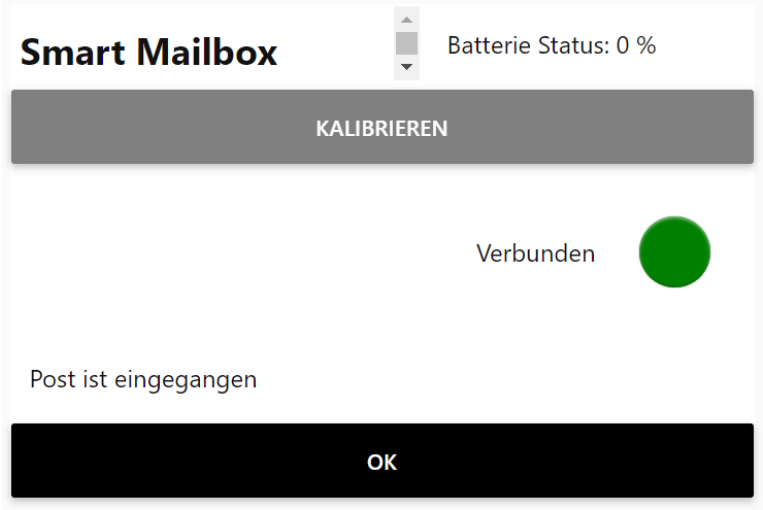

Abbildung 21: Dashbboard Version 2, Posteingang

Ebenfalls wird der Flow für die Funktion, die Auskunft über den Erhalt von Post gibt, komprimiert, aber nicht vereinfacht.

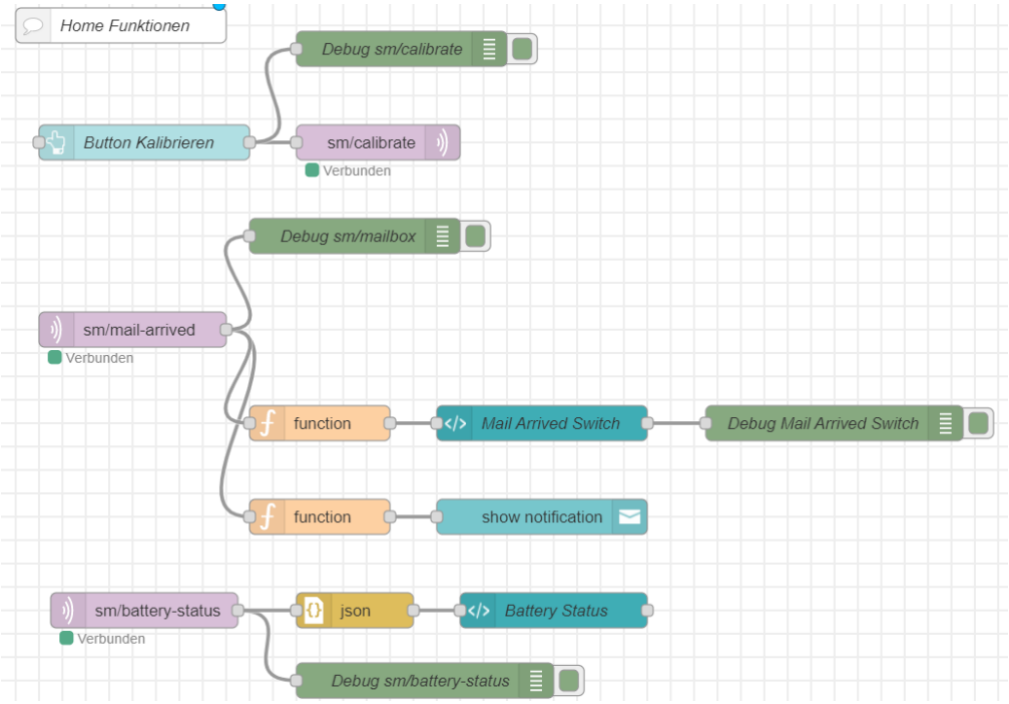

Abbildung 22: Flow Version 2, Posteingang

Eines der Probleme ist, dass die Anordnung der Gruppen im Dashboard nicht funktionieren wie vorgesehen. Die zu den Nachrichten zugehörige Gruppe wird rechts von der Statusleiste angezeigt. Allerdings ist vorgesehen, dass die Nachrichten unter der Statusleiste angezeigt werden. Um dies in Node-Red zu erreichen, müssen sowohl die Nachrichtengruppe als auch die Statusgruppe in Gruppe zusammengefasst werden. Um dies zu erreichen, wird mit AngularJS ein komplett eigener Block gebaut, der folgende Funktionen hat. Zunächst soll angezeigt werden, ob bereits Post eingetroffen ist oder nicht. Des Weiteren soll die Statusmeldung zurückgesetzt werden können, indem der OK-Knopf gedrückt wird. Als letztes soll der aktuelle Statuswert über den Eingang der Post vermerkt werden, um zu erreiche, dass der Status erhalten bleibt, wenn die Seite neu geladen wird oder der Mikrocontroller die Verbindung verliert.

Außerdem wird eine Anzeige für den Batterie Status des verbauten Akkus eingefügt, damit erkennbar ist, ob man den Akku in Gerät neu laden muss. Hierzu wird über den MQTT der aktuelle Batteriestand als JSON Objekt gesendet und schließlich im Dashboard als Nachricht angezeigt (siehe Abb. 20 und 21).

Software

Bevor mit der Entwicklung der Software gestartet werden kann, muss eine allgemeine Basis geschaffen werden. Die allgemeine Basis beinhaltet Recherchearbeiten, bspw. nach notwendigen Tools und Paketen, sowie das Aufsetzen einer für das Projekt vollständigen und plattformunabhängingen Entwicklungs- sowie Testumgebung. Während des gesamten Ablaufs sind Absprachen einzuhalten, um parallellaufende Arbeiten zu starten und / oder nicht zu blockieren. Insbesondere bei der Umsetzung der allgemeinen Basis sollte sich daher darauf fokussiert werden die gesetzten Ziele in abgesprochener Zeit einzuhalten, damit der reibungslose Ablauf gewährleistet wird.

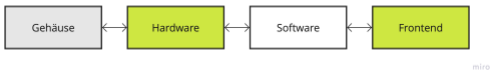

Abbildung 23: Absprachen

Recherche

Die bei der Recherche gesetzten Kriterien für Tools und Pakete sind Bekanntheitsgrad, Vollständigkeit sowie Kompositionsmöglichkeit. Mit Bekanntheitsgrad sind häufige Nutzung und implizite Beliebtheit gemeint, mit Vollständigkeit die für die Software zu erfüllenden Aufgaben und mit Komposition die einwandfreie Integration mit anderen Diensten.

Plattformunabhängikeit

Damit unabhängig vom Betriebssystem an dem Projekt gearbeitet werden kann sind die ausgewählten Tools für Entwicklungs- sowie Testumgebung plattformunabhängig.

Versionsverwaltung

Als Versionsverwaltung wird GitHub genutzt. Die Bereitstellung der Software über GitHub soll allen Projektbeteiligten einen zentralen und aktuellen Stand der Software bieten.

Alle für das Projekt erzeugten Konfigurations- und Entwicklungsdateien sowie Dokumentationen sind somit zentral abgelegt.

Repository Smart Mailbox

Repository Smart Mailbox Network

Entwicklungsumgebung

Für die Entwicklung der Software wird die CLion IDE, PlatformIO, dass PlatformIO und Docker Plugin eingesetzt. Im Gegensatz zur Arduino IDE, können die Quelldateien des Frameworks direkt in der CLion IDE eingesehen werden. Dadurch kann auf die Verwendung sogenannter Cheat Sheets verzichtet werden. PlatformIO stellt das Framework für das ESP8266 Board bereit, das PlatformIO Plugin den Projektrahmen für CLion. Mit dem Docker Plugin können Services innerhalb von CLion konfiguriert, bereitgestellt, gestartet und gestoppt werden.

Für die Testumgebung wird Docker mit den Diensten MQTT-Broker Mosquitto (im Folgenden nur noch MQTT-Broker genannt) und Node-RED eingesetzt. Docker schafft die virtuelle Umgebung, der MQTT-Broker die Kommunikationsebene zwischen Dashboard und Mikrocontroller und Node-RED die Entwicklungsplattform für das Dashboard.

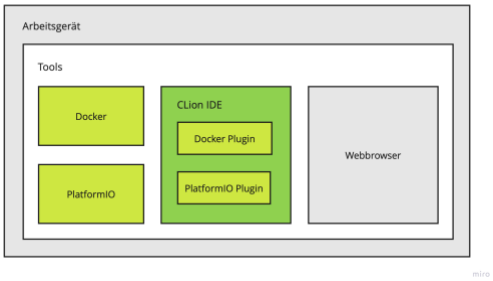

Abbildung 24: Entwicklungsumgebung

Testumgebung

Die Testumgebung wird auf einem Arbeitsgerät eingerichtet, könnte aber auch auf einem Server eingerichtet werden. Die Installation wird mit einem Kommandozeilenbefehl angestoßen, dieser führt die in der docker-compose Datei hinterlegten Anweisungen durch. Die Anweisungen beinhalten das Herunterladen und Installieren der Dienste (MQTT-Broker und Node-RED) sowie die Konfiguration und Bereitstellung. Nach Bereitstellung können die Dienste direkt verwendet werden.

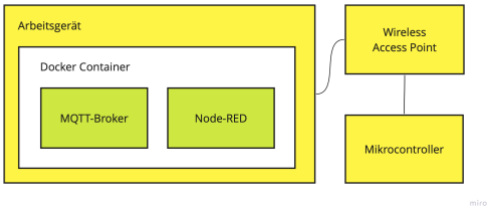

Abbildung 25: Testumgebung

Mobilität

Damit die Testumgebung im Makerspace oder auch woanders eingesetzt werden kann, wird ein weiterer Mikrocontroller zum Wireless Access Point funktionalisiert. Die dafür geschriebene Software

wird hier nicht weiter betrachtet. Die Software für den Wireless Access Point ist auch über die Versionsverwaltung bereitgestellt.

Dienste und Adressen

Die Dienste können untereinander mit den in der docker-compose Datei gesetzten Bezeichnern in der virtuellen Umgebung kommunizieren.

Anfragen an den Host müssen über die im Netzwerk gesetzte IP-Adresse und dem vom jeweiligen Dienst gesetzten Port erfolgen.

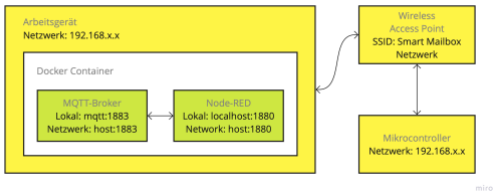

Abbildung 26: Dienste und Adressen

Pinbelegung

Die Pinbelegung erfolgt in vorheriger Absprache. Somit ist von vornherein definiert was benutzt wird und wofür. Die Definition erfolgt anschließen im Quelltext der Software. Für die Definitionen werden die Bezeichner aus der Absprache genutzt.

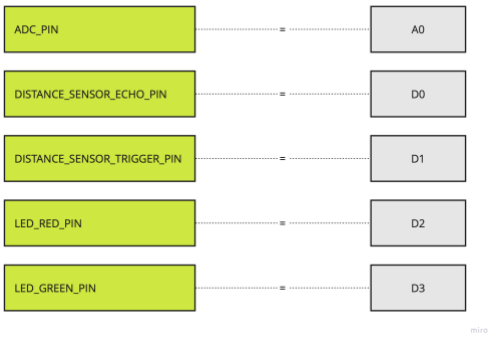

Abbildung 27: Pinbelegung

Endpunkte

Die Definition der Endpunkte erfolgt in vorheriger Absprache. Dadurch soll die Arbeit an den asynchronen Funktionalitäten des Node-RED Dashboards nicht blockiert und eine spätere Anpassung der bereits eingetragenen Endpunkte nicht erforderlich werden.

Die Endpunkte oder auch Topics werden in Subscriber und Publisher eingeteilt.

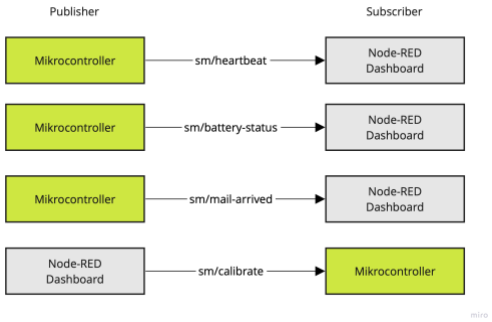

Abbildung 28: Endpunkte

Die Endpunkte tragen alle den Domänen-Präfix sm, die Abkürzung steht für Smart Mailbox.

Payload

Am Payload lässt sich schnell erkennen, dass die Endpunkte mehrheitlich als Signal vom Mikrocontroller genutzt und nur an einer einzigen Stelle zur Übermittlung von Daten eingesetzt werden. Auch wenn für das Signal ein Payload in Form eines Wahrheitswerts reicht, erweisen sich die abweichenden Werte vor allem beim Debuggen im Node-RED Dashboard oder auch dem Mikrocontroller als Mehrwert.

Der über den Endpunkt Heartbeat ausgesendete Payload ist:

```
{"msg":"I am alive"}
```

Der über den Mail-Arrived Endpunkt ausgesendete Payload ist:

```
{"msg":"Mail has arrived"}
```

Der über den Battery-Status Endpunkt ausgesendete Payload ist:

```
{"batteryStatus":<Batteriespannung in Prozent>}
```

Der über den Calibrate Endpunkt ausgesendete Payload ist:

```
{"msg":"Set calibration value"}
```

Softwarebestandteile

Der grundlegendste Bestandteil der Software ist das espressif8266 Framework. Auf dem Framework aufbauend wird die individuelle Funktionalität der Software implementiert.

Darüber hinaus werden folgende Pakete benötigt:

- WiFiManager im Folgenden Wifi Manager bezeichnet
- ArduinoJson im Folgenden JSON-Bibliothek bezeichnet
- PubSubClient im Folgenden MQTT-Client bezeichnet
- NewPing im Folgenden Entfernungssensor Bibliothek bezeichnet

Der Wifi Manager wird zur Eingabe der WLAN-Zugangsdaten und Verbindungsherstellung mit dem WLAN benötigt, außerdem zur langfristigen Speicherung der WLAN-Zugangsdaten.

Die JSON-Bibliothek wird vor dem Speichern zur Serialisierung und nach dem Lesen zur Deserialisierung der Inhalte der Konfigurationsdatei genutzt. Die abgelegte Konfigurationsdatei ist im JSON-Format.

Der MQTT-Client wird zur Verbindungsherstellung zum MQTT-Broker benötigt. Bei aufgebauter Verbindung können Subscriber gesetzt und Publisher über den MQTT-Client ausgeführt werden.

Die Entfernungssensor Bibliothek wird für Anfragen an das Ultraschallmodul sowie zur Umrechnung der vom Ultraschallmodul gelieferten Daten genutzt. Die vom Ultraschallmodul gelieferten Daten werden in Zentimeter abgefragt.

Nicht alle genutzten Funktionalitäten aus dem Framework werden hier im Einzelnen genannt. Nur Einige sind erwähnenswert. Zum Beispiel der Wifi Client, welcher für die Erstellung des Access Points in der Software des Wireless Access Points genutzt wird oder das Dateisystem (LittleFS), welches für die Speicherung und das Auslesen der Konfigurationsdatei in der Hauptsoftware genutzt wird.

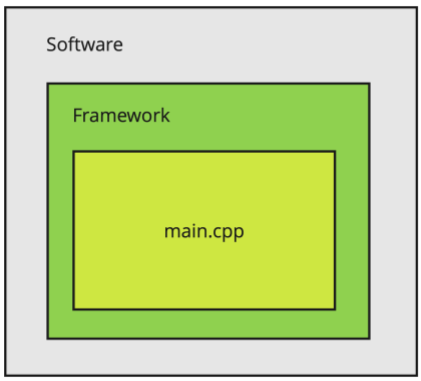

Abbildung 29: Softwarebestandteile Wireless Access Point

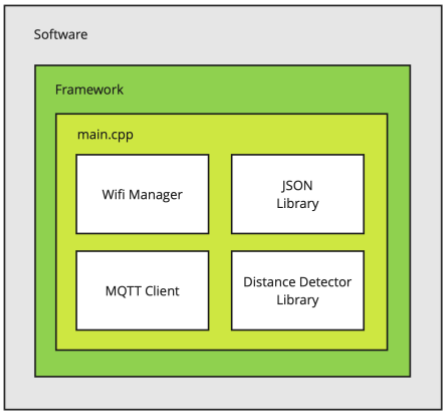

Abbildung 30: Softwarebestandteile Hauptsoftware

Softwareablauf Setup

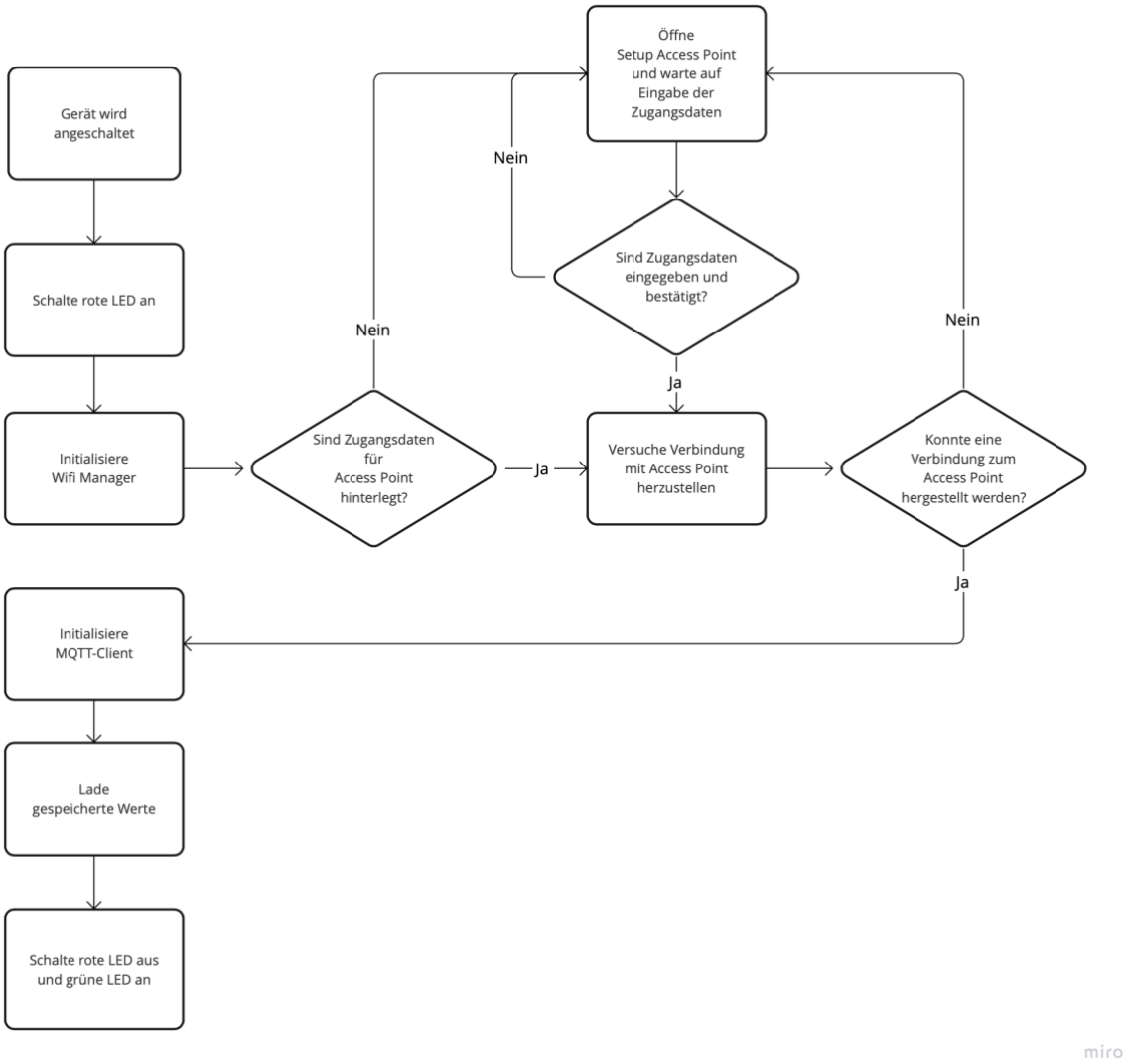

Abbildung 31: Softwareablauf Setup

Softwareablauf Loop

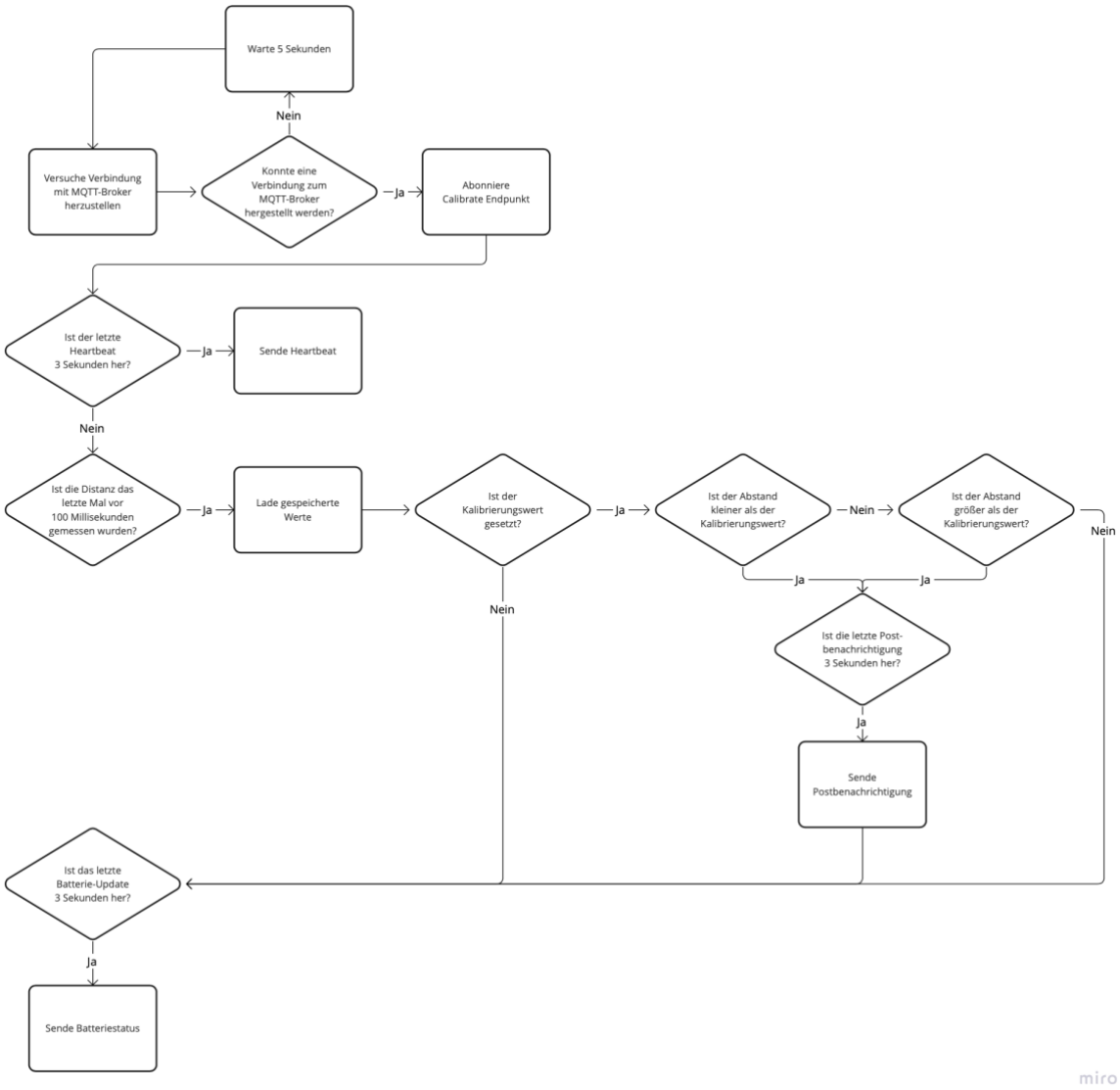

Abbildung 32: Softwareablauf Loop

Loggen

Das Loggen der Ausgaben erfolgt im seriellen Monitor nur durch benannte Präfixe im Format:

```
Serial.println("[<Paketname>] <Wert>");
```

dadurch soll im Fehlerfall eine schnellere Unterscheidung ermöglicht werden. Zusätzlich werden die Ausgaben sparsam gehalten und damit eine Überflutung des seriellen Monitors verhindert. Zuviel Ausgaben könnten der Übersicht schaden und eine Fehlerbehandlung unnötig kompliziert machen.

Debuggen

Da PlatformIO für das ESP8266 Board im Moment keine aktive Debug-Session anbietet, erfolgt das Debuggen ausschließlich über den seriellen Monitor. Daher muss hier insbesondere auf die Nutzung von sinnvollen Ausgaben sowie Debug Flags der einzelnen Pakete zurückgegriffen werden.

Nicht nur der serielle Monitor, sondern auch der Mikrocontroller unterstützt bei der Fehleranalyse. Die angebrachten LEDs zeigen den aktuellen Status des Geräts an. Dabei steht die rote LED für aktiv, aber nicht bereit und die grüne LED für aktiv und bereit. Nur jeweils eins der LEDs kann aktiv sein.

Somit wird nicht nur bei der Benutzung, sondern auch beim Entwickeln deutlich gemacht, ob das Setup nach dem Anschalten des Geräts erfolgreich durchgelaufen ist.

Konfigurationsdefinitionen

Alle Konfigurationsdefinitionen befinden sich am obersten Punkt des Quelltexts. Die Konfigurationswerte können ohne Bedenken angepasst werden. Sollten Fehler auftreten reicht das Zurücksetzen in den ursprünglichen Konfigurationswert. Um Fehler auszuschließen, sollten sinnvolle und wahrheitsgemäße Konfigurationswerte genutzt werden.

Nützliche Anpassungen können bei Folgenden Konfigurationsdefinitionen vorgenommen werden:

```
#define FLAG_DEBUG_WIFI true

#define AP_SSID "Smart Mailbox Setup"
#define AP_SECRET "abcDEF123!"

#define MQTT_SERVER_IP "192.168."
#define MQTT_SERVER_PORT 1883

const ulong LAST_HEARTBEAT_FREQUENCY = 3000;
const ulong LAST_DISTANCE_UPDATE_FREQUENCY = 100;
const ulong LAST_MAIL_ARRIVED_UPDATE_FREQUENCY = 3000;
const ulong LAST_BATTERY_STATUS_UPDATE_FREQUENCE = 3000;
```

Die genannten Konfigurationsdefinitionen sollten weitestgehend selbsterklärend sein.

Frequenzwerte

Die gesetzten Frequenzwerte regulieren die Ausführung der publish Aufrufe über den MQTT-Client.

Die Festlegung der Frequenzwerte haben sich iterativ ergeben.

Software aufspielen

Das Aufspielen der Software wird mit Hilfe des PlatformIO Plugins erledigt. Hierbei wird lediglich die Aktion Upload ausgewählt und auf den Play-Button geklickt. Das automatisch geöffnete Terminal zeigt den Ablauf des Builds an. Ist der Build erfolgreich ausgeführt, wird eine Erfolgsmeldung ausgegeben. Sollte der Build nicht erfolgreich sein, wird eine detaillierte Fehlermeldung ausgegeben.

Abbildung 33: Software aufspielen

Bekannte Probleme

Auf dem Mac kann der serielle Monitor vom PlatformIO Plugin nicht ausgeführt werden.

Wenn die Verbindung zum MQTT-Broker nicht hergestellt wird, wird die LED dennoch auf grün geschaltet. Das Problem liegt im Setup, da die LED nicht durch einen zuvor ausgeführten MQTT-Broker Verbindungstest im Fehlerfall blockiert wird. Der Verbindungstest wird erst später in der Loop ausgeführt.

Der Batteriestatus wird nicht richtig berechnet. Die Dokumentation zum ADC Pin gibt keine konkrete Definition zum eingegangenen Wert an. Daher wird im Moment der niedrige Wert als hoher Spannungseingang betrachtet und daraus der prozentuale Wert ermittelt.

Das Delta wird nicht genutzt, dadurch werden Ungenauigkeiten nicht mehr toleriert. Außerdem ist eine weitere Bedingung, ob die momentane Distanz größer als der Kalibrierungswert ist, hinzugekommen. Der Entfernungssensor scheint viel zu feinfühlig nach den genannten Anpassungen zu reagieren, schon bei kleinsten Erschütterungen wird ein Posteingang gemeldet.

Die Verbindungen zum Wireless Access Point sind instabil. Das führt zu häufigen Verbindungsabbrüchen und Neuvergaben von IP-Adressen. Da die MQTT-Broker Server-Adresse fest in der Software hinterlegt ist, ist ein Neuaufspielen der Software nicht verhinderbar.

Mögliche Lösungsansätze

Mit dem Kommandozeilenbefehl

```
14:30:00 ~/CLionProjects/Soft-Skills pio device monitor
```

wird der serielle Monitor im Terminal ausgegeben. Um die Ausgabe im seriellen Monitor nicht zu verpassen, kann mit Hilfe eines Delays das Setup verzögert werden.

Der MQTT-Broker Verbindungstest könnte vor Einschaltung der grünen LED im Setup einmal ausgeführt werden. Dadurch würde die grüne LED, solange alle Verbindungstest nicht erfolgreich durchgelaufen sind nicht umschalten.

Der Spannungswert könnte mit Hilfe eines Multimeters abgefragt werden, dadurch kann der tatsächliche vom Mikrocontroller ausgegebenen Wert abgeglichen und eventuell herausgestellt werden, ob der Wert nun für eine höhere oder niedrigere Spannung steht.

Durch das Hinzufügen des Deltas wird die Ungenauigkeit wieder toleriert. Ob nun damit die zu feinfühlige Reaktion verhindert wird, müsste getestet werden.

Die IP-Adresse des Hosts muss auf dem Wireless Access Point reserviert werden. Die IP-Neuvergabe der anderen Geräte im Netzwerk kann weiterhin dynamisch bleiben.