



Erdfeuchtigkeit einer Pflanze mittels IoT-Device messen



Dateiname: PflanzenTeam4_üK216_Projektdokumentation
Autor: Jevgenia, Abigail, Masumeh

Inhaltsverzeichnis

Einleitung	3
1.1 Sinn und Zweck	3
2 Informieren	4
2.1 Unser Auftrag	4
2.2 Ziel	4
2.3 Vorgehen	4
2.4 Anforderungen	4
2.5 Tools und Software	5
2.6 Hardware	6
3 Planen	8
3.1 Brainstorming	8
3.2 Tasklist	8
3.3 Zeitplan	9
4 Entscheiden	10
4.1 Einteilung	10
4.2 Welche Komponenten benötigen wir und welche nicht?	10
4.3 Neue Sensor/Aktuator-Entscheidung	10
4.4 Hinzufügen vom Feature	10
5 Realisieren	11
5.1 Design	11
5.1.1 Brainstorming	11
5.1.2 Flowchart	11
5.2 Setup	12
5.2.1 Repository klonen	12
5.2.2 To-Do Liste auf Notion	12
5.3 Implementieren	13
5.3.1 Code Abschnitte	13
5.3.2 Melodie Frequenz	13
5.3.3 Buzzer Output	13
5.3.4 Feuchtigkeitzzustand LED-Lampe	13
5.3.5 OLED-Display Werte	14
5.3.6 Feuchtigkeitssensor	14
5.3.7 Begegnete Probleme	14
5.3.8 MQTT	15
5.4 Node-RED	16
5.4.1 End Produkt	17
5.5 Kontrollieren	18
5.5.1 Testing	18

Erdfeuchtigkeit

Erdfeuchtigkeit

6	Auswerten	20
6.1	Timetable? Eingehalten ja nein?	Fehler! Textmarke nicht definiert.
6.2	Was ging nicht nach Plan	20
6.3	Self improvement.....	21
6.3.1	Abigail.....	21
6.3.2	Masumeh.....	21
6.3.3	Jevgenia	21

Quellen

Dossier:

00_modul-216-ku-getting-started_zh-usb-c_v1-6.pdf
 01_modul-216-ku-grundlagen-ioe-iot_v1-3.pdf
 02_modul-216-ku-esp32-node-red-mqtt_zh-usb-c_v1-7.pdf
 04_modul-216-ku-begleiter_zh-usb-c_v1-1.pdf

Internet:

<https://www.arduino.cc/>

YouTube:

https://www.youtube.com/watch?v=FdPYiK_t_Qo

Einleitung

1.1 Sinn und Zweck

Dieses Dokument enthält die Dokumentation des vorliegenden Projekts. Es beschreibt, wie das Projekt geplant, durchgeführt und getestet wurde. Es enthält die Entscheidungen und Fehler und Lösungen, die während dem Projekt entstanden sind und dokumentiert alles im Detail. Das Endprodukt sollte in der Lage sein, die Erdfeuchtigkeit zu messen und auszugeben, weder die Erde mehr, weniger oder kein Wasser mehr braucht.

Erdfeuchtigkeit

Erdfeuchtigkeit

2 Informieren

2.1 Unser Auftrag

Wir haben in der Gruppe den Auftrag erhalten, ein Gerät zu entwickeln, das die Erdfeuchtigkeit von Pflanzen misst und überwacht. Ziel dieses Projekts ist es, eine Lösung zu schaffen, die eine kontinuierliche Überwachung des Feuchtigkeitsniveaus ermöglicht und so verhindert, dass Pflanzen in Vergessenheit geraten und über längere Zeiträume nicht ausreichend bewässert werden. Das Gerät soll in der Lage sein, die Feuchtigkeit der Erde zu messen, diese Daten auszuwerten und sie anschliessend visuell darzustellen.

2.2 Ziel

Das Ziel des Projekts ist, ein System zu bauen, das die Feuchtigkeit der Erde misst und die Daten in Echtzeit anzeigt.

- Das System soll die Feuchtigkeit der Erde messen und die Ergebnisse auf einem Bildschirm anzeigen.
- Die Feuchtigkeitsdaten sollen sichtbar gemacht und Änderungen an einem MQTT-Topic gesendet werden.
- Mit Node-RED sollen wir festlegen, wann die Feuchtigkeit in Ordnung ist und den aktuellen Bewässerungsstatus anzeigen.
- Wir sollen sehen, wie die Feuchtigkeit im Laufe der Zeit verändert wird und diese Daten anzeigen können

2.3 Vorgehen

Wir haben uns zunächst einzeln mit der Aufgabenstellung und der Recherche beschäftigt, uns dann zusammengesetzt und darüber gesprochen, wodurch mehr Ideen entstanden sind. Es wurde besprochen, welche Hardware und Tools wir verwenden könnten oder in Frage kommen.

2.4 Anforderungen

Uns gestellte Anforderungen:

#	Kriterium	Objekt	Erfüllungsgrad	Maximale Punkte	Erreichte Punkte	Note
			Nicht vorhanden Viele Fehler Ungenügend Knapp genügend Gut Sehr gut			
Spezifikation, 10%						
1	Die IoT-Endgeräte können mittels Konfigurationsparametern konfiguriert werden.	Projektarbeit		2.5	0	1
2	Es wurde ein Verfahren zur strukturierten Konfiguration von IoT-Endgeräten angewendet (z.B. Checkliste).	Projektarbeit		2.5	0	1
Betrieb, 20%						
3	Die IoT-Endgeräte nutzen ein spezifisches, getrenntes Netzwerk.	Projektarbeit		2.5	0	1
4	Die IoT-Endgeräte nutzen Zugangsdaten in einer sicheren Art.	Projektarbeit		2.5	0	1
5	Es ist sichergestellt, dass schützenswerte Daten nicht direkt per IoT-Endgerät zugänglich sind.	Projektarbeit		2.5	0	1
6	Es ist sichergestellt, dass die IoT-Endgeräte über eine aktuelle Firmware bzw. Betriebssystem verfügen.	Projektarbeit		2.5	0	1
Einbindung, 45%						
7	Die IoT-Endgeräte sind in die bestehende IoT-Plattform eingebunden.	Projektarbeit		6	0	1
8	Die Daten der eingebundenen IoT-Endgeräte werden mittels IoT-Plattform verarbeitet.	Projektarbeit		6	0	1
9	Die IoT-Plattform ist in der Lage bi-direktional mit den IoT-Endgeräten zu kommunizieren.	Projektarbeit		5	0	1
10	Die IoT-Plattform erlaubt die Daten der IoT-Endgeräte zu filtern und anzupassen.	Projektarbeit		5	0	1
Testing und Dokumentation, 15%						
11	Die IoT-Endgeräte und die Einbindung der Geräte sind getestet.	Testing		2	0	1
12	Die Tests der IoT-Endgeräte sind dokumentiert.	Testing		2	0	1
13	Die Lösung (IoT-Plattform inklusive IoT-Endgeräte) ist dokumentiert.	Dokumentation		1	0	1
14	Die Dokumentation enthält mindestens eine Grafik, welche die Systemübersicht aller Komponenten zeigt.	Dokumentation		0.5	0	1
15	Die Dokumentation enthält mindestens eine Grafik, welche den Datenfluss aller Komponenten zeigt.	Dokumentation		0.5	0	1
16	Sind genügend qualitativ gute Unterlagen zum Fortführen der Arbeit vorhanden?	Dokumentation		0.5	0	1
17	Wie gut und wie ausführlich wurde die Ausgangslage, die Ziele und der Ausblick beschrieben?	Dokumentation		0.5	0	1
18	Wie gut und wie ausführlich wurden die technischen Unterlagen gepflegt?	Dokumentation		0.5	0	1
19	Entspricht der Gesamteindruck der Dokumentation den gängigen Erwartungen?	Dokumentation		0.5	0	1
Arbeitsweise, 10%						
20	Die Projektarbeit wurde selbstständig erarbeitet.	Arbeitsweise		1.5	0	1
21	Die Projektarbeit wurde mit hohem Engagement erarbeitet.	Arbeitsweise		1.5	0	1
22	Eine mehr oder weniger gleichverteilte Arbeitsaufteilung hat stattgefunden und ist nachvollziehbar.	Arbeitsweise		2	0	1
Endbearbeitung				50	0	1.00

Erdfeuchtigkeit

Erdfeuchtigkeit

Anforderungen vom Auftrag

Umsetzung gem. Bewertungsraster (siehe Dokument «Bewertung» in Microsoft Teams)

Messen des Wasserbestands / Erdfeuchtigkeit

Anzeige von Daten auf dem Device mittels einer passenden Ausgabekomponente

Status bei Änderung an MQTT-Topic «**zuerich/pflanzen/[name]/[xyz]**» übermitteln

Eingabe und Ausgabe von Daten mittels Node-RED

- Eingabe: Schwellwerte
- Ausgabe: Bewässerungszustand

IO-Matrix: Welche Eingaben und Ausgaben passieren beim Device und Node-RED?

	Device(s)	Node-RED
Eingabe	<ul style="list-style-type: none"> Messung eines passenden Umgebungswerts 	<ul style="list-style-type: none"> Schwellwerte (bis zu welchem Wert es «ok» ist)
Ausgabe	<ul style="list-style-type: none"> Hinweis auf Umgebungswert basierend auf konfigurierte Schwellwerte 	<ul style="list-style-type: none"> Genauer Umgebungswert und zeitlicher Verlauf

2.5 Tools und Software

Ressourcen zur Dokumentation

Word

- Wird verwendet, um die gesamte Projektdokumentation zu schreiben und zu formatieren.

Softwareressourcen

Arduino Entwicklungsumgebung

- Die Arduino IDE wird genutzt, um den Code für das ESP32-Board zu schreiben und zu testen. Bibliotheken werden für die Kommunikation mit den Sensoren und das Steuern der Aktuatoren benötigt werden.

Node-RED

- Node-RED wird verwendet, um die Daten aus dem System zu verarbeiten und zu visualisieren.

MQTT Explorer

- Ein Tool zur Überwachung und Verwaltung von MQTT-Nachrichten. MQTT Explorer hilft uns, die Kommunikation zwischen den Geräten und der IoT-Plattform zu testen und sicherzustellen, dass die Daten korrekt übertragen werden.

Draw.io

- Draw.io wird verwendet, um ein Flussdiagramm zu erstellen, das den Ablauf und die Logik des Codes visualisiert.

WLAN

- Das WLAN-Netzwerk wird für die Kommunikation zwischen den Geräten und zur Übertragung von Daten genutzt.

Notion

- Notion wird verwendet, um die Aufgaben im Projekt zu organisieren.







Google

- Google wird als Unterstützung, z.B. für den Aufbau der Hardware oder Geräteinformationen verwendet.

Erdfeuchtigkeit




2.6 Hardware

Komponenten:

Bezeichnung	Funktion	Anzahl	Bild
Steckplatine (Breadboard)	Board zur elektronischen Prototypentwicklung	1x	
ESP32-basierter Mikrokontroller	System-on-Chip (SoC)	1x	
0.91 OLED-Display	Display mit Auflösung 128x32 Pixel	1x	
GPIO Male-Male Kabel	Überbrückungskabel für die Steckplatine	6x	
GPIO Male-Female Kabel	Verlängerungskabel von Pin zu Steckplatine	10x	
Sensor V2.0.0	Boden-Feuchtigkeitssensor	1x	

Erdfeuchtigkeit

Erdfeuchtigkeit

USB-C Kabel	Stromversorgung / Datentransfer	1x	
RGB Ampel	LED Ampel, leuchtet wenn etwas passiert	1x	
Buzzer	Spielt Melodien aufgrund von Frequenzen	1x	

Erdfeuchtigkeit

Erdfeuchtigkeit

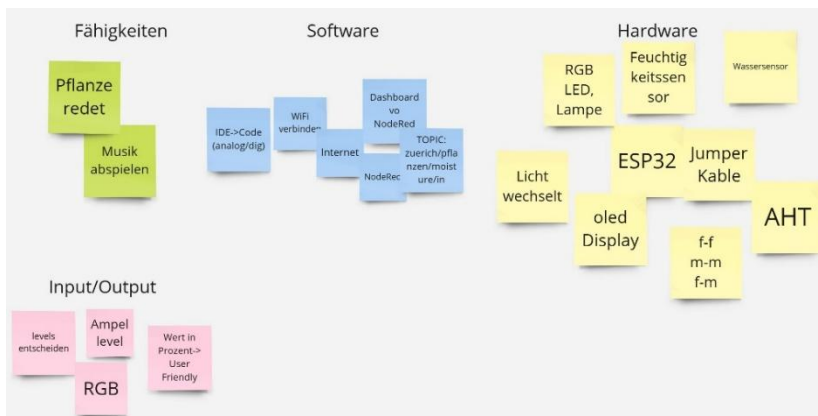
3 Planen

3.1 Brainstorming

Wir überlegten uns die wichtigsten Dinge, die wir für unsere Planungsphase benötigten, und notierten sie. So wollten wir sicherstellen, dass jeder zu Beginn seine Fragen klärt, damit alles reibungslos verläuft und niemand im Unklaren darüber ist, wer welche Aufgaben übernimmt.

Wir stellten uns unter anderem folgende Fragen:

- Welche Hardwarekomponenten werden wir benötigen?
- Wer übernimmt welche Aufgaben?
- Haben wir den Code bereits während des Experimentiertages geschrieben?
- Wie möchten wir das Produkt präsentieren?
- Welche Personas könnten von diesem Projekt profitieren, wenn es realisiert wird?
- Wie können wir das Dashboard erstellen



3.2 Tasklist

Tasklist						
Personen: Abigail(A), Masumeh(M), Jevgenia(J)						
Date: 11.12.2024						
Letzter Status (St): 13.10.2024						
Done						
In Progress						
Todo						
Time adjustment						
Titel	Start (2024)	Bis (2024)	Zeit (h)	Gebrauchte Zeit (h)	S	Person
Informieren Phase						
Anforderungen verstehen	11.12.	11.12.	30	15		A/M/J
Sensoren Recherche	11.12.	11.12.	60	30		M/A
Analog/Digital	11.12.	11.12.	30	30		M/A
Aktuatoren Recherche	11.12.	11.12.	45	45		J
NodeRed informieren	11.12.	11.12.	15	15		A/M/J
Bewertungsriterien verstehen	11.12.	11.12.	45	45		J
Planung						
Task List erstellen	11.12.	12.12.	45	35		M
Gruppen Einteilung	11.12.	11.12.	15	5		A/M/J
Brainstorming	11.12.	11.12.	30	20		A/M/J
Entscheidung						
Prioritäten setzen	12.12.	12.12.	10	10		A/M/J
Welche Hardware gebraucht?	12.12.	12.12.	10	10		A/M/J
Realisierungs Phase						
Repository erstellen	12.12.	12.12.	10	10		A
Documentation	12.12.	12.12.	120	70		J
Inform. Plan	11.12.	12.12.	60	60		J
Entscheidung	12.12.	12.12.	120	120		A/J
Dokumentation	12.12.	12.12.	60	60		A/M/J
Realisieren	12.12.	12.12.	60	60		M
Dokumentation	12.12.	12.12.	40	30		J
Control und Access	11.12.	12.12.	60	60		A
Aufbauen	11.12.	12.12.	60	60		A
NodeRed	11.12.	12.12.	240	240		A
Code schreiben	12.12.	12.12.	45	35		M
Kontrol Phase						
Test plan schreiben	12.12.	12.12.	30	45		J/A
Bug vom Code	12.12.	12.12.	15	5		A
Stop testing	12.12.	12.12.	30	30		A
Code finalisieren	12.12.	12.12.	120	120		A/M/J
Finalisieren	13.12.	13.12.	45	-		J
Dokumentation	13.12.	13.12.	45	-		A
Auswertung						
Self Reflection	13.12.	13.12.	30	-		A
Time management	13.12.	13.12.	30	-		A
Produkt Auswertung	13.12.	13.12.	5	-		offen
Code Auswertung	13.12.	13.12.	5	-		offen
Projektdokumentation	13.12.	13.12.	5	-		offen
Abgeben	13.12.	13.12.	5	-		offen

Um die Aufgaben grob festzuhalten, haben wir Start- und Endtermine für jede Aufgabe, um die Zeit zu verwalten und, wie lange jede Aufgabe dauern würde. Wir haben darauf geachtet, dass die Aufgaben in kleinere Teilaufgaben wie 30 Minuten oder 2 Stunden eingeteilt sind. Ebenso Aufgaben auf die Teammitglieder, dass jeder mit seinen Aufgaben vertraut, ist Arbeitsbelastung ausgewogen ist, sodass sich niemand überfordert fühlt. Wir erstellten Aufgaben in Notion und besprachen zu Beginn des Tages, was unsere Ziele für den Tag waren (z. B. bis zum Mittag). Dazwischen, sowie kurz vor Abschluss der jeweiligen Aufgaben, informierten wir die Teammitglieder über den aktuellen Stand.

3.3 Zeitplan

Tag [2024]	Tätigkeit	Zeit/Task [min]
Mittwoch, 11.12. @NY	Input zum Testing erhalten	60
	Einteilung der Gruppen und Themen	15
	Aufgabenstellung eigenständig lesen und im Team austauschen	30
	Alle Komponenten sammeln und Hardware aufbauen	35
	Ein Repository erstellen	10
	Code für den Feuchtigkeitssensor und AHT schreiben	60
	Node-RED erstellen	40
	MQTT-Verbindung herstellen	70
	Layout für die Dokumentation erstellen	30
	Testen, ob die Verbindung funktioniert	25
	Debuggen des Codes	35
	Projektdokumentation für die Informing-Phase schreiben	70
	Total	480
Donnerstag, 12.12. @NY	Code fertig geschrieben, AHT entfernt, OLED hinzugefügt	120
	Code debuggen und Produkt getestet	110
	Planning Phase schreiben	60
	Code für den Buzzer implementiert	60
	Dokumentation für die Decision Phase geschrieben	40
	Node-RED Dashboard User Interface verbessert	30
	Input für das Fachgespräch erhalten	25
	Total	445
Freitag, 13.12. @NY	Eigenständiges Lernen	180
	Projektdokumentation schreiben	180
	Projektdokumentation beenden und korrigieren	150
	Total	510

4 Entscheiden

In diesem Kapitel wird beschrieben, welche Entscheidungen während des Projekts getroffen wurden und aus welchen Gründen diese getroffen wurden. Die Entscheidungen wurden demokratisch getroffen und mündlich besprochen.

4.1 Einteilung

Zunächst haben wir uns entschieden, unsere Aufgaben aufzuteilen, damit nicht alle Personen gleichzeitig am gleichen Task arbeiten.

Die Einteilung ist hier zu finden:

Wer	Was
Abigail	Code schreiben
Masumeh	Hardware zusammenbauen Recherche durchführen
Jevgenia	Dokumentation schreiben Node-Red erstellen

4.2 Welche Komponenten benötigen wir und welche nicht?

Zu Beginn unseres Projekts haben wir überlegt, welche Sensoren und Aktuatoren wir benötigen könnten und welche nicht. Zunächst stellten wir fest, dass der Feuchtigkeitssensor für unser Projekt am wichtigsten war. Zusätzlich nahmen wir einen Temperatursensor und zwei Aktuatoren, ein Lämpchen und einen Buzzer. Schnell merkten wir jedoch, dass der Wassersensor nicht notwendig war und legten ihn wieder zurück. Letztlich entschieden wir uns, das Lämpchen und den Buzzer zu behalten.

4.3 Neue Sensor/Aktuator-Entscheidung

Im Laufe des Projekts haben wir festgestellt, dass wir nicht alle Sensoren und Aktuatoren, die wir ursprünglich ausgewählt hatten, benötigen. Deshalb entschieden wir uns, den Temperatursensor wegzulassen und das Lämpchen durch eine Ampel zu ersetzen. Der Grund für den Wechsel war, dass wir drei verschiedene Zustände anzeigen wollten: Zu viel Wasser, zu wenig Wasser und der ideale Wasserstand. Mit dem Lämpchen konnten wir nur zwei Zustände darstellen, weshalb wir uns entschieden, eine Ampel zu verwenden, um diese drei Zustände klar zu visualisieren. Der Zustand mit zu wenig Wasser wird durch rotes Licht angezeigt, der Zustand mit zu viel Wasser durch gelbes Licht und der ideale Zustand durch grünes Licht.

4.4 Hinzufügen vom Feature

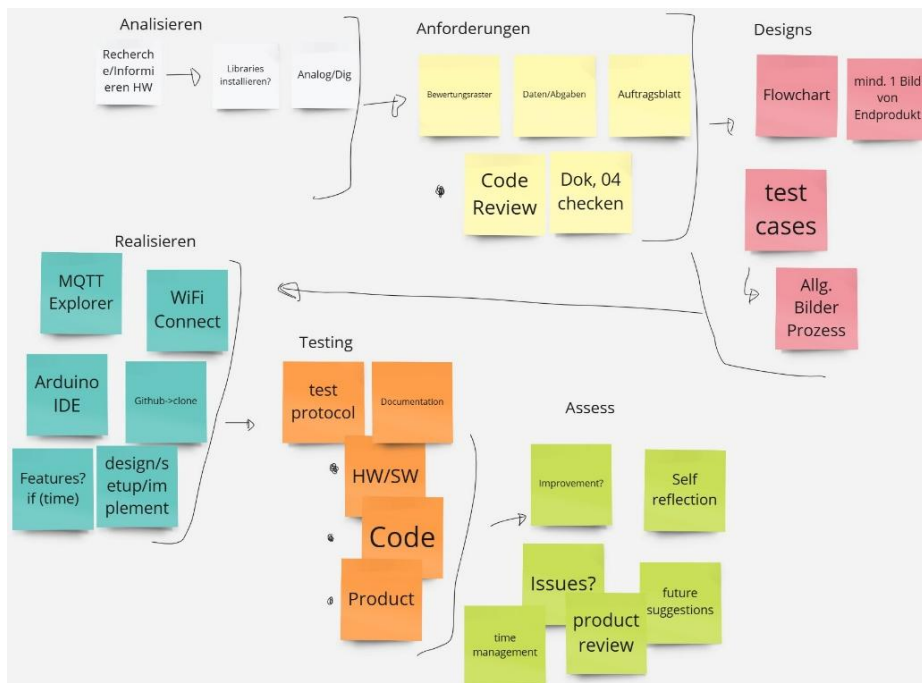
Wir haben uns vorgenommen, das IoT-Gerät so benutzerfreundlich wie möglich zu gestalten, vorausgesetzt, es bleibt genug Zeit. Unser Ziel war es, dass möglichst viele Personen das Konzept nutzen können und es so einfach wie möglich ist. Um auch Menschen zu unterstützen, die zum Beispiel nicht sehen können, haben wir einen Buzzer in die Hardware integriert. Der Buzzer gibt unterschiedliche Töne von sich: Bei zu wenig Wasser spielt er traurige Musik, bei zu viel Wasser ruhige Musik, damit der Benutzer weiss, dass er die Pflanze in Ruhe lassen kann, und bei optimaler Feuchtigkeit wird fröhliche Musik abgespielt.

5 Realisieren

5.1 Design

5.1.1 Brainstorming

Wir haben eine Skizze erstellt, um einen Überblick darüber zu haben, was in der Realisierungsphase enthalten sein soll.



5.1.2 Flowchart

Für unser Projekt haben wir mit drawio ein Flussdiagramm erstellt, das den gesamten Verlauf abbildet.

Die Bedeutung unseres Flussdiagramms

Klarheit

- Das Flussdiagramm macht die Programmierung klar und einfach zu verstehen, auch für diejenigen, die nicht direkt an der Programmierung beteiligt sind.

Planung

- Bevor wir mit der Programmierung beginnen, hilft uns das Flussdiagramm, alle wichtigen Schritte und Entscheidungen zu planen, so dass wir mögliche Probleme frühzeitig erkennen können.

Kommunikation

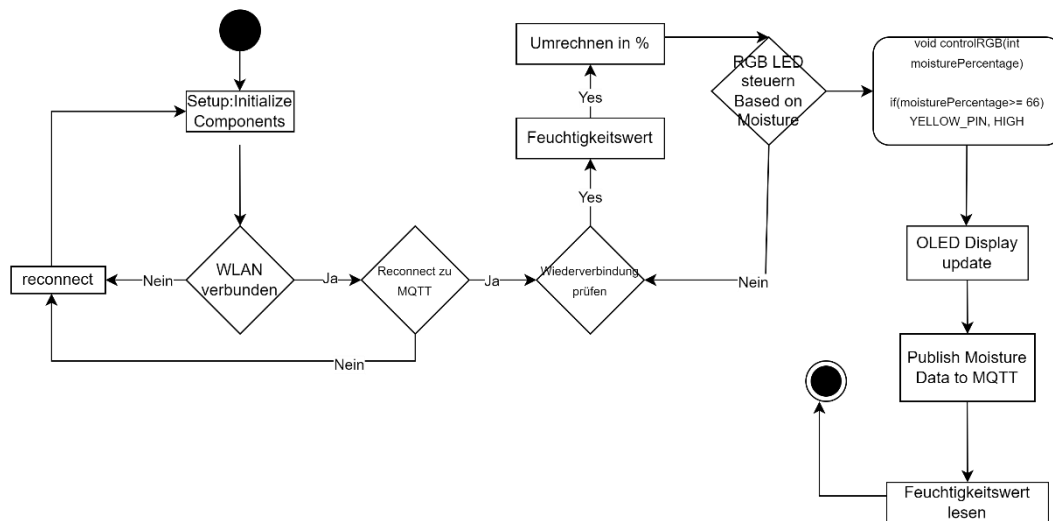
- Das Flussdiagramm dient als gemeinsame Grundlage für das gesamte Team.

Fehlervermeidung

- Durch die visuelle Darstellung des Spielablaufs können wir logische Fehler erkennen und beheben, bevor sie zu grösseren Problemen werden.

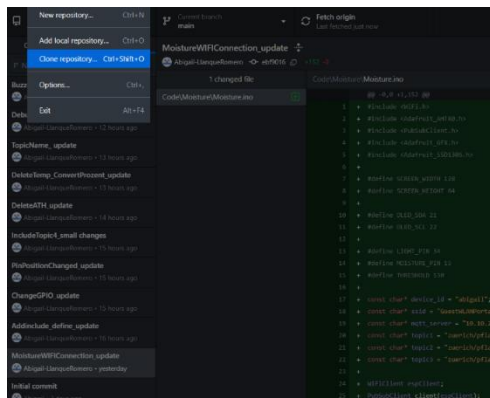
Erdfeuchtigkeit

Erdfeuchtigkeit



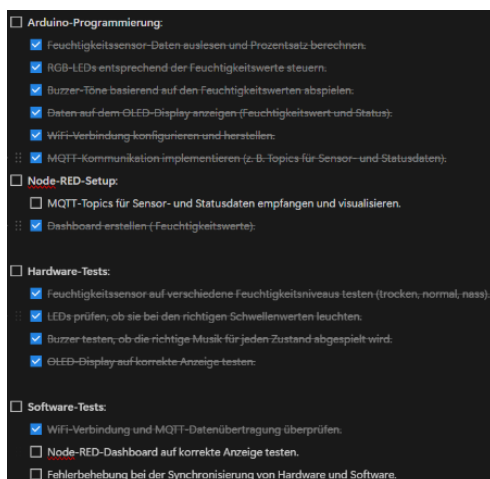
5.2 Setup

5.2.1 Repository klonen



Um ein GitHub-Repository zu klonen, wurde in GitHub Desktop über *File > Clone Repository* das gewünschte Repository ausgewählt. Nach dem Klonen konnten Änderungen vorgenommen werden, wobei darauf geachtet wurde, aussagekräftige Commit-Nachrichten zu schreiben. Dies hilft, bei zukünftiger Nutzung des Codes nachvollzogen werden kann, welche Änderungen vorgenommen wurden und wie Fehler behoben wurden.

5.2.2 To-Do Liste auf Notion



Wir markierten Aufgaben als erledigt, damit jedes Teammitglied, auch wenn es an einem anderen Ort arbeitet, dies mitbekommt. Wenn wir bemerkten, dass eine Aufgabe umfangreicher war, teilten wir sie in Teilaufgaben auf und schätzten den Zeitaufwand für jede Aufgabe.

5.3 Implementieren

5.3.1 Code Abschnitte

Für das Schreiben des Codes haben wir uns an den erstellten Flowchart gehalten sowie an den Experimentiertag, an dem wir bereits mit dem Code für das OLED-Display gearbeitet haben. Ausserdem hatten wir einige Beispielfcodes, beispielsweise für die Erstellung einer WiFi-Verbindung oder die Einbindung eines Sensors.

Um sicherzustellen, dass wir keine Zeit verschwenden, haben wir uns im Vorfeld informiert, ob wir alle benötigten Bibliotheken besitzen.

Im Folgenden werden einige Codeabschnitte erklärt:

5.3.2 Melodie Frequenz

```
int happyMelody[] = {523, 523, 587, 587, 523, 523, 523, 0, 523, 523, 587, 587, 523, 523, 523};
int happyDurations[] = {4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4};

int calmMelody[] = {440, 440, 440, 440, 0, 440, 440, 440, 440};
int calmDurations[] = {8, 8, 8, 8, 8, 8, 8, 8, 8};

int sadMelody[] = {220, 220, 220, 220, 220};
int sadDurations[] = {8, 8, 8, 8, 8};
```

Drei Arrays sind definiert, die verschiedene Melodien basierend auf der Feuchtigkeit der Pflanze auszugeben und die entsprechenden Dauerwerte für die Noten enthalten.

5.3.3 Buzzer Output

```
void playMelody(int* melody, int* durations, int size) {
    for (int i = 0; i < size; i++) {
        int noteDuration = 1000 / durations[i];
        tone(BUZZER_PIN, melody[i], noteDuration);
        int pauseBetweenNotes = noteDuration * 1.3;
        delay(pauseBetweenNotes);
        noTone(BUZZER_PIN);
    }
}
```

Spielt eine Melodie ab, indem jede Note nacheinander wird.

5.3.4 Feuchtigkeitszustand LED-Lampe

```
void controlRGB(int moisturePercentage) {
    if (moisturePercentage >= 66) {
        digitalWrite(REDA_PIN, LOW);
        digitalWrite(GREEN_PIN, LOW);
        digitalWrite(YELLOW_PIN, HIGH);
        playMelody(calmMelody, calmDurations, sizeof(calmDurations) / sizeof(int));
    } else if (moisturePercentage >= 45) {
        digitalWrite(REDA_PIN, LOW);
        digitalWrite(GREEN_PIN, HIGH);
        digitalWrite(YELLOW_PIN, LOW);
        playMelody(happyMelody, happyDurations, sizeof(happyDurations) / sizeof(int));
    } else {
        digitalWrite(REDA_PIN, HIGH);
        digitalWrite(GREEN_PIN, LOW);
        digitalWrite(YELLOW_PIN, LOW);
        playMelody(sadMelody, sadDurations, sizeof(sadDurations) / sizeof(int));
    }
}
```

Die drei Teile geben unterschiedliche Feuchtigkeitszustände an. Wenn der Wert zwischen 45 % und 65 % liegt, was den optimalen Zustand darstellt, leuchtet die grüne LED.

5.3.5 OLED-Display Werte

```
display.clearDisplay();
display.setCursor(0, 0);

display.print("Moisture: ");
display.print(moisture_percentage);
display.println(" %");

if (moisture_percentage >= 66) {
  display.println("Too Much Water");
} else if (moisture_percentage >= 45) {
  display.println("Perfect Water Level");
} else {
  display.println("Add Water");
}
display.display();
```

Die Feuchtigkeitswerte und Statusanzeigen werden auf dem OLED-Display angezeigt.

5.3.6 Feuchtigkeitssensor

```
int moisture_reading = analogRead(MOISTURE_PIN);
int moisture_percentage = map(moisture_reading, 0, 4095, 100, 0);
```

Der Feuchtigkeitswert wird vom Sensor abgerufen, indem der Pin, an dem der Sensor angeschlossen ist, ausgelesen und in einen Prozentsatz umgewandelt wird.

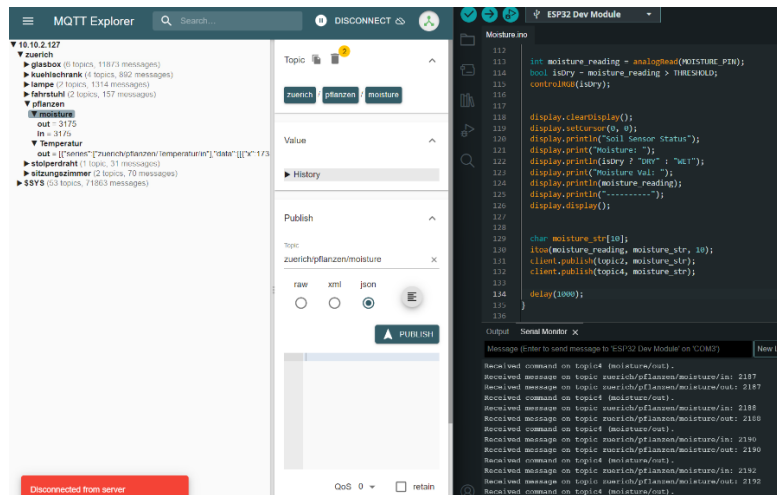
5.3.7 Begegnete Probleme

Wir hatten im Code Probleme, eine Verbindung zum AHT-Sensor herzustellen. Durch Troubleshooting fanden wir Schritt für Schritt heraus, was falsch war. Da dies jedoch nicht zu den Anforderungen des Auftrags gehörte, konzentrierten wir uns auf die wichtigsten Aufgaben. Zunächst hatten wir auch Schwierigkeiten mit der Internetverbindung. Wir verglichen den Code mit dem, den wir am Experimentiertag verwendet hatten, und entdeckten einen Copy-Paste-Fehler im WiFi-Code, den wir bereits genutzt hatten.

Beim Testen des Codes funktionierte dieser anfangs nicht. Wir überprüften daraufhin die Hardware und stellten fest, dass ein Kabel zerdrückt war. Nachdem wir es ausgetauscht hatten, funktionierte alles wieder und der Wert wurde korrekt angezeigt.

5.3.8 MQTT

Um uns mit dem MQTT-Server zu verbinden, erhielten wir zu Beginn des Projekts alle Informationen, die wir benötigten. Wir konnten uns mit dem MQTT-Server, der uns zur Verfügung gestellt wurde, über den Port 1883 und die IP-Adresse 10.10.2.127 verbinden. Da uns ausserdem der gesamte Code zur Verbindung mit dem MQTT-Server und Node-RED bereitgestellt wurde, war dies eine sichere Verbindung, weshalb wir uns keine Sorgen um die Sicherheit machen mussten.



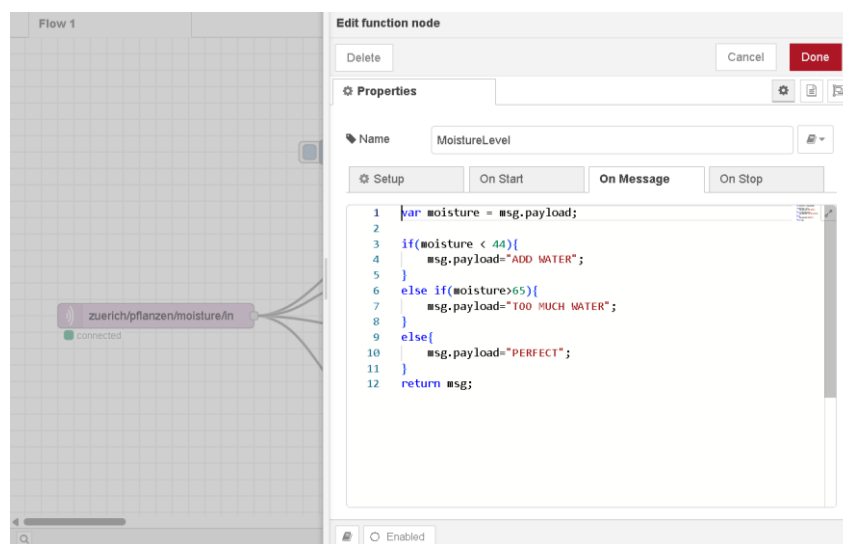
MQTT Server und Code NodeRed zeigen

5.4 Node-RED

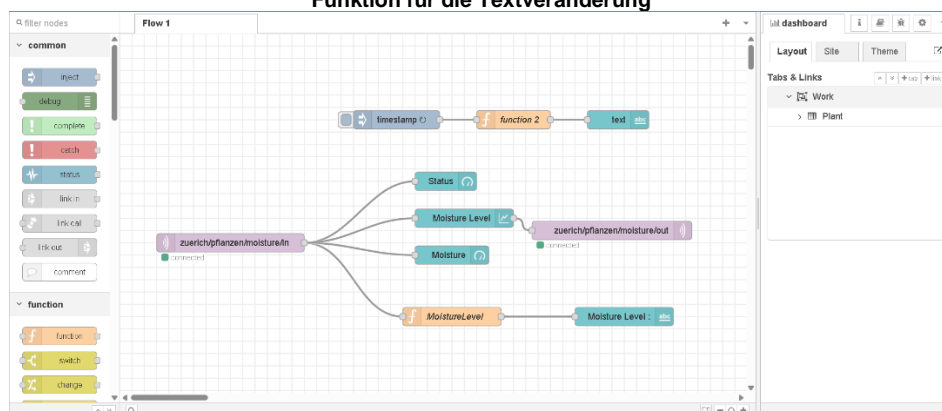
Zu Beginn des Projekts erhielten wir eine persönliche Node-RED-URL, über die wir unsere Daten anschaulich visualisieren konnten. Wir mussten uns mit dem Benutzernamen „admin“ und dem Passwort „kGzwSmuu“ einloggen. Dort konnten wir ein Topic definieren, das bei uns „zuerich/pflanzen/moisture/in“ hiess.

Node-RED diente dazu, uns die Möglichkeit zu geben, die Feuchtigkeit der Erde schön darzustellen. Dafür konnten wir verschiedene Charts, Gauges und Texte verwenden, um unser Node-RED-Dashboard zu gestalten.

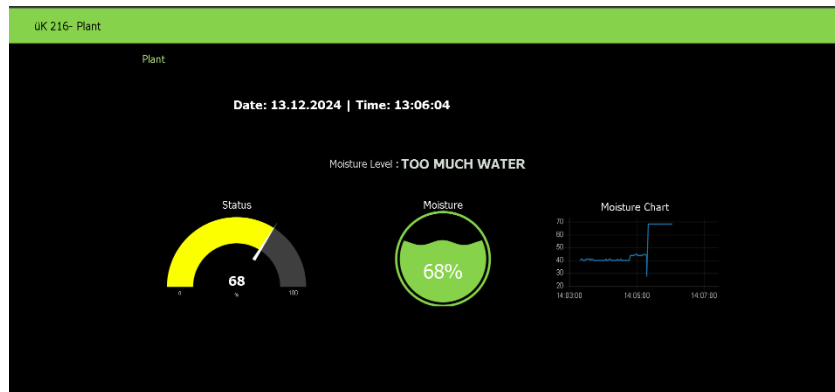
Zusätzlich haben wir uns entschieden, den Feuchtigkeitszustand der Erde als Text darzustellen, der anzeigt, ob der Zustand gut, zu hoch oder zu niedrig war. Dazu mussten wir eine Funktion erstellen, die einen direkten Bezug auf den Text hatte.



Funktion für die Textveränderung



Dashboard in Node-RED



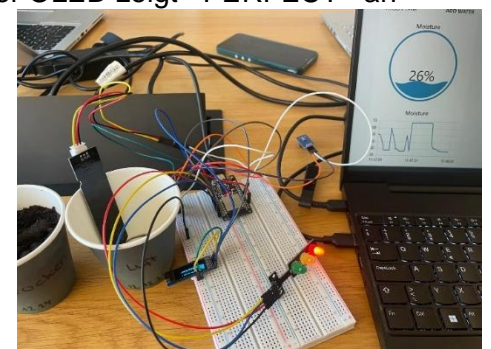
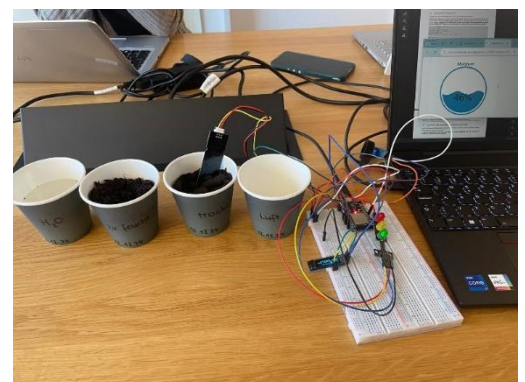
Endprodukt Node Red

5.4.1 End Produkt

Unser Endprodukt funktioniert nach dem Prinzip, dass ein Feuchtigkeitssensor die Feuchtigkeit der Erde misst und diese Werte an den Code abgibt. Die Werte gehen von 0 bis 4095, wobei 0 eine sehr hohe und 4095 gar keine Feuchtigkeit anzeigen. Wir haben in unserem Code diese Werte zu Prozentzahlen von 0-100% umgewandelt. Diese Werte wurden so definiert, dass wenn die Erde eine Feuchtigkeit von 0-45% hat, so hat die Pflanze zu wenig Wasser. Wenn sie die Werte von 45-65 zurückgibt, dann hat sie die perfekte Menge an Wasser und wenn sie die Werte 65-100 zurückgibt, so hat sie zu viel Wasser. Dies wird dann zuerst auf dem OLED-Display genauso angezeigt, wobei man die Prozentanzahl und eine kurze Beschreibung sieht. Zusätzlich werden die Resultate auch anhand von einer LED-Ampel angezeigt, wobei ein rotes Licht für zu wenig, das grüne Licht für perfekte Menge und das gelbe für zu viel Wasser steht. Als Zusatz haben wir auch einen Buzzer, welcher bestimmte Melodien abspielt, anhand von der Prozentanzahl. Gleichzeitig kann man dasselbe auf dem Node Red mitverfolgen. Mit 2 verschiedenen Gaugen und ein Chart sieht man die Prozentmenge der Feuchtigkeit (siehe bei Abbildung Endprodukt Node Red). Zudem erscheint auch ein Text, welcher den Zustand kurz beschreibt.

Auf den Bildern rechts sieht man im ersten Bild ein Beispiel für einen perfekten Wasserzustand. Das Lämpchen leuchtet grün, der OLED zeigt «PERFECT» an und das Node Red zeigt all dies auch an.

Auf dem zweiten Bild sieht man einen Wasserzustand mit zu viel Wasser. Dies erkennt man daran, dass die LED-Lampe gelb leuchtet, das OLED den Text «TOO MUCH WATER» zeigt und das NodeRed die zu starke Prozentanzahl anzeigt.



Erdfeuchtigkeit

Erdfeuchtigkeit

Das letzte Bild beschreibt einen Wasserstand, bei welchem die Erde zu wenig Wasser enthält. Die LED-Ampel leuchtet hier rot, der Display zeigt den Text «ADD MORE WATER» und das Node Red zeigt dasselbe an.

5.5 Kontrollieren

5.5.1 Testing

Hardware:

Was wird getestet?	Erwartetes Ergebnis	Ergebnis	OK/Nicht OK	Tester
Sind alle Kabel vorhanden?	Alle Kabel vorhanden und eingesteckt	Ja, alle Kabel sind vorhanden und eingesteckt	OK	Masumeh
Pins Überprüfen	Sind die Kabel mit den richtigen Pins verbunden (Analog/Digital)	Ja, Alles stimmt	OK	Masumeh
Wurden unterschiedliche Farben von Kabeln gebraucht?	Verschiedene Kabel wurden unterteilt verwendet	Es wurden verschiedene Farben gebraucht, jedoch nicht unterteilt (wurde versucht)	Nicht OK	Masumeh

Code:

Was wird getestet?	Erwartetes Ergebnis	Ergebnis	OK/Nicht ok	Tester
Gibt es Fehlermeldungen?	Nein, keine Fehlermeldungen	Es gibt keine Fehlermeldungen	OK	Abigail

Erdfeuchtigkeit

Erdfeuchtigkeit

- ☒ Funktionalität des Programmes ist gegeben und getestet
- ☒ Code ist vollständig kompilierbar ohne Warnings und Errors
- ☒ Include-Files enthalten eine Include-Guard
- ☐ ~~Code einer Zeile soll nicht länger als 80 Zeichen sein~~ *in*
- ☒ Code Convention gemäss diesem Dokument ist eingehalten, z.B. Struktur mit Klammern, eingerückt mit einheitlich 3-4 Leerschlägen
- ☒ Bezeichner und Dateien folgen der Naming Convention gemäss diesem Dokument z.B. keine Umlaute und Leerschläge *EN Sprache*
- ☒ Sprechende Namensgebung für Bezeichner (z.B. Variablen, Funktionen, Dateien)
- ☒ Abstand vor und nach Operatoren (Ausnahme: i++)
- ☒ Redundanz vermeiden, d.h. Codezeilen, die mehrfach gebraucht werden in Funktionen auslagern
- ☒ Kein «Toter Code» (heisst keine nicht verwendete Variablen, unerreichbarer oder auskommentierter Code)
- ☐ ~~Defensiver Programmierstil, d.h. Vergleiche mit Konstante links, Vergleichswert rechts des Vergleichs~~
- ☒ Konstanten wurden sinnvoll angewendet (keine redundanten String-Literals)
- ☒ Alle Variablen sind sinnvoll initialisiert
- ☐ ~~Globale Variablen sind verpönt, d.h. sie müssen sehr gut begründbar sein~~
- ☐ ~~Kommentare sind prägnant und unverzichtbar, also keine überflüssigen oder trivialen Kommentare~~ *nachträglich gemacht*
- ☐ ~~Funktionen sind übersichtlich und eher kurzgefasst, max 50 Zeilen~~
- ☒ Treffende Verwendung von Loops, weshalb while, for, do-while
- ☒ Wenn immer möglich positive Logik verwenden
- ☐ ~~Fallthrough in switch-Statements müssen kommentiert sein, wenn es nicht offensichtlich ist (z.B. gleiches Verhalten bei 'a' und 'A' wäre offensichtlich)~~

Node Red:

Was wird getestet?	Erwartetes Ergebnis	Ergebnis	OK/Nicht ok	Tester
Funktioniert die Funktion Wert < 45?	Wenn der Wert < 44 dann "ADD MORE WATER"	Gibt aus "ADD MORE WATER"	OK	Jevgenia
Funktioniert die Funktion Wert > 65?	Wenn der Wert > 65 dann "TOO MUCH WATER"	Gibt aus "TOO MUCH WATER"	OK	Jevgenia
Funktioniert die Funktion 45 < Wert < 65?	Wenn 45 < Wert < 65 dann "PERFECT"	Gibt aus "PERFECT"	OK	Jevgenia
Gauge: Wechselt diese die Farben?	Ja, sie wechselt die Farben, wenn ein neuer Wert erhalten wird	Farben werden gewechselt	OK	Jevgenia

Erdfeuchtigkeit

Erdfeuchtigkeit

Produkt:

Was wird getestet?	Erwartetes Ergebnis	Ergebnis	OK/Nicht ok	Tester
Leuchtet die Ampel bei unterschiedlichen Werten?	Abhängig vom Wert Rot/Geld/Grün	Rot: 0-44% (zu wenig Wasser) Grün: 45-65% (optimaler Wasserstand) Gelb: 66-100% (zu viel Wasser)	OK	Abigail
Zeigt das OLED-Display den Wert an und ob der Wert optimal ist?	Perfect Add Water Too much water Moisture Level: xy%	Add Water Moisture Level: 30%	OK	Abigail
Spielt der Buzzer bei unterschiedlichen Feuchtigkeitsstadien einen Ton ab?	Je nach Kategorie, drei unterschiedliche Melodien	Spielt drei verschiedenen Melodien nach Frequenz.	OK	Abigail
Misst der Feuchtigkeitssensor den aktuellen Zustand?	Individuelles Ergebnis	66%	OK	Abigail

6 Auswerten

6.1 Timetable

Wir haben uns jeden Tag neue Ziele gesetzt, um unsere Hauptziele zu erreichen. Glücklicherweise konnten wir einen Grossteil davon umsetzen. Allerdings hatten wir beim Einsetzen des AHT-Sensors Schwierigkeiten mit dem Code. Aufgrund der knappen Zeit entschieden wir uns schliesslich, ihn nicht zu verwenden.

6.2 Was ging nicht nach Plan

Da der Code den AHT-Sensor nicht erreichen konnte, bleibt der Code bei der Ausführung in dem Setup AHT loop stecken, weswegen wir keine Daten empfangen konnten. Dies hat uns einige Zeit gekostet und deswegen haben wir uns entschieden den Sensor komplett auszulassen.

Die Verwendung einer dreifarbigem LED hat uns nicht überzeugt, daher haben wir stattdessen eine Ampellampe verwendet, die deutlich sichtbarer war. Dabei haben wir jedoch etwas Zeit verloren.

6.3 Self improvement

6.3.1 Abigail

Ich war mehrheitlich für den Code zuständig. Als ich jedoch auf Probleme stiess und diese nicht eigenständig lösen konnte, wandte ich mich an meine Teamkollegen. Wir haben uns zusammengesetzt und gemeinsam überlegt, was es sein könnte. Besonders gut hat mir die Teamarbeit gefallen, da wir uns gegenseitig unterstützt haben und offen miteinander reden konnten. Wenn jemand eine Information brauchte, konnte man jederzeit ohne Hemmungen in der Gruppe nachfragen.

Ich denke, dass ich gut in der Gruppe mitarbeiten konnte und durch dieses Projekt einiges gelernt habe. Zum Beispiel möchte ich künftig beim Programmieren noch stärker auf Details achten und Flüchtigkeitsfehler vermeiden. Was uns definitiv geholfen hat, war die klare Zielsetzung, die wir uns gesteckt haben: vormittags bis mittags sowie nachmittags bis abends. Zusätzlich haben wir zwischendurch kleinere Zwischenziele gesetzt, wodurch wir einen besseren Überblick über die verbleibenden To-Do-Aufgaben hatten. Insgesamt bin ich froh, mein erlerntes Wissen direkt und praktisch anwenden zu können ebenso auf unser gelungenes Produkt im Team.

6.3.2 Masumeh

Ich war froh, meinen Teamkameraden bei auftretenden Problemen helfen zu können.

Probleme, die im Team auftraten, wurden gemeinsam besprochen, dabei wurden verschiedene Ideen eingebracht und diskutiert. Diese Ideen wurden anschliessend anhand von Kriterien wie Zeitaufwand und Schwierigkeitsgrad bewertet und entweder angenommen oder abgelehnt.

Ich war für die Kabelverbindungen und die Hardwarekomponenten zuständig und habe ausserdem die Benutzeroberfläche von Node-RED optisch verbessert. Ich habe dazu recherchiert und geplant, ein paar Bilder einzufügen. Allerdings scheiterte dies an der fehlenden Media-Node. Ansonsten lief alles gut.

6.3.3 Jevgenia

Mir hat die Teamarbeit während des Projekts sehr gefallen. Ich fand es gut, dass wir uns die grössten Aufgaben und Verantwortungen zuerst eingeteilt hatten. Meiner Meinung nach fand ich es übersichtlicher, da jede Person immer wusste, was sie tun konnte, in Fällen, wo es keine Gruppenarbeit benötigte. Meine Aufgabe war es, das Layout für die Dokumentation zu erstellen und für das Node Red zu achten. Bei der Dokumentation habe ich gelernt, wie ich am einfachsten eine Dokumentation aufteilen kann. Beim Node Red hatte ich schon ein wenig Vorwissen von dem Experiment Tag, welches mir geholfen hat, das Grundgerüst aufzubauen. Zusätzlich haben Masumeh und Ich uns das Ziel vorgenommen, das Node Red und das Layout ein wenig anzupassen und verschönern, wobei ich vieles durch Recherchen herausgefunden hatte. Im Ganzen hat mich das Projekt sehr gefallen und die Teamarbeit funktionierte sehr gut, was zu einem guten Ergebnis führte.

Ich nehme mir zudem vor an nächsten Projekten mit demselben Prinzip vorzugehen und der zukünftigen Gruppe diese Ideen der Aufteilung vorzuschlagen.

Was ich besser machen könnte in der Zukunft, wäre es versuchen bei einem Punkt weiterzuarbeiten, anstatt zum nächsten weiterzufahren, weil ich nicht mehr weiterweiss. Was ich mir hier Vornehme ist eigentlich bei einem Thema dranbleiben und solange dran arbeiten, bis ich es verstehe.