Projekt Pflanze

Thema: üK 216 Endgeräte in bestehende Systeme einschliessen

Dokumentinformationen

Dateiname: blj2025_uek216_2_dokumentation
Speicherdatum: 23.01.2026

Autoreninformationen

Autor 1: Nico Schult
Autor 2: Lionel Schutze
Autor 3: Julian Gasser

Inhaltsverzeichnis

Abbildungsverzeichnis 3
Tabellenverzeichnis 3
Änderungsgeschichte 3
1 Einleitung 4
1.1 Sinn und Zweck 4
1.2 Referenzdokumente 4
1.3 Ausgangslage 4
1.4 Auftrag 4
1.5 Ressourcen 4
2 Planung 5
2.1 Rollenaufteilung 5
2.2 Hardwarekomponente 5
2.3 Logistik 9
2.4 Vorgehen 9
3 Ausführung 10
3.1 Physisch 10
3.2 Digital 10
4 Systemübersicht 11
4.1 Sensoren und Aktoren 11
4.2 ESP32 12
4.3 MQTT Server 12
4.4 Node-RED 12
5 Netzwerk und Kommunikationskonzept 12
5.1 Netzwerkplan 12
5.2 Internetanbindung ESP32 13
6 MQTT Konzept 13
6.1 Topic Struktur 13
6.2 Publish Logik 13
7 Parameter und Konfiguration 13
7.1 Erste Automatisierung 13
7.2 Zweite Automatisierung 14
8 Hardwareeingang 15
9 Zugriffskonzept 15
9.1 MQTT Lesen 15
9.2 Node-RED ändern 15
9.3 Schwellenwerte einstellen 15
10 Schutzkonzept 15
11 Update-Konzept 16
11.1 ESP32 16
11.2 Node-RED 16

12 Testkonzept und Protokoll 16
13 Erweiterungen 18
14 Fazit 18

Abbildungsverzeichnis

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

Tabellenverzeichnis

Tabelle 1 Versionen 3
Tabelle 2 Abkürzungen ..... **Fehler! Textmarke nicht definiert.**

Änderungsgeschichte

Version	Datum	Autor	Details
1.0	21.01.2026	Julian	Dokument erstellt
1.1	22.01.2026	Julian	Inhalt bis Punkt 3 erhalten
1.2	22.01.2026	Julian	Bis und mit Punkt 12 erweitert
2	23.01.2026	Julian	Fertigstellung der Dokumentation

Tabelle 1 Versionen

1 Einleitung

1.1 Sinn und Zweck

Das vorliegende Dokument beschreibt die Dokumentation und sonstige Informationen zum Projekt Pflanze im üK 216, zweite üK Gruppe.

1.2 Referenzdokumente

[1] Unser GitHub, mit allen Projektunterlagen

1.3 Ausgangslage

Für unser Projekt haben wir folgende Ausgangslage:

Die Noser Young AG in Zürich verfügt über viele Pflanzen, wovon die meisten echt sind. Die echten Pflanzen müssen natürlich bewässert werden, jedoch unterschiedlich oft. Bei einem einfachen Bewässerungs-Zeitplan werden Pflanzen gelegentlich überwässert. Gleichzeitig gibt es Pflanzen, die beim Bewässern oftmals vergessen gehen, weil sie ein wenig versteckt sind.

1.4 Auftrag

Unser Auftrag ist es, aus der vorhin genannten Ausgangslage mittels zweier vorgegebenen Automatisierungen eine Lösung zu entwickeln.

Folgende Voraussetzungen wurden uns gegeben:

- Die erste Automatisierung soll der Zustand der Pflanze beobachten und visualisieren
- Die zweite Automatisierung befasst sich mit der Konfiguration von Schwellenwerten
- Für MQTT-Topics ist das Muster zuerich/plant// zu verwenden
- MQTT-Publish passiert aufgrund von sinnvollen Datenänderungen, in regelmässigen Zeitintervallen nur mit sinnvoller Begründung

1.5 Ressourcen

Uns stehen folgende Ressourcen zur Verfügung:

- Kursunterlagen 00 bis 04 (Theorie, Codebeispiele)
- Max. 3 ESP32-basierte Mikrocontroller pro Gruppe
- Komponentenkiste für Sensoren, Aktuatoren und Peripherie
- Vorkonfigurierter MQTT-Server
- Basisinstallation von Node-RED und benötigte Zugangsdaten
- WLAN für Drahtlosverbindung zum MQTT-Server

2 Planung

2.1 Rollenaufteilung

Am wichtigsten ist es, klare Rollen für ein Team zu haben. Vergebende Rollen sind:

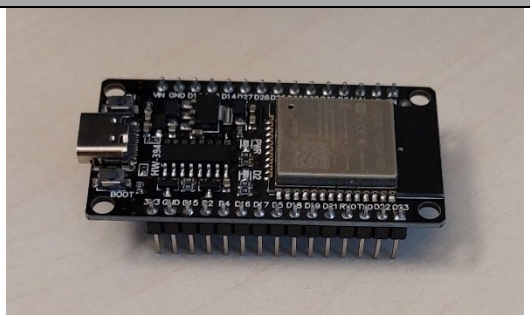
- Dokumentation
- Analoger Teil des Produktes (Elektronik)
- Code schreiben
- Node-RED hosten
- Tester
- Präsentation hauptverantwortlich

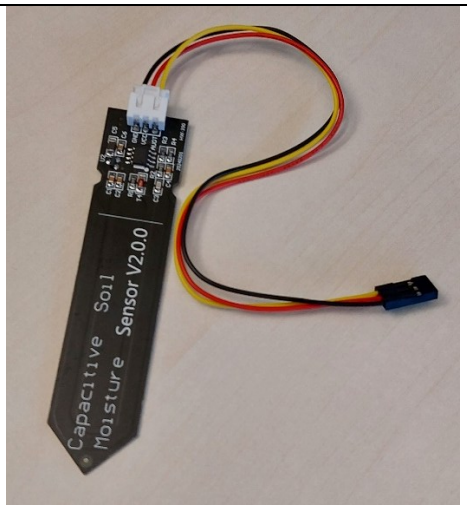
Dokumentation	Elektronik	Code	Node-RED	Tester	Präsentation
Julian	Lionel, Nico	Julian, Nico	Lionel	Alle	Nico

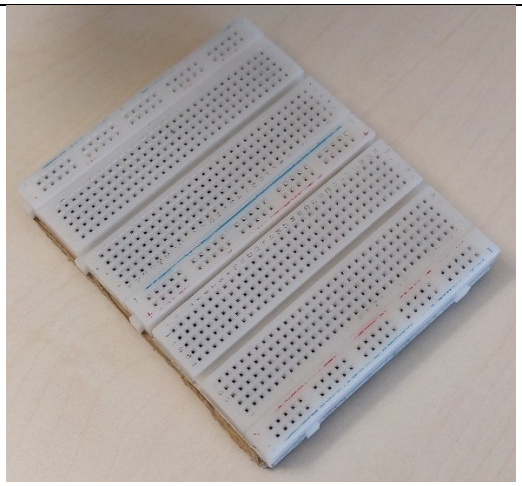
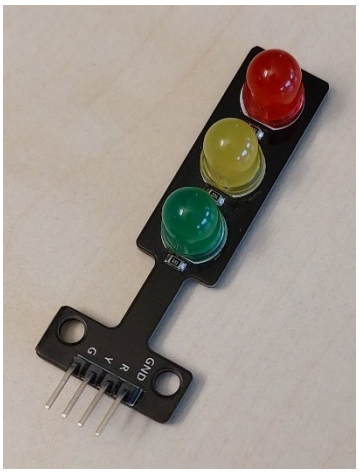
**Diese Aufteilung dient zur Orientierung und muss nicht vollständig eingehalten werden.*

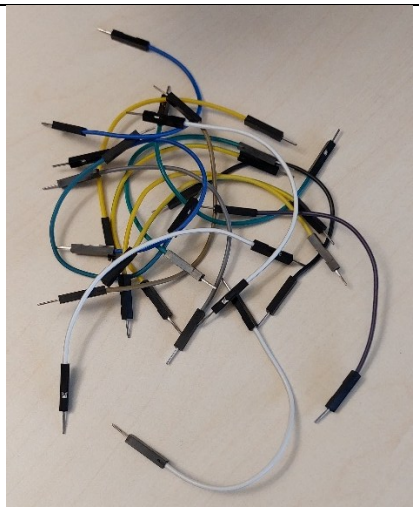
2.2 Hardwarekomponente

Um zu erkennen, wie feucht die Erde ist, haben wir uns für folgende Komponenten entschieden:

Komponenten	Bild	Rolle
ESP32		Mikrocontroller Für Koordination aller Teile und Berechnungen
Feuchtigkeitssensor		Sensor Misst die Feuchtigkeit der Erde.

		
USB-A zu USB-C Kabel		Datenübertragung
OLED Display		Aktor Zeigt den Feuchtigkeitswert und den Status an.
Breadboard		Hilfmittel

		
Ampel		Aktor Zeigt den Status in Farben an.
Buzzer		Aktor Erweiterung, die als Alarm dienen könnte.
Jumper Kabel		Datenübertragung

		
Becher		Behälter Hält die Erde und das Wasser
Erde		Testobjekt Daten, die von Sensor gemessen werden.

2.3 Logistik

Die Automatisierungen fordern, dass man den Zustand beobachten soll und das man Schwellenwerte konfigurieren kann. Um zu wissen, wann eine Pflanze in einem kritischen Zustand ist, braucht man maximal und minimal Werte für die Feuchtigkeit.

Die zweite Automatisierung erfordert, dass der User einfach Grenzwerte bestimmen kann. Das muss möglichst intuitiv sein und wir entschieden uns für folgende Lösung. Man kann über das Node-RED UI eingeben, was für Maximal- und Minimalwerte eine Pflanze hat. Diese Werte sollen dann im Code gebraucht werden, um den Status der Pflanze zu errechnen.

Um den Status vom Wasser von überall zu kontrollieren, entschieden wir uns dazu, die gemessenen Telemetriedaten auf den MQTT Server zu publishen und gleichzeitig auf dem Display anzeigen.

2.4 Vorgehen

Um einen strukturiert arbeiten zu können und die Orientierung nicht verlieren, haben wir uns einen groben Ablauf erstellt. Als erstes haben wir alle Anforderungen durchgelesen, damit wir wissen, was für unser Projekt gefordert ist. Der Ablauf sieht so aus:

1. Auftrag entgegennehmen und verstehen
2. Aufgaben aufteilen
3. Elektronik zusammensetzen
4. Code schreiben, Daten nur lokal anzeigen
5. Daten publishen
6. Maximal- und Minimalwerte per Node-RED festlegen
7. Mit den Grenzwerten den Status ausgeben
8. Kontrollieren ob Automatisierungen erfüllt sind
9. Testen
10. Fortlaufende Dokumentation kontrollieren
11. Präsentation erstellen

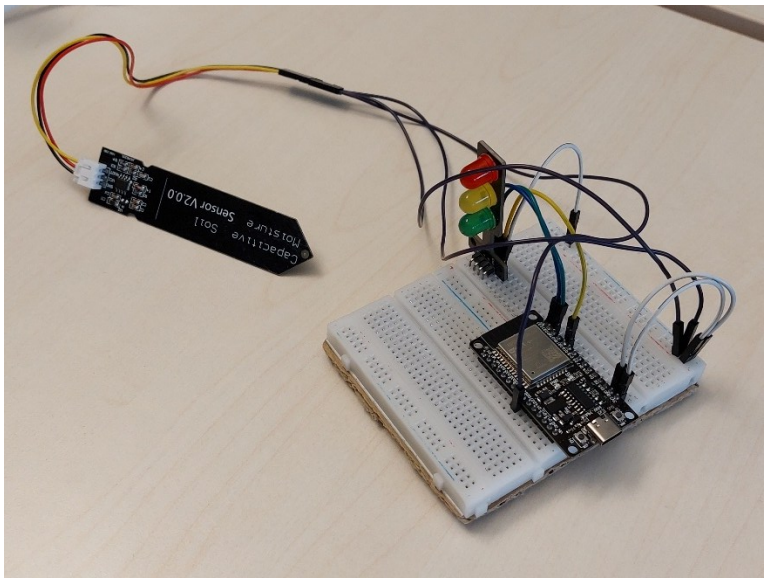
12. Überprüfen, ob alles vollständig ist
13. Präsentieren

3 Ausführung

3.1 Physisch

Als allererstes haben wir die Teile, die oben angegeben sind, angeschlossen und mit dem ESP32 verbunden. Die Ampel sollte dafür zuständig sein, den Status der Erde anzuzeigen, also ob die Mindest- oder Maximalwerte überschritten sind. Das Display zeigt einfach den aktuell gemessenen Wert an. Natürlich haben wir auch noch den Erdfeuchtigkeitssensor, der konstant die Werte misst.

Wir haben uns auch überlegt, einen Buzzer als Alarm zu verwenden. Wir haben uns aber dagegen entschieden, da wir uns aufgrund des Lärms in einen Gruppenraum hätten setzen müssen und der Buzzer keine hohe Priorität hat. Dieser könnte auch als Erweiterung verwendet werden.



3.2 Digital

Anschliessend schrieben wir einen Code, der als erstes einfachheitshalber nur den gemessenen Wert auf dem Display darstellt. Das lief reibungslos und wir gingen zum nächsten Schritt.

Nun wollen wir uns mit dem WLAN verbinden, damit wir die Daten auf den MQTT Server publishen können. Wir brauchten ein paar Versuche, bis wir uns richtig mit dem WLAN verbinden konnten und den MQTT Server auch erreicht haben. Hier hatten wir die Schwierigkeit, dass sich der ESP32 manchmal vom Server trennte und wir weder eine Warnung erhielten, noch der ESP32 sich erneut mit dem Server verbindete. Das mussten wir anpassen, funktionierte am Schluss jedoch.

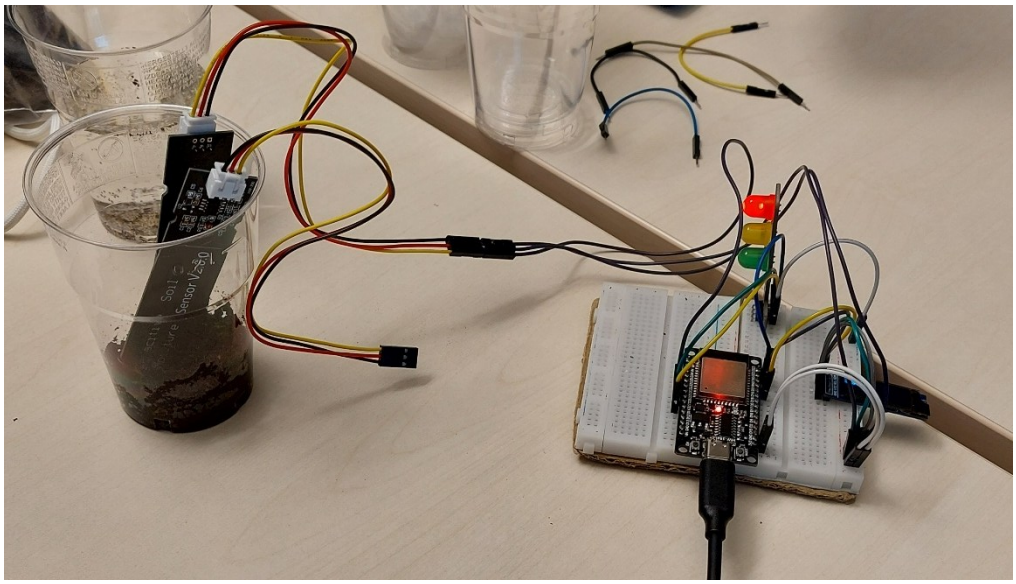
Um die 2. Automatisierung zu erfüllen, müssen wir auch Schwellenwerte konfigurieren können. Wir haben uns dazu entschieden, dass man die Schwellenwerte im Node-RED Dashboard eingeben kann. Um diese Werte zu erhalten, muss man die Werte subscriben und wenn sie sich ändern, soll man kontrollieren, ob man noch im guten Bereich ist. Dieser wird dann wieder auf der Ampel ausgegeben. Hier hatten wir zum einen Probleme, die Werte korrekt zu erhalten und

zum anderen die Werte richtig darzustellen. Das lag daran, dass wir den Pfad falsch geschrieben haben, also MAX, statt Max. Das hat uns viel zu viel Zeit gekostet.

Auf Node-RED haben wir den Feuchtigkeitswert nämlich auf einer gauge*, die die Farbe basierend auf dem Stand ändert. Also wenn der Wert unter oder über den Schwellenwerten ist, soll die gauge rot werden und wenn es im richtigen Bereich ist, grün. Da die Schwellenwerte verändert werden können, muss auch die Darstellung dynamisch sein. Dass stellte sich als sehr schwierig heraus, denn man kann keine Variablen in Node-RED für die Maximal- und Minimalwerte einfügen. Nach etwa vier Stunden, entschied sich Lionel, den Status separat auf Node-RED darzustellen, was wesentlich simpler ist.

Anschliessend mussten wir die Publish- und Subscription-Rate noch minimieren. Der Mindest- und Maximalwert werden ja nur neu gepublished, wenn man den Wert aktualisiert. Beim Sensorwert finde ich, macht es Sinn, dass der Wert nur neu gepublished wird, wenn sich der aktuelle Wert um 20 vom alten gepublishten Wert unterscheidet. Der Stand, ob die Schwellenwerte unter- oder überschritten wird, soll nur neu gepublished werden, wenn sich der Stand ändert.

**gauge – englisch für Messgerät, sieht hier wie der Tachometer eines Autos aus.*



4 Systemübersicht

Das System ist folgendermassen aufgebaut:

4.1 Sensoren und Aktoren

Um Daten zu messen und auszugeben, braucht es Sensoren und Aktoren. Der Sensor ist hier ein Erdfeuchtigkeitssensor. Dieser gibt einen analogen Wert zurück, der später weiterverarbeitet wird.

Aktoren haben wir zwei. Zum einen das OLED Display, der Wert vom Sensor und den Status ausgibt, der berechnet wurde. Zum anderen die Ampel, die nur den Status mittels LEDs anzeigt. Hier ist Grün nicht optimal sondern Gelb, denn Gelb ist mittig und macht von der Auslegung her mehr Sinn. Grün ist daher zu wenig Wasser und Rot zu viel Wasser.

4.2 ESP32

Der ESP32 ist ein Mikrokontroller und führt lokal Rechnungen aus. Er nimmt die Telemetriedaten vom Sensor und rechnet den Status basierend auf den gegebenen Maximal- und Minimalwerten aus. Der Status gibt er an die beiden Aktoren und published sie unter *zuerich/plant/1/state*. Die Telemetriedaten unter *zuerich/plant/1*. Um den Status zu berechnen, subscribed er zu *zuerich/plant/Min_Rohwert* und *zuerich/plant/Max_Rohwert.*

Der ESP32 muss sich auch mit dem WLAN und dem MQTT Server verbinden. Beides macht er sobald er Strom erhält. Der ESP32 kann seine Verbindung zum MQTT Server auch während er läuft verlieren. Das erkennt er und stellt automatisch erneut eine Verbindung zum Server her.

4.3 MQTT Server

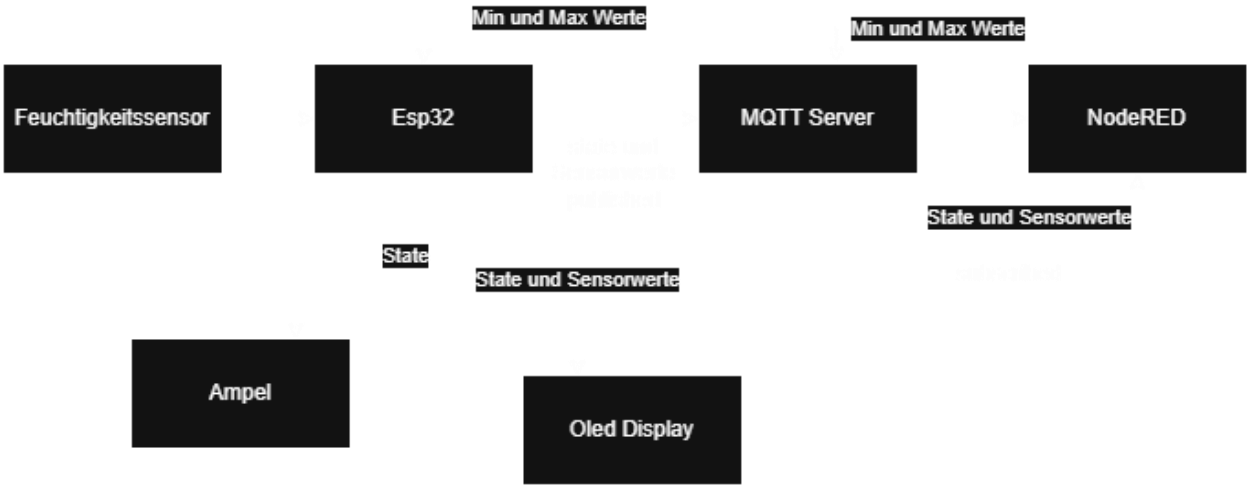
Auf dem MQTT Server werden alle Daten gespeichert. Alle, die ein Topic abonnieren, erhalten die Daten auf dem Topic und alle die auf ein Topic publishen geben Daten an den Server, der sie speichert. Dieser Server wird uns von NoserYoung zur Verfügung gestellt.

4.4 Node-RED

Auf Node-RED haben wir ebenfalls ein UI, bei dem der Sensorwert und der Status angezeigt wird. Wir haben noch einen Graph hinzugefügt um den Verlauf der Daten zu sehen. Auf dem Node-RED UI kann man ebenfalls den Maximal- und Minimalwert festlegen, um die Ansprüche an die Pflanze anpassen zu können. Diese Werte werden unter *zuerich/plant/Max_Rohwert* und *zuerich/plant/Min_Rohwert* im MQTT Server gepublished.

5 Netzwerk und Kommunikationskonzept

5.1 Netzwerkplan



Der ESP32 ist mit der restlichen Hardware, spezifisch: OLED Display, Ampel und Feuchtigkeitssensor per Kabel verbunden. Der ESP32 ist per WLAN mit dem MQTT Server verbunden und Node-RED ist ebenfalls per Internet mit dem MQTT Server verbunden. Der ESP32 published die Telemetriedaten und den Status und subscribed zu den Schwellenwerten von Node-RED. Node-RED published die Schwellenwerte und subscribed zu den Telemetriedaten und dem Status.

5.2 Internetanbindung ESP32

Der ESP32 ist mit dem GuestWLANPortal verbunden und ist bereits vorregistriert, was bedeutet, dass er sich beim WLAN nicht anmelden muss. Die Adresse vom MQTT Server ist 10.10.2.127 und der Port ist 1883. Wenn sich der ESP32 mit dem Server trennt, versucht er automatisch, sich wieder mit dem Server zu verbinden, bis es ihm gelingt. Das gleiche mit dem WLAN. Wenn der ESP32 am Anfang von Loop nicht mit dem WLAN verbunden ist, wird die WLAN Verbindungsfunktion aufgerufen.

6 MQTT Konzept

6.1 Topic Struktur

Der Auftrag hat uns eine Struktur vorgegeben, wie wir unsere Topics benennen müssen. Diese ist wie folgt: *zuerich/plant/<plant-id>/<etc>.* Das macht vor allem Sinn, wenn man viele Sensoren über viele Orte verteilt hat und unterscheiden muss, wo die Sensoren sind. Es ist eigentlich wie eine Ordnerstruktur, um die Daten strukturierter zu speichern.

6.2 Publish Logik

Damit wir nicht die ganze Zeit neu publishen und das Programm weniger Rechenleistung erfordert.

Telemetriedaten werden erst gepublished, wenn sich die Werte um 30 vom vorherig gepublishten Wert unterscheiden. Somit werden nur relevante Änderungen gepublished.

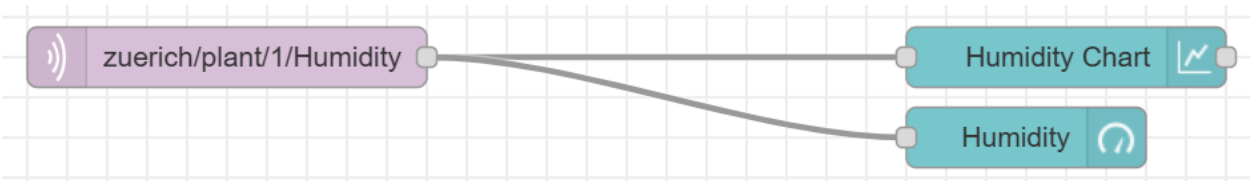
Die Schwellenwerte werden nur neu gepublished, wenn der Wert sich ändert. Der ESP32 lädt dann automatisch den neuen Status der Erde.

Der Status wird ebenfalls nur neu gepublished wenn der Status sich ändert.

7 Parameter und Konfiguration

7.1 Erste Automatisierung

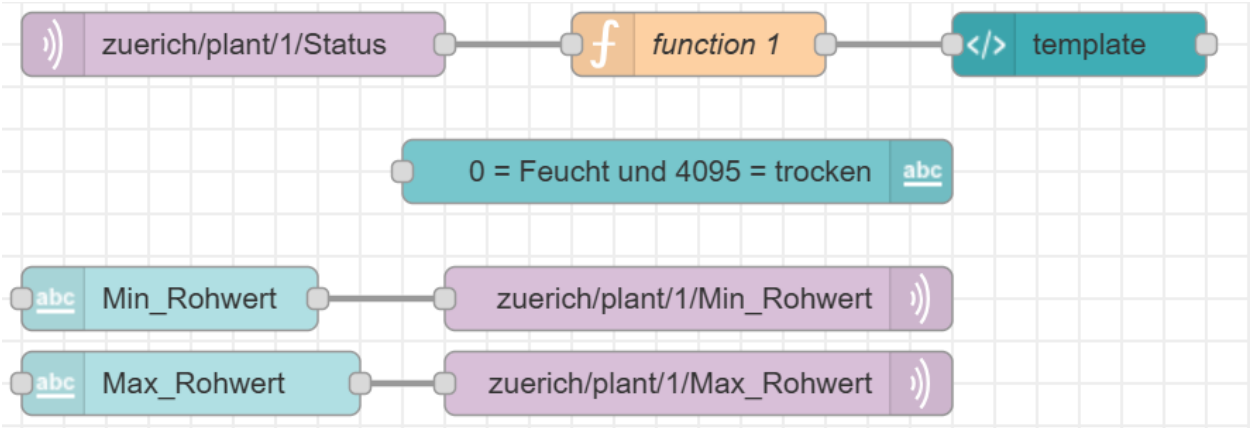
Um den Zustand der Pflanze zu beobachten und zu visualisieren, nimmt man die Telemetriedaten, die man mit Hilfe von analogRead() einliest, also ein analoger Wert von 0 bis 4095. Dieser wird auf den MQTT Server gepublished und anschliessend von Node-RED subscribed. Dort wird der Wert in einem gauge dargestellt.



```
unsigned long currentMillis = millis();

// check if 500 milis over
if (currentMillis - lastMeasureTime >= interval) {
  lastMeasureTime = currentMillis;

  int moisture = readMoisture();
  String status = getStatus(moisture);

/ Only publish when change is over 30
  if (lastPublishedMoisture < 0 || abs(moisture - lastPublishedMoisture) >= 30) {
    String payload = String(moisture);
    client.publish(mqtt_topic_value, payload.c_str());
    lastPublishedMoisture = moisture;
    Serial.print("Published moisture: ");
    Serial.println(moisture);
  }
```

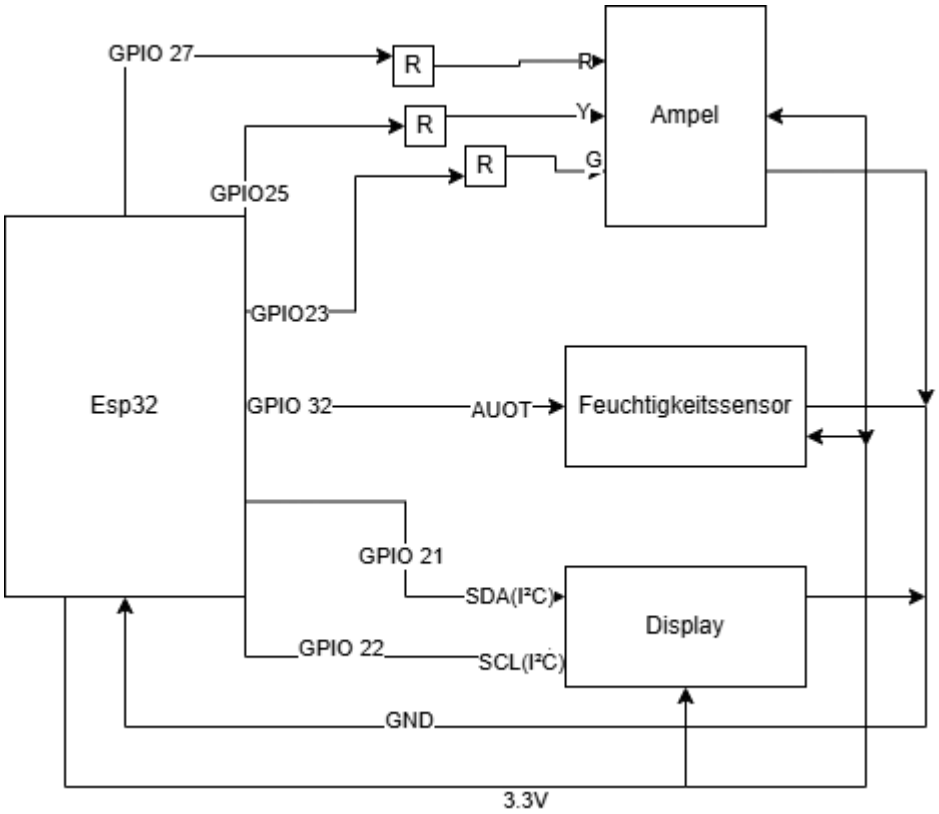
7.2 Zweite Automatisierung

Um Schwellenwerte konfigurieren zu können, haben wir auf Node-RED ein Eingabefenster gemacht, wo man die Schwellenwerte eingeben kann. Diese werden anschliessend auf den MQTT Server gepublished und der ESP32 subscribed dann die Werte. Dann wird auf dem ESP32 mit den Schwellenwerten der Status errechnet. Dieser wird auf der Ampel und dem Bildschirm angezeigt und auf den Server gepublished. Der Status wird wieder auf Node-RED subscribed und angezeigt.




```
String getStatus(int moisture) {
  if (moisture < min_humid) return "wet";
  if (moisture > max_humid) return "dry";
  return "ok";
}
```

8 Hardwareeingang



Die meisten GPIO Pins sind relativ flexibel, nur die Pins 21 und 22 können SDA und SCL gibt es nur einmal. Das Pinout von https://lastminuteengineers.com/ESP32-pinout-reference/ war hier sehr hilfreich.

9 Zugriffskonzept

9.1 MQTT Lesen

Node-RED und der ESP32 darf MQTT lesen, um die gebrauchten Daten zu erhalten.

9.2 Node-RED ändern

Jeder, der das Passwort zu Node-RED hat, darf Node-RED bearbeiten.

9.3 Schwellenwerte einstellen

Jeder der Zugriff auf das Node-RED UI hat, kann die Schwellenwerte bearbeiten.

10 Schutzkonzept

Um Dinge auf Node-RED zu bearbeiten, braucht man ein Passwort. Das Passwort ist stark und damit ist dieser Teil des Systems gesichert.

Der ESP32 selbst hat ebenfalls ein Schutzkonzept und kann nicht einfach angegriffen werden.

Die Nachrichten werden per Caesar-Verschlüsselung verschlüsselt. Das ist nicht wirklich sicher, aber man kann wenigstens nicht direkt den Wert auslesen.

Der MQTT-Server befindet sich im internen Netzwerk und ist nicht direkt über das Internet erreichbar. Dadurch ist der Zugriff auf die MQTT-Daten nur innerhalb der Infrastruktur von Noser Young möglich. Externe Zugriffe sind somit grundsätzlich ausgeschlossen.

Der zur Verfügung gestellte MQTT-Server ist nicht mit Benutzername und Passwort abgesichert. Diese Einschränkung liegt ausserhalb unseres Einflussbereichs, da der Server vorgegeben ist. Aus diesem Grund erfolgt der Schutz nicht auf MQTT-Ebene, sondern über das Netzwerk, in dem sich der Server befindet.

Der Zugriff auf das Netzwerk ist zwar öffentlich, kann aber nur mit Anmeldedaten benutzt werden. Der ESP32 ist vorgängig im Netzwerk registriert, sodass er sich automatisch verbinden kann, ohne dass Benutzername und Passwort im Code hinterlegt werden müssen. Damit wird verhindert, dass sich unbekannte Geräte mit dem Netzwerk verbinden können.

11 Update-Konzept

Updates werden jeweils von den Autoren durchgeführt. Updates von Arduino Libaries können installiert werden, müssen aber nicht, da das System auf dem jetztien Stand funktioniert.

Wenn wir ein Update durchführen, gehen wir wie folgt vor:

11.1 ESP32

1. Das Update wird lokal programmiert.
2. Der neue Code wird auf den ESP32 geladen und getestet (Blackbox)
3. Der Code wird überarbeitet, wenn der Code Fehler beinhaltet.
4. Wenn der Code funktioniert, commitet man auf GitHub.

11.2 Node-RED

1. Das Update wird lokal entwickelt.
2. Das Update wird hochgeladen und getestet (Blackbox)
3. Das Update wird entweder übernommen oder überarbeitet.

12 Testkonzept und Protokoll

Nr.	Testfall	Beschreibung	Erwartetes Ergebnis	Resultat	Status
1	WLAN Verbindung	ESP32 verbindet sich am Start mit WLAN	Verbindung wird hergestellt	Verbindung wird hergestellt	Ok
2	MQTT Verbindung	ESP32 verbindet sich am Start mit MQTT Server	Verbindung wird hergestellt	hergestellt	Ok
3	Sensorwert	Sensorwert wird ausgelesen	Sensorwert wird im Serial Monitor angezeigt	Sensorwert wird im Serial Monitor angezeigt	Ok
4	Sensor publish	Sensorwert wird gepublished	Sensorwert wird am richtigen Topic gepublished	Sensorwert wurde im falschen Topic gepublished, danach korrigiert	Ok

5	Schwellenwert setzen	Min/Max in Node-RED setzen und gepublished	MQTT Server erhält gesetzte Werte	MQTT Server erhält gesetzte Werte	Ok
6	Schwellenwerte empfangen	ESP32 subscribed Schwellenwerte	ESP32 zeigt die subscribed Werte an	ESP32 empfing die Werte nicht, danach Topic angepasst	Ok
7	OLED Anzeige	OLED Display zeigt die gewünschten Daten an	OLED Display zeigt Telemetriedaten und Status an	OLED Display zeigt Telemetriedaten und Status an	Ok
8	Ampel Anzeige	Ampel zeigt den Status an	Korrekte LED leuchtet auf.	Korrekte LED leuchtet auf.	Ok
9	MQTT reconnect	ESP32 verliert Verbindung zu MQTT Server	ESP32 verbindet sich automatisch wieder	ESP32 verbindet sich automatisch wieder	Ok
10	WLAN reconnect	ESP32 verliert Verbindung zu WLAN	ESP32 verbindet sich automatisch wieder	ESP32 hatte keine Funktion dafür, danach hinzugefügt	Ok
11	Telemetriedaten Auf Node-RED	Anzeige auf Node-RED möglich	Telemetriedaten werden auf Node-RED dargestellt	Telemetriedaten werden auf Node-RED dargestellt	Ok
12	Status auf Node-RED	Anzeige auf Node-RED möglich	Status wird auf Node-RED dargestellt	Status wurde nicht korrekt angezeigt, danach korrigiert	Ok
13	Telemetriedaten Verschlüsselung	Telemetriedaten werden verschlüsselt	MQTT Server hat falsche Werte	MQTT Server hat falsche Werte	Ok
14	Status Verschlüsselung	Status wird verschlüsselt	MQTT Server hat falsche Werte	MQTT Server hat falsche Werte	Ok
15	Telemetriedaten Entschlüsselung	Telemetriedaten werden entschlüsselt	Telemetriedaten sind auf Node-RED gleich wie auf ESP32	Telemetriedaten sind auf Node-RED gleich wie auf ESP32	Ok
16	Status Entschlüsselung	Status wird entschlüsselt	Status wird auf Node-RED gleich angezeigt wie auf ESP32	Status wurde verschlüsselt angezeigt, danach korrigiert	Ok

13 Erweiterungen

Um den kritischen Zustand der Pflanze offensichtlicher zu machen, könnte man mit einem Buzzer eine Art Alarmanlage erstellen, die ertönt, wenn der Stand zu trocken oder zu nass ist. Das würde man dann zum Beispiel gerade bei der Ampel hinzufügen und immer wenn die rote oder grüne LED angeht, geht auch der Buzzer für eine gewisse Zeit an.

Man könnte ebenfalls eine stärkere Verschlüsselung, wie zum Beispiel die Vignère-Verschlüsselung einbauen. Das bräuchte dann schon mehr, um sie zu knacken. Die Verschlüsselung würde dann im ESP32 stattfinden und die Entschlüsselung in Node-RED.

Man könnte ebenfalls gewisse Sicherheitsfeatures im ESP32 aktivieren.

Eine andere Idee wäre, dass man die Ampel an einen zweiten ESP32 anschliesst und man diese Ampel dann zum Beispiel an den Schreibtisch nimmt, um von dort den Stand anzusehen. Man müsste vom zweiten ESP32 dann einfach den Status abonnieren und entschlüsseln, um den richtigen Status zu erhalten.

Man könnte auch einen besseren Schlüssel für die Caesar-Verschlüsselung brauchen, also einer, der sich verändert. Praktisch könnte man hier zum Beispiel einen Sensorwert nehmen und diesen noch zusätzlich mit einer Rechnung verändern.

14 Fazit

Das Projekt wendet sich dem Ende zu und zum Abschluss reflektieren wir, was wir bei diesem Projekt gemacht haben. Der Auftrag besteht aus seiner Ausgangslage und zwei Automatisierungen. Wir konnten beide Automatisierungen realisieren, wobei wir die Erdfeuchtigkeit mittels Feuchtigkeitssensor ermitteln und die Statuswerte visuell darstellen. Die Schwellenwerte sind ebenfalls dynamisch konfigurierbar, wodurch sich die Statuswerte automatisch neu berechnen.

Wie in jedem Projekt hatten auch wir Herausforderungen. Am Anfang wollten wir den Status der Pflanze mit Hilfe von Farben im Gauge anzeigen lassen. Dafür hatte Lionel viel Zeit investiert. Ich war hierbei nicht viel besser, denn als es darum ging, die Schwellenwerte zu subscriben, habe ich zwei Stunden damit verbracht, nur um am Ende herauszufinden, dass ich die Topics gross statt klein geschrieben habe. Die Schwellenwerte waren sowieso tricky, denn wir mussten aus den Schwellenwerten einen Status errechnen und wieder publishen. Hierbei hatten wir Probleme, den Status korrekt auf Node-RED anzuzeigen. Zusätzlich wurden zuerst die Schwellenwerte auf Null gesetzt, wenn man sie löschte, wobei wir wollten, dass sie dann wieder auf den Standardwert gehen.

Im ganzen üK lernten wir viel über IoT, wie es aufgebaut ist und wie es funktioniert. Im Projekt konnten wir dann auch lernen, wie man IoT-Geräte praktisch umsetzt und was man alles zu beachten hat. Wir lernten das ganze Systemdenken von IoE und natürlich auch, wie man Topics sinnvoll erstellt und richtig subscribed.

Der wichtigste Punkt von einer Reflexion ist jeweils das, was man lernt. Am Anfang des Projektes hätten wir zum Beispiel potenzielle Fehlerquellen merken können, die leicht zu übersehen sind, wie die Topicnamen. Es würde ebenfalls Sinn machen, wenn wir früher Testkompetenzen festlegen und auch früher mit dem Testprotokoll anfangen. Denn jetzt haben wir das auf den Schluss aufgespart und zwar schon getestet, aber das nicht protokolliert. Das heisst, ich musste alles getestete nachprotokollieren. Ich glaube der Fehler geschah uns, da für mich ein Testprotokoll eher dafür da ist, um am Schluss zu testen, ob wirklich alles funktioniert, also nicht über den Verlauf des Projektes.