Projektdokumentation „DIY Smart Home“

## Motivation/Problemstellung:

Die Projektgruppe hat sich zu Beginn der Ideenfindungsphase an Alltagssituationen erinnert, in denen der Einsatz einer Smart Home Lösung sinnvoll wäre. Ein Projektmitglied hat sich daran erinnert, dass vor kurzem im Kellerraum seiner Wohnung ein Fahrrad entwendet wurde. Da der Kellerraum von der Wohnung aus nicht einsehbar ist, kam die Idee auf, eine Art Bewegungsmelder zu installieren, der mittels Benachrichtigung auf ein Endgerät vor unbefugtem Zutritt warnt. Der Bewegungsmelder sollte sich „scharf“ stellen lassen, damit dieser nicht jederzeit Alarm schlägt. Ziel ist es also, mit der „Security Box“ Einbrecher und Diebe auf frischer Tat zu ertappen.

## Anforderungskatalog:

###### Notwendige technische Voraussetzungen:

* ESP-8266 WEMOS D1 Mini als Mikrocontroller
* Mobiles Endgerät
* WiFi-Signal für Mikrocontroller und mobiles Endgerät
* USB-Mini Stromversorgung
* Sensoren:
  + PIR Bewegungssensor (U1 HC-SR501)
* Aktoren:
  + 2 5V-LEDs: rot und grün
  + ESP-8266 Button Shield

###### Mögliche technische Voraussetzungen:

* Widerstände 200 Ohm (soll)
* Battery Shield (kann)
* Batterie (kann)

###### Physische Voraussetzungen:

* Design: kompakte Box & Deckel mit Löchern für Sensorik und Aktorik, sowie Stromversorgung
  + Material: Kunststoff-Filament eines 3D-Druckers (PLA)
* Kabel und Lötzubehör zum Verbauen der einzelnen Teile

###### Eingesetzte Hilfssoftware:

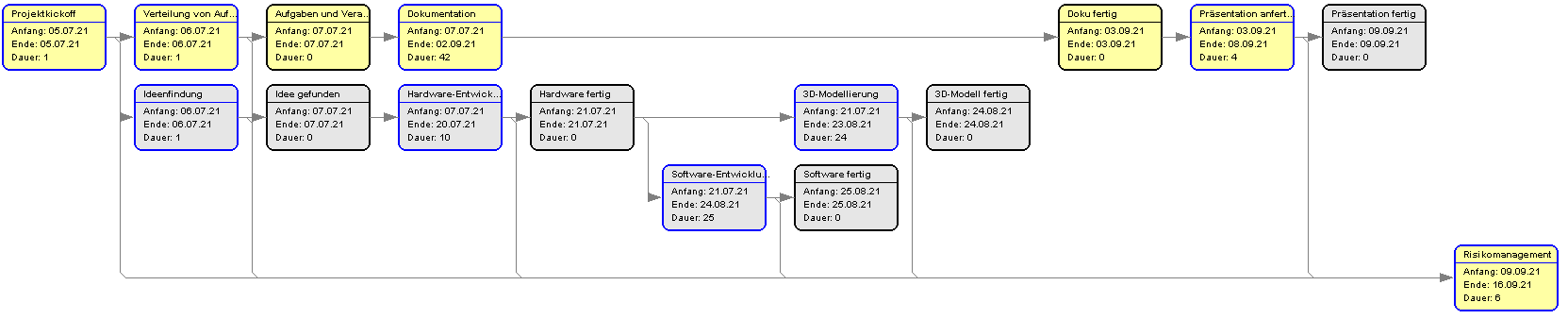
* Arduino IDE
* Note RED
* Autodesk Fusion 360

Software-Funktionsweise (Textuell erklärt):

Die „Security Box“ verfügt zwei LEDs, eine rote und eine grüne. Die grüne LED leuchtet, wenn die Box im Zustand „Home“ ist. Die rote LED leuchtet, wenn die Box im Zustand „Away“ ist. Im Nachtmodus leuchten keine der beiden LEDs. Sobald das Sicherheitssystem in den Zustand „Away“ wechselt, wird durch das Initialisieren eines zwei Minuten Timers eine Früherkennung unterbunden. Nach dem Ablauf des zweiminütigen Timers ist das Sicherheitssystem aktiv, nach dem Erkennen einer Bewegung wird eine entsprechende Nachricht via Discord oder E-Mail an den Benutzer verschickt. Im Zustand „Home“ bleibt das Sicherheitssystem inaktiv. Die Zustände werden durch das Betätigen des Buttons geregelt. Sobald der Button gedrückt wird, wechselt der Status der Box auf „Home“ oder „Away“. Entsprechend leuchtet auch die passende LED.

## Projektplan und Verantwortlichkeiten

In Rahmen dieses Projektes wird ein Projektplan mit Meilensteinen erstellt und die Verantwortlichkeiten und Aufgaben werden in der Gruppe verteilt. In dem Projektplan werden Meilensteine definiert, welche wichtige Fertigstellungen von Arbeitspaketen markieren.



Auf dem Bild wird der Projektplan dargestellt. Die Kästchen mit blauer Umrandung sind Arbeitspakete, die mit schwarzer Umrandung Meilensteine.

Projektkickoff ist der 05.07.21, bei dem die Limitierungen des Projektes bekanntgegeben werden. Daraufhin werden die Verantwortlichkeiten und Aufgaben in der Gruppe verteilt, für diesen Schritt wird ein Tag angesetzt. Verantwortlich für die Hardware ist die gesamte Gruppe, für die Software Leif Meyer und Hendrik Nessen, für die 3D-Modellierung Marvin Timmermann und Felix Büntemeier, welche ebenfalls für die Dokumentation verantwortlich sind.

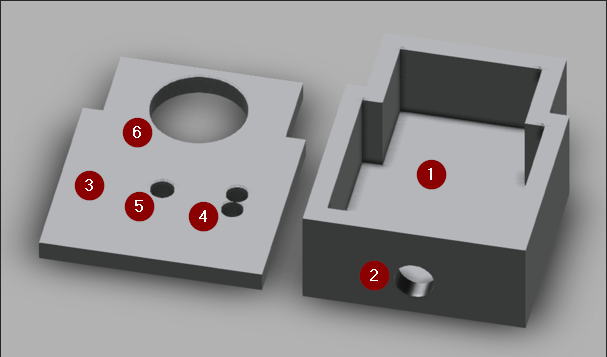
Zeitgleich wird sich mit der Ideenfindung befasst, welche ebenfalls ein Tag braucht bei der alle Gruppenmitglieder beteiligt sind. Nach diesen Meilensteinen wird die Hardware-Entwicklung starten sowie die Dokumentation. Für die Hardware-Entwicklung werden zehn Tage angesetzt, da sich erst ein Bild gemacht werden muss von der verfügbaren Hardware und dem darauffolgenden Zusammenbau. Für die Dokumentation wird ein Zeitraum bis Anfang September eingeplant, damit über die gesamte Projektdauer dokumentiert werden kann.

Die 3D-Modellierung kann, nachdem die Hardware fertiggestellt ist, beginnen, da die Maße der Hardware vorher unklar sind. Für diesen Schritt werden zwölf Tage eingeplant, damit die Modelle genau angepasst werden können. Zeitgleich wird mit der Software-Entwicklung begonnen, welche mit einem Monat eingeplant ist, damit alle Funktionen ausgiebig getestet werden können. Zeitgleich wird die Dokumentation mit den jeweiligen Inhalten aus den Arbeitspaketen gefüllt. Nachdem die Dokumentation fertiggestellt ist, wird mit der Erstellung der Präsentation begonnen, dafür werden vier Tage eingeplant.

Damit ein Risikomanagement in dem Projekt stattfinden kann, werden sechs Tage eingeplant. Falls ein Problem auftreten sollte, werden die darauffolgenden Aufgabenpakete entsprechend nach hinten verlagert. Zudem wird im gesamten Projekt für eventuelle Probleme oder Verzögerungen Zeit eingeplant, sodass Hardware, Software, 3D-Modell sowie die Dokumentation frühzeitig fertiggestellt werden können.

## Beschreibung der 3D-Modellierung

Nachdem die Hardware-Entwicklung fertiggestellt worden ist, konnte der Modellierungsprozess beginnen. Durch die fertige Hardware waren Maße gegeben, wie groß die „Security Box“ sein muss. Damit diese auch kompakt ist, sollte das Gehäuse für die Hardware so klein wie möglich sein. Zunächst wurde ein einfaches Gehäuse modelliert, bei dem der Deckel abnehmbar ist. Zusätzlich mussten Aussparungen eingefügt werden: Für die Stromversorgung, die LEDs, den Button und den Bewegungssensor. Diese wurden mit den Maßen der Hardware vorgenommen.

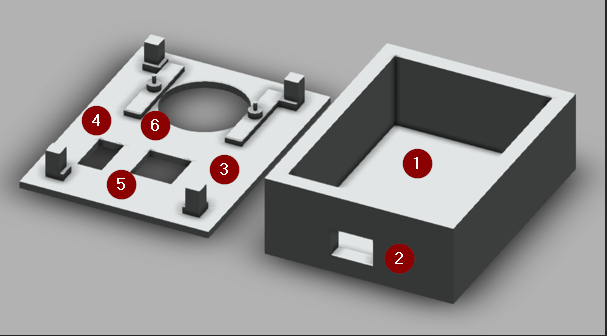


Auf dem Bild ist das erste Modell zu sehen:

* Punkt 1 ist der Körper,
* Punkt 2 die Aussparung für die Stromzufuhr,
* Punkt 3 ist der Deckel,
* Punkt 4 sind die Aussparungen für die LEDs,
* Punkt 5 ist die Aussparung für den Button und
* Punkt 6 ist die Aussparung für den Bewegungssensor.

Zunächst wurde nur der Deckel gedruckt, da der Bewegungssensor, die LEDs und der Button durch die Aussparungen passen müssen. Es stellte sich heraus, dass die Aussparungen nicht passend waren, sodass das Modell überarbeitet werden musste. Hierfür wurde eine Skizze vom Bewegungssensor für Autodesk Fusion 360 benutzt (https://www.thingiverse.com/thing:1673521), welche die genauen Abmessungen des Bauteils hatte.

Zusätzlich wurde diese Skizze erweitert, sodass die LEDs und der Button genau passen. Damit der Deckel auf dem Körper fest ist, wurden vier Eckpfeiler eingefügt, sodass diese genau in die Ecken des Körpers passen. Bei dem Körper wurde die Aussparung für die Stromversorgung ein wenig nach links verschoben und zudem eckig modelliert.

Auf dem Bild ist das fertige Modell zu sehen:

* Punkt 1 ist der Körper,
* Punkt 2 die Aussparung für die Stromzufuhr,
* Punkt 3 ist der Deckel,
* Punkt 4 ist die Aussparung für die LEDs,
* Punkt 5 ist die Aussparung für den Button und
* Punkt 6 ist die Aussparung für den Bewegungssensor sowie die Haltevorrichtung für diesen.

Maße

Dieses Modell wurde im „Makerspace“ gedruckt.

## Hardware-Entwicklung

Im Folgenden wird beschrieben, wie mit den im „Makerspace“ ausgesuchten Hardware-Bauteilen ein funktionierender Schaltplan entwickelt und dieser Schritt für Schritt kompakt zusammengelötet wurde:

Hier sind erneut die verwendeten Bauteile:

* ESP8266 Lolin Wemos D1 Mini
* ESP8266 Button-Shield
* Zwei 5V LEDS (Rot und Grün)
* Ein Bewegungssensor (U1 HC-SR501)
* (100 Ohm Widerstand)
* Prototyp Platine (4.5cm x 3.5cm)

Zu Beginn hieß es, sich Gedanken über die Verkabelung der einzelnen Bauteile zu machen, da nur so eine logische Verlötung der einzelnen Pins am Mikrocontroller mit den Pins der Bauteile möglich ist. Der abgebildete Schaltplan (ohne Widerstand) stellt dar, wie das aktuelle Projekt verlötet ist.

Diagram

Description automatically generated

Die darauffolgende Abbildung stellt denselben Schaltplan mit Widerstand dar. Dieser wird zwar empfohlen, wurde allerdings nicht genutzt, da die 5V-LEDs auch ohne einen passenden Widerstand mit einer angelegten Spannung von 5V funktionieren.

Diagram, schematic

Description automatically generated

Zum Verlöten wurde zuerst der ESP-8266 auf der Platine platziert, sodass Platz für die LEDs auf der rechten Seite (von oben gesehen) geschaffen wurde. Mittels des Stecksystems kann der Button-Shield auf den ESP-8266 gesteckt werden. Die LEDs wurden neben ihren Pins (D2 für Grün und D4 für Rot) mittels der Anode mit deren untersten Kante auf Höhe der Platine des Button-Shields platziert. Die Kathode der beiden LEDs wurde auf Höhe von Pin D4 gesteckt (falls ein Widerstand eingebaut wird, müsste dies angepasst werden).

Zu Beginn wurden nur die ersten Bauteile verlötet: Der ESP-8266, der Button-Shield und die Steckhalterung für den Bewegungssensor. Es wurde auf folgendes geachtet:

* Nur 5V, GND und D3 wurden beim Button Shield verlötet.
* 5V, GND und D1-D4 wurden bei dem ESP-8266 verlötet.
* Lötstellen sind am ESP-8266 mit Stecksystem trotzdem zu verlöten, auch wenn das Stecksystem dabei nicht mehr bündig schließt.
* Bauteile wurden nur an den nötigen Stellen sowie den Eckpunkten verlötet, um möglichst viel Halt mit möglichst wenig Aufwand zu gewährleisten.

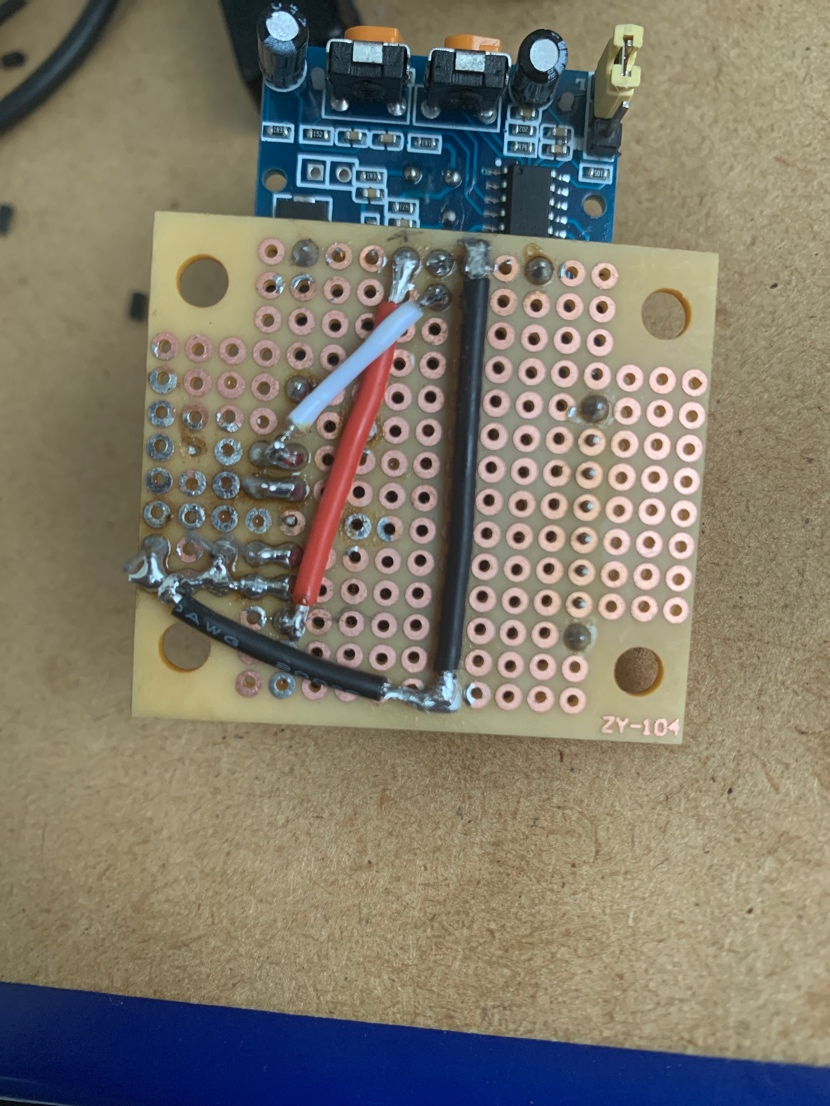
Im weiteren Verlauf des Projektes wurden die LEDs sowie die Verkabelung des Bewegungssensors verlötet. Davor war noch nicht klar, ob LEDs mit oder ohne Widerstand verlötet werden sollten, es wurde entschieden, ohne Widerstand zu verlöten. Es wurde hier auch auf folgendes geachtet:

* Die LEDs wurden mittels einer Brücke/Leitung an GND verlötet.
* Die LEDs wurden direkt zu ihrem zugehörigen Pin verlötet.

A picture containing electronics

Description automatically generated

* Von der Steckhalterung aus wurde wie im folgenden Bild die Verkabelung von 5V, GND und OUT/D1 verlötet. Dabei wurde aufgepasst, dass die Kabel über möglichst direkte, aber saubere Wege verfügen.



## Probleme bei der Hardware-Entwicklung

Da wir hier schrittweise vorgegangen sind, gab es immer nur kleinere Probleme, die aber immer innerhalb eines Tages gelöst wurden. Größtenteils war die Ursache, dass die Lötstellen nicht sauber waren. Folgende Probleme wurden gelöst:

* Der Button Shield funktionierte nicht wie erwartet, da die Lötstellen zwischen dem ESP-8266 und dem Button Shield nicht funktionstüchtig waren.
* Es war zu einem Zeitpunkt nicht klar, wie der Widerstand einzubauen war. Dabei wurde versucht, die LEDs mit dem Widerstand zu verbinden und dann den Widerstand mit GND zu verlöten. Die LEDs hatten sich danach nicht mehr ansprechen lassen. Darauffolgend wurden die LEDs erneut ohne Widerstand verlötet, die LEDs konnten wieder angesprochen werden.

Ein Problem wurde nicht gelöst:

* Die Rote LED wurde auf den Pin D4 gelötet. Das Problem ließ sich erst bei der Programmierung bemerken. Die eingebaute interne LED vom ESP-8266 lässt sich auch mit D4 ansteuern. So gab es einen Konflikt mit der Software, da immer nur eine der beiden LEDs willkürlich leuchtete.

## Beschreibung der Systemarchitektur und der Softwareentwicklung

Diagram

Description automatically generated

Abbildung 1/ Systemübersicht

Das Sicherheitssystem stellt bei Inbetriebnahme eine Verbindung zu einem spezifizierten Netzwerk her. Dies ist notwendig, um eine Kommunikation zwischen den Systemen zu gewährleisten. Erkennt das Sicherheitssystem eine Bewegung, wird eine entsprechende Meldung via MQTT über das Internet übermittelt. Die Daten werden von einem Broker (Mosquitto) empfangen und an einen Node-RED Server weitergeleitet.

Die empfangenen Daten werden mittels Node-RED als Email-, sowie Discord-Nachricht verarbeitet und weitergeleitet. Die verarbeitete Meldung erscheint schließlich auf den jeweiligen Endgeräten.

Diagram

Description automatically generated

Abbildung 2/ Zustandsdiagramm

## Setup

#### Parameter/Schnittstellen

Text, letter

Description automatically generated

Abbildung 3/ Bibliotheken und Globale Parameter

Text

Description automatically generated

Abbildung 4/ Setup Methode

#### Netzwerk Setup

Bei Start des Sicherheitssystems wird beginnend das „NETWORK SETUP“ des Programms ausgeführt. In dieser Phase werden Verbindungen zum Netzwerk, zum MQTT-Broker sowie zum Node-RED Client hergestellt und aufrechterhalten. Der Nutzer muss hier Wifi und MQTT Parameter anpassen. Graphical user interface, text, application

Description automatically generated

Abbildung 5/ Netzwerk und MQTT Parameter

Der Quellcode stammt aus den vorherigen Übungsaufgaben und wurde nicht weiter überarbeitet.

#### Button Class von Spivey

Text, letter

Description automatically generated

Abbildung 6/ Button Class

Die von [Spivey entwickelte Button-Class/Logik](Spivey%20entwickelte%20Button-Class/Logik) (https://www.hackster.io/Spivey/wemos-d1-mini-esp8266-smart-iot-button-with-messagebird-dbef7e) ermöglichte eine flexible Programmierung des Tasters. Im Setup wird die obige Klasse erstellt. Dabei sind Parameter für den Pin des Tasters, die zwei Dauerangaben, „short“ und „long“, und ein Zähler für die Dauer eines Drucks initialisiert. Zudem wird auch den vorherigen und aktuellen Zustand des Tasters (HIGH oder LOW) initialisiert.

Es ist wichtig zu bemerken das der Taster „active-low“ ist. Dies bedeutet das der Taster, falls gedrückt, ein LOW Signal ausgibt.

## Loop Methode

Nachdem alle Verbindungen erfolgreich hergestellt werden konnten, beginnt die „Loop-Methode“ des Programms. In diesem Teil des Programms wird die von Spivey entwickelte Button-Logik (https://www.hackster.io/Spivey/wemos-d1-mini-esp8266-smart-iot-button-with-messagebird-dbef7e) und des Sicherheitssystems umgesetzt. Diese Button Logik wurde wie folgt abgeändert.

Abbildung 7/ Button Logik

Text

Description automatically generated

Die von Spivey entwickelte Button-Logik (https://www.hackster.io/Spivey/wemos-d1-mini-esp8266-smart-iot-button-with-messagebird-dbef7e) nutzt Zustände und Timer, um zwischen kurzen und langen Tastendrücken zu unterscheiden. Dies ist für die spezifische Funktionalität des Sicherheitssystems notwendig. Erkennt das System einen kurzen Knopfdruck, wechselt das System abhängig vom vorherigen Zustand in den „Home“- oder den „Away“-Zustand. Registriert das System einen langen Knopfdruck, wird der Nachtmodus des Sicherheitssystems aktiviert. Die LED-Leuchten werden deaktiviert und der aktuelle Zustand bleibt bestehen.

Text

Description automatically generated

Abbildung 8/ Short und Long Press

Das eingeschaltete System befindet sich im Ausgangszustand „Home“. In diesem Zustand wird lediglich die grüne LED-Leuchte aktiviert. Es werden in diesem Zustand keine Bewegungsdaten erfasst und ausgewertet.

Nachdem das System in den „Away“ -Zustand gelangt, erlischt die grüne LED-Leuchte und die rote LED wird aktiviert. Mithilfe eines im Setup definierten Zählers „counter“, kann zwischen Zuständen vor und nach der Timerlogik separiert werden. Dabei wird bei erstmaligem Eintritt in den „Away“-Zustand durch den vordefinierten Zähler das Sicherheitszeitintervall von zwei Minuten gestartet. Nachdem der Timer abgelaufen ist, gelangt man durch das Inkrementieren des Zählers in den Folgezustand, welcher die Bewegungssensorik aktiviert.

Erkennt das Sicherheitssystem eine Bewegung, wird eine entsprechende Meldung an den MQTT-Broker übermittelt. Nach einer kurzen Verzögerung wird der Zähler wieder zurückgesetzt und das Programm gelangt wieder in den Zustand, welcher für die Bewegungserkennung verantwortlich ist. So kann eine Überwachung ohne erneute Einhaltung des Sicherheitszeitintervalls gewährleistet werden.

Die Datenübertragung wird über einen mit Node-RED eingerichteten Server abgewickelt. Innerhalb des Programms wird ein vordefiniertes Topic („MFHLSensor“) genutzt, um eine Kommunikation über den MQTT Broker zu ermöglichen. Bei eintretender Bewegungserkennung wird mit der „publishString“ Methode eine Kommunikation zwischen den Schnittstellen eingeleitet. Nachdem die Daten an Node-RED übermittelt worden sind, muss eine Weiterleitung an die entsprechenden Endgeräte stattfinden.

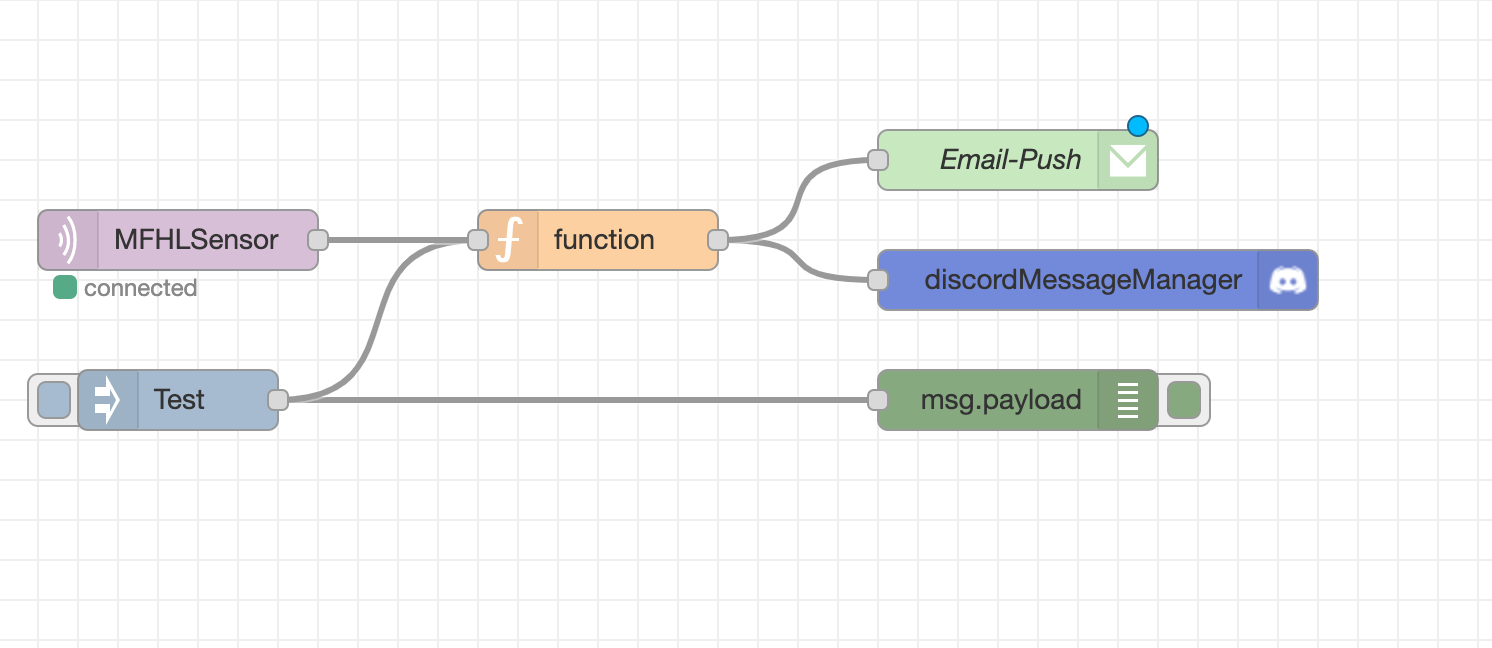


Abbildung 9/ Node-RED Konfiguration

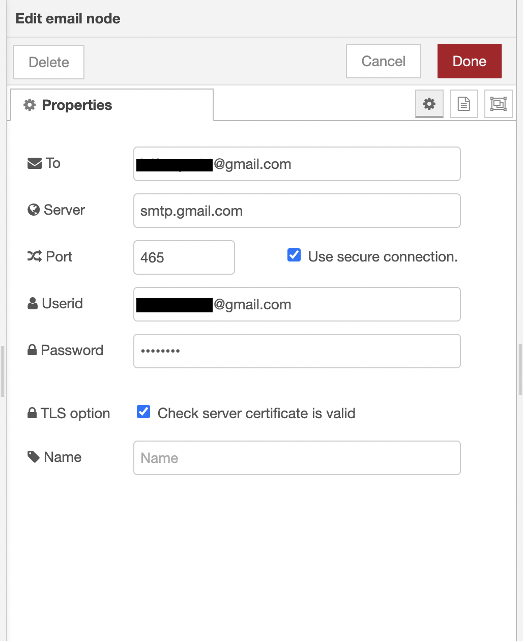
Graphical user interface, text, application, email

Description automatically generated

Abbildung 10/ Function Node

Über das Netzwerk-Node „mqtt-in“ kann durch das Festlegen eines vorspezifizierten Servers, sowie Topics ein Dateninput verarbeitet werden. Dabei werden die Nodes „discordMessageManager“ und „email“ zur Weiterleitung einer Push-Benachrichtigung genutzt.

Um eine Push-Benachrichtung an eine E-Mail Adresse weiterzuleiten, muss das „email“ Node lediglich mit Server, Port und den ID-Daten des Nutzers synchronisiert werden.



Um eine Push-Benachrichtigung via Discord-Bot durchzuführen, muss eine Applikation, sowie ein Discord-Bot initialisiert werden. Über das Developer-Portal von Discord kann eine entsprechende Applikation erstellt werden, welche das Verwenden eines Discord-Bots ermöglicht.

Zuerst wird ein Discord-Bot erstellt. Dafür folgt man den folgenden Link, um die Discord Developer Seite zu erreichen <https://discord.com/developers/applications>. Nach den einloggen kann es sein, dass der Link erneut eingegeben werden muss.

Zunächst ist eine Applikation sowie ein Bot in dieser Applikation zu erstellen. Dann muss der Bot mit einem Server verbunden werden. Der Bot wird mittels der OAuth2 Funktion einen Server zugewiesen (dafür muss aber ein Server schon erstellt sein). Die Attribute des Bots werden hier auch ausgesucht.

Hier wird als SCOPES:={bot} und BOT PERMISSIONS:={Send Messages} ausgewählt.Graphical user interface

Description automatically generated

Der Link muss hier kopiert und in dem Browser eingefügt werden. Danach wird nach dem Server gefragt, der diesen Bot empfangen soll. Dies schließt das Setup von Discord aus ab.

Graphical user interface, application

Description automatically generated

Abbildung 11/ Discord Developer Portal

Der erstellte Discord-Bot wird mit einem gegebenen Token über Node-RED initialisiert. So kann eine Kommunikation zwischen den Systemen stattfinden und eine Meldung an die Endgeräte weitergeleitet werden.

Code & Bibliotheken

Probleme bei der Software-Entwicklung: