CO²-Projekt des Bürgernetzes Dillingen e.V.

aus dem LORAWAN Projekt übernommen zum Weiterführen

http://esp8266-server.de/CO2Ampel.html

etwas Theroie zu CO2 Sensoren:
https://www.msxfaq.de/sonst/bastelbude/co2_sensor.htm

etwas zur Orientierung (um ein Gefühl für die Werte zu bekommen)
https://frida-kahlo-schule.lvr.de/media/lvrfridakahloschule/aktuelles/corona/Lueften_in_Klassenraeumen_Empfehlungen_LVR_Dezernat_12.40_Arbeitssicherheit.pdf

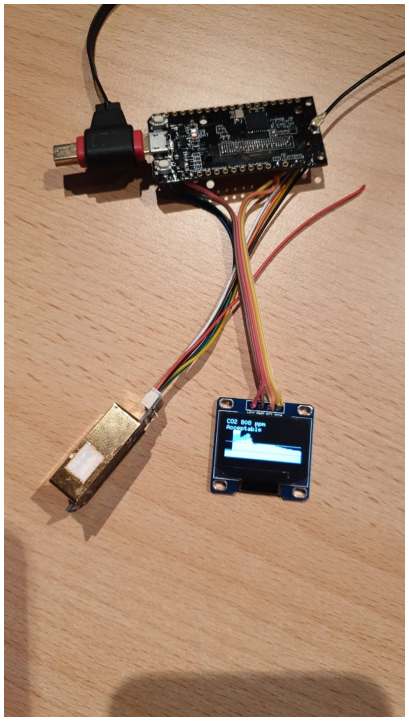
Na dann legen wir mal los …
Startpunkt dieses Projekts ist Sergey Smolnikov's esp32-with-co2-sensor-mh-z19b-and-lcd-display-nokia-5110
Link dazu: https://github.com/satr/esp32-with-co2-sensor-mh-z19b-and-lcd-display-nokia-5110
Zuerst wurde das Nokia Display durch das auf dem TTGO Lora Board verbaute SSD1306 ersetzt. Da das Display meines letzten vorhandenen TTGO ESP32 LORA allerdings defekt war, habe ich mir kurzerhand per ebay ein kompatibles beschafft und pinkompatibel angeschlossen. Dies bedingt natürlich eine Änderung der Displayansteuerung in Smolnikov's Code:
aus:

```
//pins description are above
U8G2_PCD8544_84X48_F_4W_SW_SPI display(U8G2_R0, /* clock=*/ 14, /* data=*/
13, /* cs=*/ 15, /* dc=*/ 27, /* reset=*/ 26);  // Nokia 5110 Display
//put another display from this file:
https://github.com/olikraus/u8g2/blob/master/tools/inoupdate/frame_buffer.in
o
const byte DISPLAY_WIDTH = 84;
const byte DISPLAY_HEIGHT = 48;
```

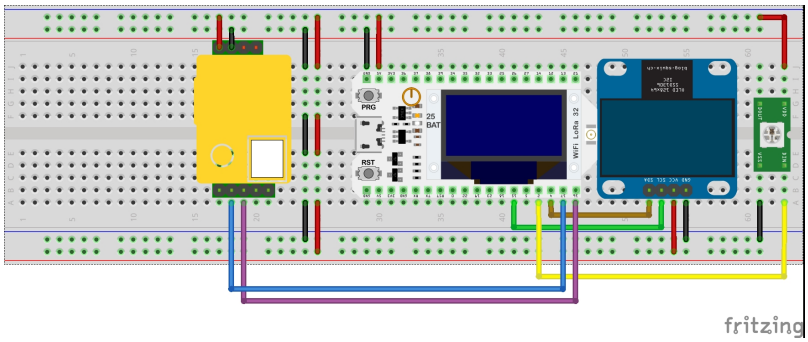
wird:

```
//pins description are above
U8G2_SSD1306_128X64_NONAME_F_HW_I2C display(U8G2_R0, /* reset=*/
U8X8_PIN_NONE, /* clock=*/ 15, /* data=*/ 4);   // ESP32 Thing, HW I2C with
pin remapping
//put another display from this file:
https://github.com/olikraus/u8g2/blob/master/tools/inoupdate/frame_buffer.in
o
const byte DISPLAY_WIDTH = 128;
const byte DISPLAY_HEIGHT = 64;
```

Den Rest seines Codes habe ich erst mal unverändert in meine „fliegend“ aufgebaute Hardware übernommen:



Um nun daraus eine Ampel zu kreieren muss das ganze Projekt natürlich noch eine „ampelmässige“ Anzeige in Form einer WS2812 LED erhalten. Diese wird am +/-5V und an Pin 2 des TTGO Boards (Datenleitung) angeschlossen. Damit ist erst mal die Hardware komplett. Das Foto dazu spare ich mir. Um das Ganze ein wenig verständlicher aufzubereiten, hab ich mal ein bisschen mit Fritzing „gezeichnet“:



in Textform sind die Verdrahtungsanweisungen dann auch noch im Sourcecode zu finden. Verwendet man das TTGO (Heltec) ESP32 LORA Board, oder Ähnliches, so ist das Display natürlich schon integriert, und man spart sich das separate Display (war bei mir nur separat, weil ich das integrierte gekillt hatte)

Die Routinen zur LED Ansteuerung habe ich aus https://esp8266-server.de/CO2Ampel.html entnommen und für unsere Zwecke angepasst.

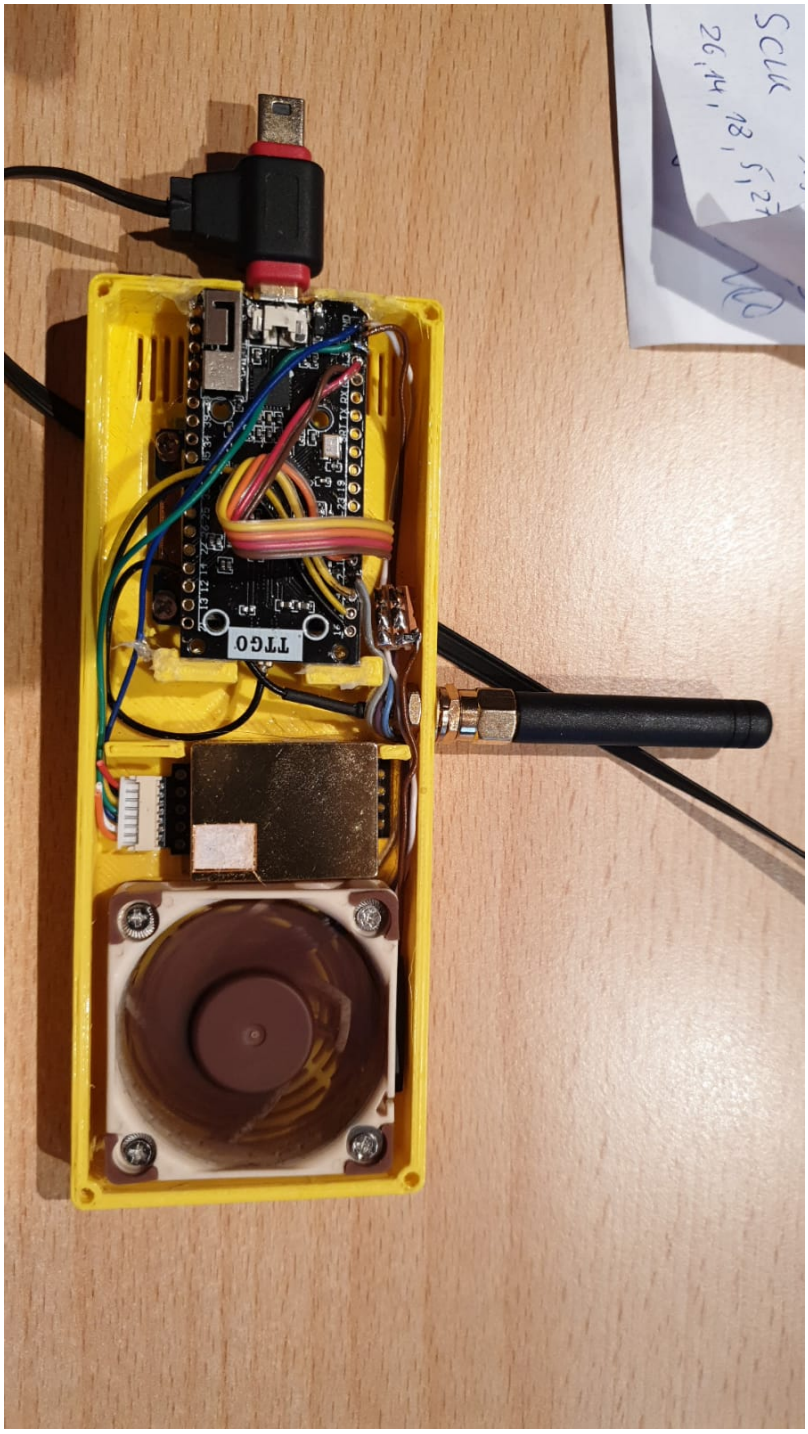
Die Einbindung ins TTN findet ihr unter
https://nathanmcminn.com/2018/09/12/tutorial-heltec-esp32-board-the-things-network/

Zuletzt sollte die Hardware dann noch in ein mehr oder weniger ansprechendes Gehäuse verpackt werden. Dazu hab ich mit SolidWorks einen ersten Gehäuseentwurf gezeichnet, nicht schön, aber zweckmässig … frei nach dem Motto „Form follows Function“. Hat trotzdem länger gedauert, als die Sotwarebausteine zusammen zu packen, da ich als Softwerker nun mal kein Konstrukteur bin, sondern im besten Fall Modellbauer. Hinter den runden Schlitzen verbirgt sich übrigens kein Lautsprecher, sondern ein 40mm Lüfter um ein gewisses Luftvolumen pro Minute durch den Sensor zu

saugen. Dieser ist direkt an die +5V Schiene angeschlossen. Der Lufteintritt ist die gerade Reihe an Schlitzen auf der rechten Seite, die Luft wird durch den Sensor gesaugt und vom Lüfter durch die runden Schlitze wieder ausgeblasen.

Und so sieht das Ergebnis aus:




TTGO



Hier gibts noch die Quelldateien dazu:
Software (Arduino Sketch):

bndlg_esp32_co2_ampel_lora_01.zip

Solidworks-Konstruktion:

gehaeuse-kpl-03.zip

STL-Dateien für 3D Druck:

gehaeuse-stls.zip

Und hier eine Einkaufsquelle (beispielsweise):
https://www.amazon.de/TOOGOO-Bluetooth-Entwicklungs-Anzeige-Antenne-Schwarz/dp/B07NKY9PNW

https://www.amazon.de/Dasorende-Infrarot-MH-Z19B-Monitor-0-5000Ppm/dp/B0838W5JVL

kurzes Python Script zum Auslesen der Ampel (access key bitte erfragen bei frefle@bndlg.de)

```
import requests
import sys
import pandas as pd
from datetime import datetime
import matplotlib.pyplot as plt

# Aufrufparameter
url =
"https://co2_corona_ampel.data.thethingsnetwork.org/api/v2/query/esp32_coron
a_ampel_bndlg"
args = '?last=7d'
access_key = 'ttn-account-.....'
headers = {'Accept': 'application/json', 'Authorization': 'key ' +
access_key}

# Abfrage der gespeicherten Daten per Swagger UI
try:
    response = requests.get(url + args, headers=headers)
except OSError as e:
    print("Error: {0}".format(e))
    sys.exit(0)
if response.status_code == 200:
    print("Status 200, OK")
    data = response.json()
else:
    print("JSON data request not successfull!")
    sys.exit(0)

# Darstellung
df = pd.DataFrame(data)
df['time'] = pd.to_datetime(df['time'])
ts = df.set_index('time')
print(ts)
ts.plot()
# plt.ylim(15,30)
plt.grid()
plt.show()
```

das sind Echtzeitdaten, ich hab die Ampel in verschiedenen Räumen aufgestellt. am 12.12. 17:30 z.B. 5 Personen im Esszimmer ...

auch MQTT geht vom TTN Server:

Ich habe mal quick 'n dirty MQTT in den Sketch eingebaut, jetzt spricht die Ampel MQTT mit dem Mosquitto Server auf meinem RasPi und LORA mit dem TTN (TheThingsNetwork). Über das Topic esp32/output läßt sich die blaue onboard LED des ESP32 Boards per on oder off Befehl schalten, das

geht per LORA und TTN auch, ist aber noch nicht implementiert. Allerdings bedarf WiFi und MQTT noch einiger Überarbeitung, es müsste noch einiges an Fehlerbehandlung eingebaut werden. Denn findet sich bspw. kein WLAN sollte die Ampel trotzdem noch LORA sprechen, momentan hängt sie in der Initialisierung fest, weil sie auf ein WLAN wartet. Ebenso muss für einen ESP32 ohne LORA Transceiver der LORA Code wieder entfernt oder per ifdef ausgeblendet werden. Es bleibt noch viel zu tun, packen wir's an.

Hier der neue Sketch:

bndlg_esp32_co2_ampel_lora_wifi_mqtt_01.zip

und so sieht's dann im MQTT Explorer aus:

Hier ein erster Versuch der Fehlerbehandlung: Diese Version wartet nicht ewig aufs WLAN, sondern macht nur 5 Versuche sich zu verbinden, dann gibt sie auf und bedient nur noch das TTN:

bndlg_esp32_co2_ampel_lora_wifi_mqtt_02.zip

Hier noch ein Fundstück aus dem Netz zum Thema Luftqualität, CO2 und IoT:
https://www.umwelt-campus.de/forschung/projekte/iot-werkstatt/ideen-zur-corona-krise

Michael Lotter hat folgende Lösung nachgebaut, welche gut funktioniert:
Hier zu der CO2-Ampel-Lösung von der HFT Stuttgart, welche den Sensor Sensirion SCD30 benützt. Überblick und Startpunkt für weitere Details:
https://www.hft-stuttgart.de/forschung/news/co2-ampel-lueften-gegen-covid-19

Weitere Dokumentationen und Firmware:
https://transfer.hft-stuttgart.de/gitlab/co2ampel

Einfacher Einstieg mit Sensirion:
https://developer.sensirion.com/tutorials/create-your-own-co2-monitor/

Bestellung (hohe Verfügbarkeit und blitzschnelle Lieferung):
https://www.mouser.de/new/sensirion/sensirion-scd30/?gclid=EAIaIQobChMI_uyin5ac7gIVitiyCh3uwA0ZEAAYAyAAEgK7JfD_BwE

From:
https://www.linuxag.bndlg.de/dokuwiki/ - **DokuWiki der Linux-AG im Bürgernetz Dillingen e.V.**

Permanent link:
https://www.linuxag.bndlg.de/dokuwiki/doku.php?id=co_-projekt_-_iot

Last update: **2021/01/15 21:47**