Glasbox Dokumentation

üK 216

Dokumentinformationen

Dateiname: Glasbox_doku_sujan_noe_ramon
Speicherdatum: 05.12.2025

Autoreninformationen

Autoren: Ramon Rosenberg, Sujan Suthakaran, Noe Eberli

Inhalt

Abbildungsverzeichnis

Einleitung

In diesem Dokument werden wir unseren Arbeitsprozess Dokumentieren und erklären. Wir werden erklären, was wir gemacht haben und wieso wir es so gemacht haben. Hier kann man die verschiedenen Arbeitsschritte verfolgen und nachschauen, wie unser Projekt entstanden ist.

Bei unserem Projekt handelt es sich um die Kontrolle über den Luftzustand einer Glasbox, es geht darum zu entscheiden, wann es zu heiss ist oder wann es zu kalt ist. Da die Luftfeuchtigkeit auch eine Rolle spielt haben wir das auch noch mit reingenommen.

Wir finden es wichtig, dass solche Dinge gemessen werden, da es auch Gesundheit schädlich sein kann an einem zu trockenen Ort zu sein.

Vorbereitung

Als aller erstes haben wir eine Todo-Liste erstellt, um immer auf dem aktuellen Stand zu sein. Wir haben die verschiedenen Aufgaben nach Wichtigkeit geordnet und dazu geschrieben, wer was macht. Dazu haben wir alle Dokumente dort abgelegt die wir im Verlauf von unserem Projekt brauchen können.

Ein Teil der Vorbereitung war ebenfalls das wir abgemacht haben, wer was macht. Da haben wir uns entschieden, dass Ramon die gestalterischen Dinge wie das UI, die Doku oder die Präsentation macht. Sujan und Noe haben sich mehr um das technische gesorgt. Was auch erklärt wieso Ramon nicht auf Git Hub commitet hat.

Name	Status	Priority	Zuständig	Done
Dokumentation schreiben	In progress	high	Ramon, Sujan, Noe	Done
Präsentation machen	In progress	mid	Ramon, Sujan	Done
Präsentation üben	Not started	mid	Noe, Sujan, Ramon	Done
Visualisieren Zusatz	In progress	low	Ramon	Done
Luftfeuchtigkeit messen	Done	high	Noe	Done
Piep Sound wenn es die Temperatur grenze überschreitet.	Done	high	Noe	Done
Temperatur beobachten	Done	mid	Noe,Sujan	Done
Material besorgen	Done	high	Noe, Sujan	Done
Temperatur visualisieren	Done	mid	Ramon	Done
Schwellwert konfigurieren	Done	high	Ramon	Done
Nachricht senden wenn Temperatur zu heiss	Done	mid	Ramon, Sujan	Done
Display mit Temperatur ansteuern	Done	high	Noe	Done
+ Neue Seite				

Abbildung 1: ToDo Liste in Notion

Planung

Unsere Idee war es, dass wir ein Messgerät bauen, welches die Temperatur misst und ab einer gewissen eine Warnung herausgibt, die Nachricht sollte beinhalten, dass es zu heiss oder die Luftfeuchtigkeit zu hoch ist.

Ausserdem wollten wir eine Anzeige, die mit Hilfe von Smileys anzeigt, wie der jetzige Zustand ist. Wir wollten ein einfaches und benutzerfreundliches UI machen, damit niemand durch ein überkompliziertes UI verwirrt wird. Ein kleines Display sollte ebenfalls mit dem ESP32 verbunden sein und konstant die aktuelle Temperatur anzeigen.

Schutzmassnahmen haben wir bei unserem Projekt keine implementiert, da wir Temperaturdaten des Büro als nicht schützenswerte Daten eingestuft haben. Mögliche Massnahmen wären jedoch zum Beispiel Ende zu Ende Verschlüsselung oder SSL.

In der Planung haben wir weiterhin unser Notion Planungsboard verwendet, dieses haben wir konstant weiter ausgefüllt und sind so auf dem neuesten Stand unsere Aufgaben geblieben.

Als weitere Vorbereitung haben wir ein Flowchart gemacht, um unsere Anfangsidee grob auf Papier zu bringen, leider ist dieses Dokument nicht unser Endprodukt und sehr simpel gehalten. Es ist aber trotzdem interessant zu sehen, wie unsere Idee sich verändert hat

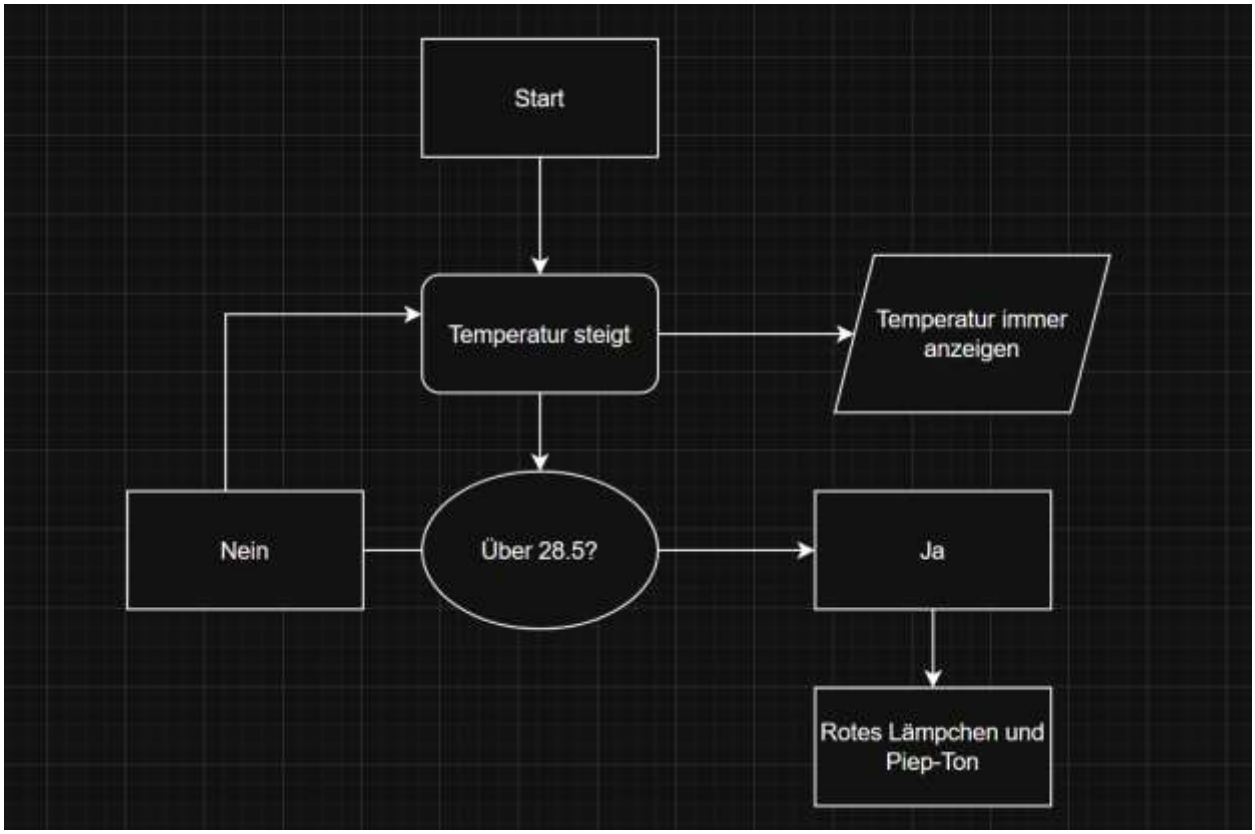

Abbildung 2 Flowchart

Netzwerkplan:

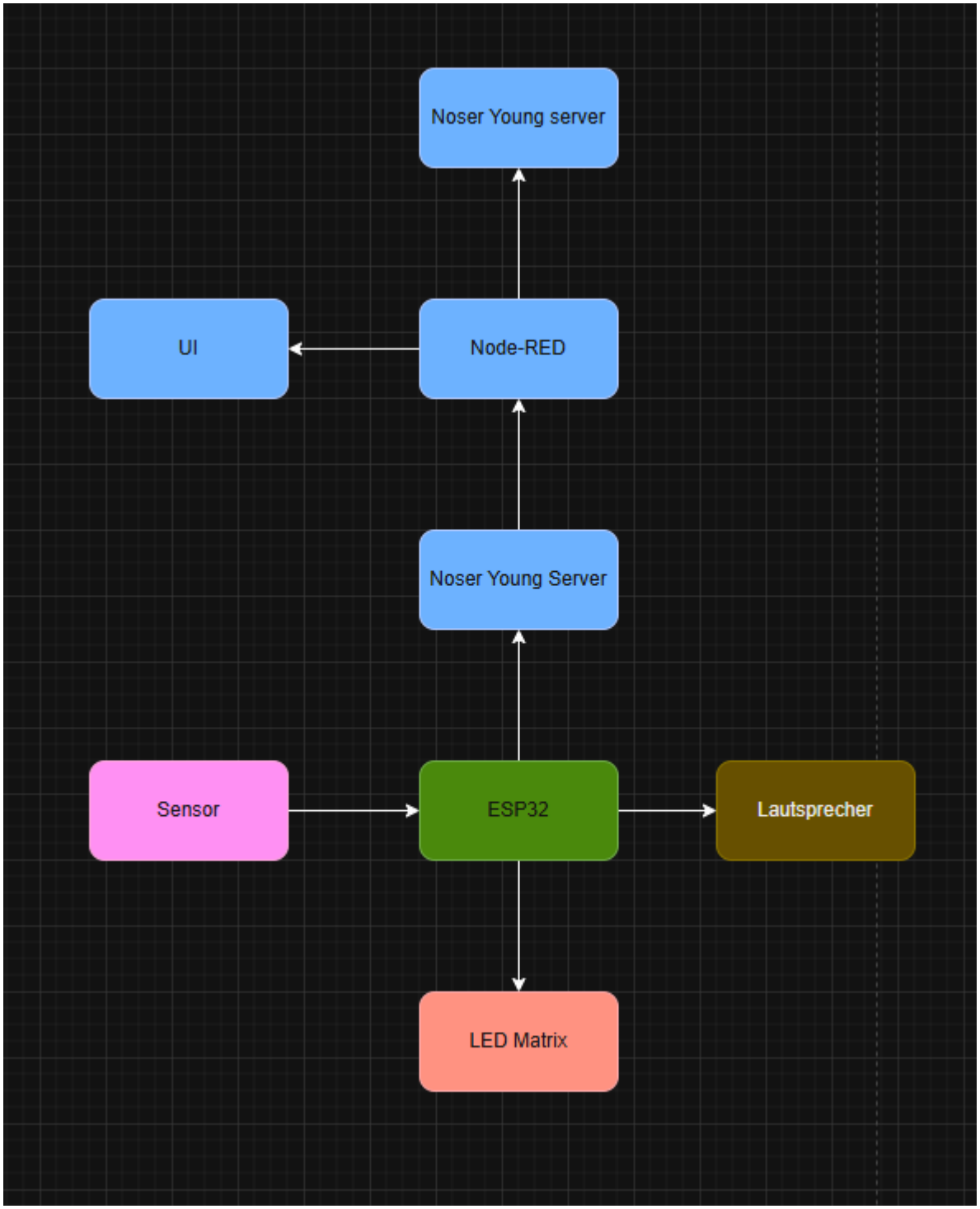

Abbildung 3 Netzwerkplan

Realisierung

Als erstes haben wir alle Materialien besorgt, die wir für das Projekt benötigen.

Folgend haben wir eine Materialliste erstellt, welche alles beinhaltet was wir brauchen.

Materialien:

Typ	Name	Anzahl
Sonstige	ESP-32	1
Sonstige	Breadboard	1
Sonstige	F-M Kabel	5
Sonstige	M-M Kabel	6
Sensor	Temperatur und Feuchtigkeit Sensor	1
Aktuator	7 Segment Matrix	1
Aktuator	Lautsprecher	1

Abbildung 4 Materialliste

Zuerst haben wir den ESP32 mit dem Breadboard verbunden. Den ESP32 mit dem Sensor zu verbinden war nicht besonders schwierig da wir das Pinout-Modell im Internet gefunden hatten. Danach konnten die wir die Temperatur beobachten. Bei diesem Vorgang hatten wir keine Probleme.

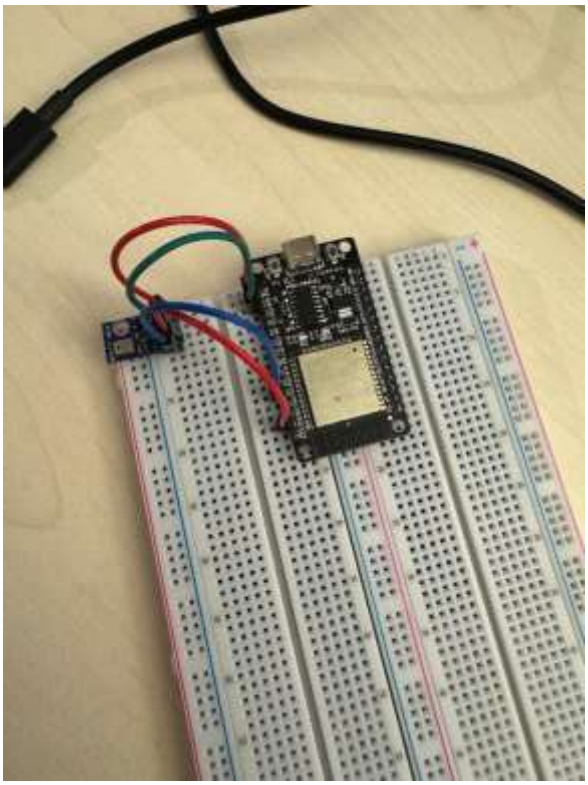

Abbildung 5 Breadboard mit ESP32 und Temperatursensor

Nachdem wir fertig waren mit dem Installieren des Sensors, wollten wir mit einem Display, die Temperatur anzeigen lassen.

Wir hatten sehr grosse Schwierigkeiten bei der Installation des Displays, weil unser Code ständig Fehler enthielt und wir das Problem sogar mit Hilfe von ChatGPT nicht lösen konnten. Nach mehreren Versuchen gelang es uns schliesslich doch.

Als wir unseren Code jedoch ausprobieren wollten, stellte sich heraus, dass unser Display defekt war und überhaupt nicht funktionierte. Während wir unser Problem zu lösen versuchten, hat Ramon in dieser Zeit ein UI erstellt, dass die Temperatur und die Luftfeuchtigkeit anzeigen kann.

Wir haben zum Glück ein neues Display gefunden und nun funktioniert es Einwand frei. Am Schluss haben wir eine Funktion eingebaut, dass wenn es zu heiss wird, dass man eine Nachricht erhält in dem steht man sollte die Glasbox lüften, weil es zu wenig Luftfeuchtigkeit hat oder es zu Heiss ist. Zusätzlich zum Display haben wir einen kleinen Lautsprecher eingebaut, dieser ist dazu da, dass gewarnt wird, wenn es im Raum zu heiss ist.

Der Lautsprecher lässt ab einer Temperatur von 28.5°C einen lauten pfeif Ton heraus welcher dazu bewegen sollte, dass man das Zimmer lüftet.

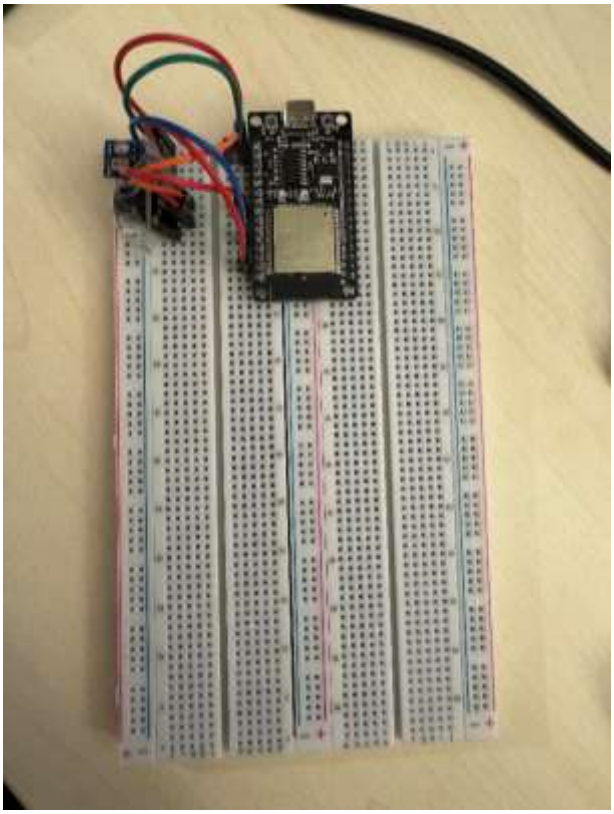

Abbildung 6 Breadboard mit ESP32, Temperatursensor und Lautsprecher

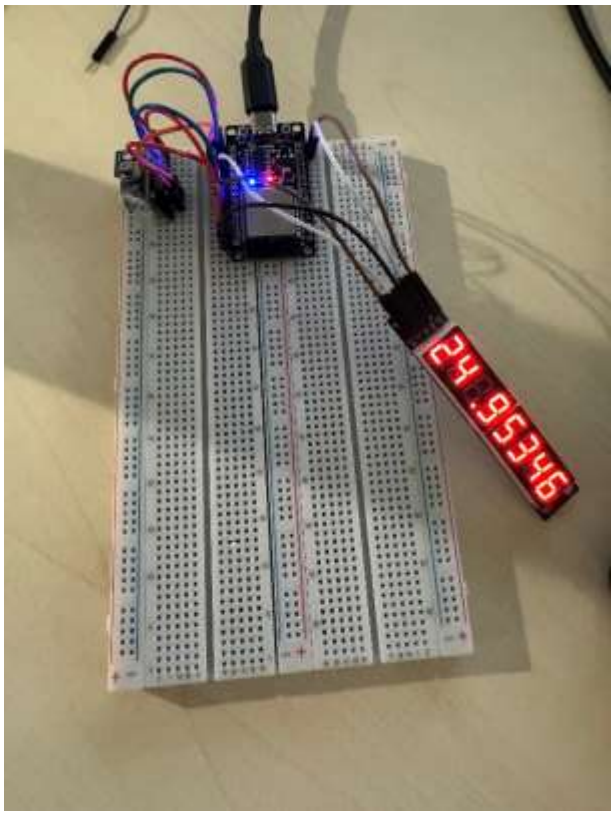

Abbildung 7 Breadboard mit ESP32, Temperatursensor, Lautsprecher und sieben Segment Matrix

Einen Teil unseres Programmes kommt von Node-RED, wir haben Node Red genutzt, um ein UI zu erstellen und die Zahlen zu runden. Alle Blauen und Gelben Nodes wurden durch Ramon erstellt und sind für die verschiedenen UI Elemente Zuständig.

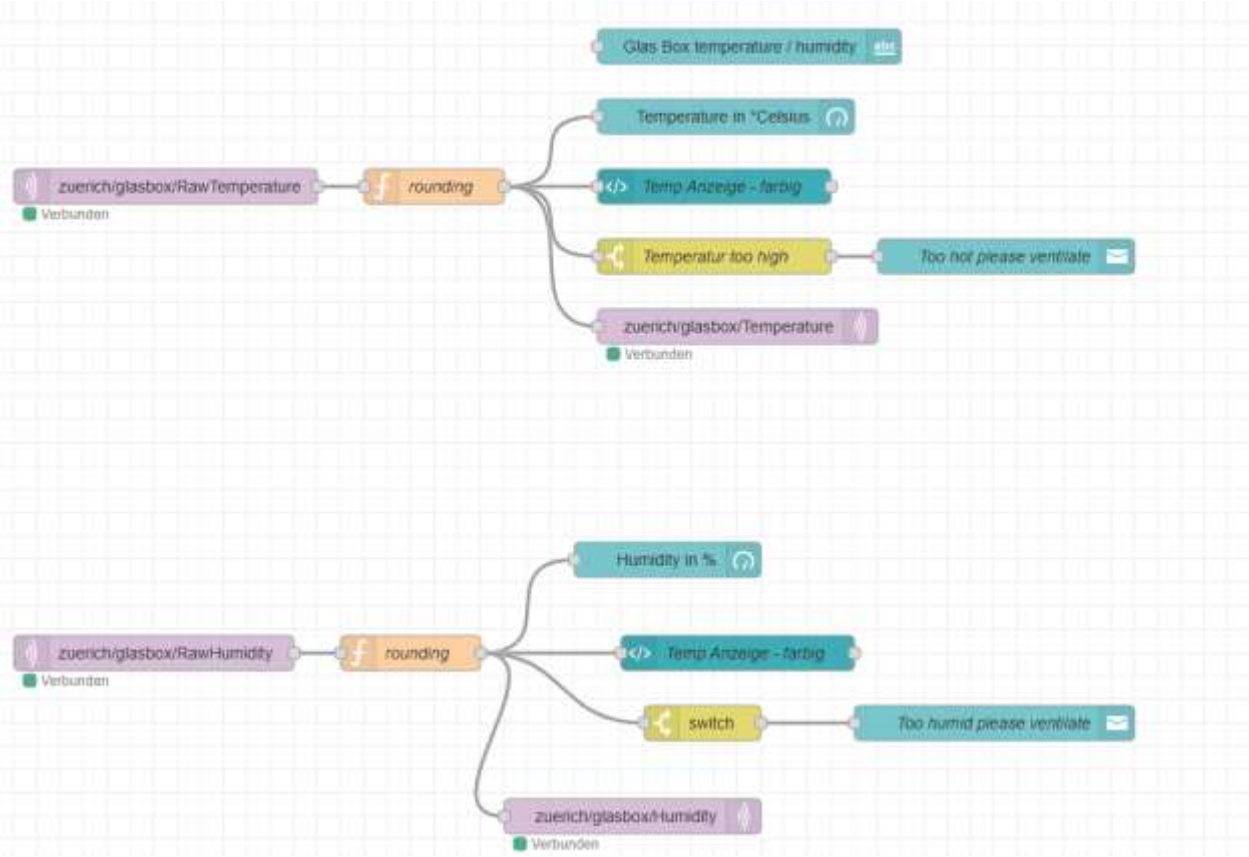

Abbildung 8 Node-RED des Projektes

UI erstellen (Ramon)

Hier hatte ich am Anfang keine Idee, was ich tun soll, da ich das vorher noch nie gemacht habe. Dann kam ich während der Planung auf die Idee es mit Smileys zu machen die aufgrund des Zustandes verschieden Glücklich sind.

Da ich noch nie Java programmiert habe musste ich ChatGPT fragen, wie man so etwas machen konnte. Ich habe einen Code bekommen und ihn in ein UI-Template Node eingesetzt. Am Anfang hatte ich ein Problem mit der Darstellung, da die Darstellung nicht im Vollbildmodus angezeigt wurde. Man musste scrollen, um die ganze Grafik zu sehen.

Nachdem ich das repariert habe indem ich das «Spacing» der Grafik verändert habe ist mir aufgefallen, dass zwei Statements auf einmal angezeigt werden. Dafür musste ich in den Code gehen und ihn verstehen.

```
// -------------------------
// ZONE 2: 20–26°C
// -------------------------
else if (temp >= 20 && temp <= 26) {
    box.style.background = "#A5D6A7";  // grün
    box.style.color = "#000";
    box.innerHTML = "😊<br><div style='font-size:24px;'>great!</div>";
}
```

Abbildung 9 Code in Node-Red um lachenden Smiley mit grünem Hintergrund zu machen.

Zum Glück hat das nicht sehr lange gedauert den Code zu verstehen, da mir mein mittelmässiges C-Verständnis in die Karten gespielt hat.

Um das Problem mit den zwei Statements zu lösen, musste ich die Vorzeichen z.B. «>, <, >=» ändern. Weil die falsch gesetzt wurden, wurden mehrere Statements gleichzeitig angezeigt. Da zu diesem Zeitpunkt nur der Smiley angezeigt wurde wollte ich noch eine Hintergrundfarbe, die sich automatisch ändert, wenn sich die Temperatur ändert.

Dafür musste ich wieder AI zurate ziehen. Nachdem ich den farbigen Hintergrund gemacht hatte, wollte ich, dass das ganze UI die Farbe gleichzeitig ändert.

Leider hatte ich keine Ahnung ich es machen sollte und habe verschiedene Ausbildner gefragt. Mir wurde gesagt, dass das sehr kompliziert sei und ich es lassen sollte. Darum habe ich mir vorgenommen zuerst den Rest zumachen und danach, wenn ich noch Zeit habe das zu versuchen.

Jetzt hatte ich ein UI, dass mit Farben und Smileys den Stand der Zimmer Temperatur anzeigt.

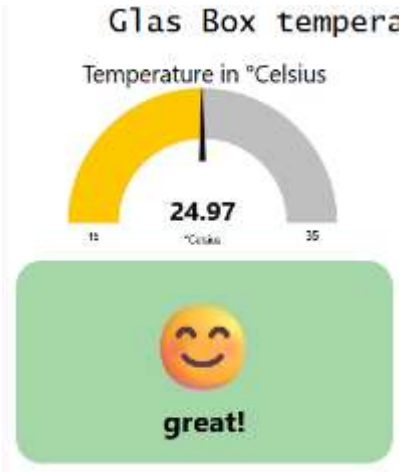

Abbildung 10 UI Objekt zu Temperatur

Anschliessend wollten wir, dass wenn es zu heiss wird, eine Nachricht aufpoppt, dass es zu heiss ist.

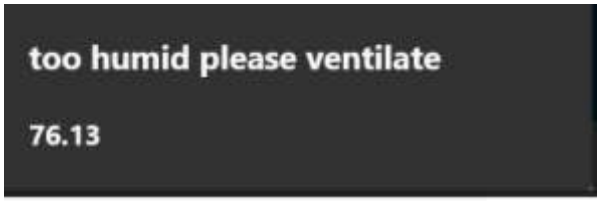

Abbildung 11 In diesem Beispiel Nachricht bei zu viel Luftfeuchtigkeit

Hier habe ich ein Switch-Node gebraucht, um eine Bedingung zu stellen, dass nur an einem bestimmten Punkt genau das passieren sollte. Hierbei hatte ich nicht sehr viele Probleme, da ich mich zu diesem Zeitpunkt immer besser in Node-Red auskannte. Danach musste ich das gleiche UI beim Luftfeuchtigkeit Tab machen. Das war nicht allzu kompliziert, da ich nur ein paar Zahlen anpassen musste.

Code

Der Code in Arduino IDE haben Sujan und ich (Noe) grösstenteils zusammengeschrieben. Beim Code der Sieben Segment Matrix hatten wir etwas Probleme, da wir das Display nicht dazu bringen konnten etwas anzuzeigen, wir haben deswegen eine längere Zeit damit verschwendet eine Lösung dafür zu finden.

Nach einer längeren Zeit sind wir jedoch darauf gekommen, dass es möglicherweise auch ein Hardware-Problem sein könnte. Diese Vermutung hatte sich dann auch als korrekt herausgestellt.

Als wir unser Display ausgetauscht hatten, waren alle unsere Probleme erledigt. Danach konnten wir endlich die Aktuelle Temperatur grafisch darstellen. Ganz oben im Code mussten wir definieren mit welchem Host wir uns verbinden wollen. In unserem Fall war das der 10.10.2.127.

Den Code haben wir in C++ geschrieben, was wir von Anfang an gut konnten, da wir die Programmiersprache C lange angeschaut hatten.

Da wir vermehrt fehlerhafte Daten übermittelt bekamen, haben wir uns dazu entschieden in Arduino IDE Meldungen einzubauen, welche immer dann erscheinen, wenn wir Daten versenden. Diese Massnahme ergriffen wir, weil wir so sicherstellen konnten, ob die Daten tatsächlich von uns hochgeladen wurden oder von anderen Gruppen.

```
LedControl display = LedControl(DIN, CLK, CS, 1);

void setup() {
  Serial.begin(115200);
  display.begin(15);
  display.clear();
  setup_sensor();
  setup_wifi();
  client.setURI(server);
  client.setMqttClientName(client_id);
  client.loopStart();
  pinMode(A, OUTPUT);
}

void displayString(const char* s) {
  uint16_t len = (uint16_t)strlen(s);

  for (uint16_t i = 0; i < len; i++) {
    display.setChar(0, 8 - i - 1, s[i], false);
  }
  Serial.println(s);
}

void loop() {
  sensors_event_t humidity, temp;
  aht.getEvent(&humidity, &temp);

  float temperature = temp.temperature;
  std::string msg = std::to_string(temperature);

  float hum = humidity.relative_humidity;
  std::string teststring = std::to_string(hum);
  client.publish("zuerich/glasbox/Rauhumidity", teststring);

  Serial.print(msg.c_str());
  Serial.print("      ");
  Serial.println(temperature);
  client.publish(pub_topic, msg);
  if (temperature > 28.5){
    digitalWrite(A, HIGH);
  }
  else {
    digitalWrite(A, LOW);
  }
  delay(1000);
  displayString(msg.c_str());
}
```

Abbildung 12 Codebeispiel

Testprotokoll

Testname	Was	Wie	Wer	erreicht
UI-Übereinstimmung	UI zeigt korrekte Temperatur an	Temperatursensor erhitzen und abgleichen mit mqtt.	Ramon	ja
Display Temperaturanzeige	Display zeigt ungerundete Temperatur konstant live an	Temperatursensor erhitzen und sieben Segment Matrix beobachten	Noe	ja
Alarm	Alarm ertönt wen Temperatur von 28.5°C überschritten wird	Temperatursensor auf 28.5°C	Noe	ja
UI-Ausgabe	Wird das richtige Emoji mit der richtigen Farbe angezeigt?	Erhitzen und schauen, ob sich das Emoji plus Farbe verändert.	Ramon	Ja
UI-Ausgabe	Wird die Nachricht bei Überschreiten des Schwellwerst korrekt ausgegeben?	Bis an den Schwellwert erhitzen.	Noe, Ramon	Ja

Abbildung 13 Testprotokoll