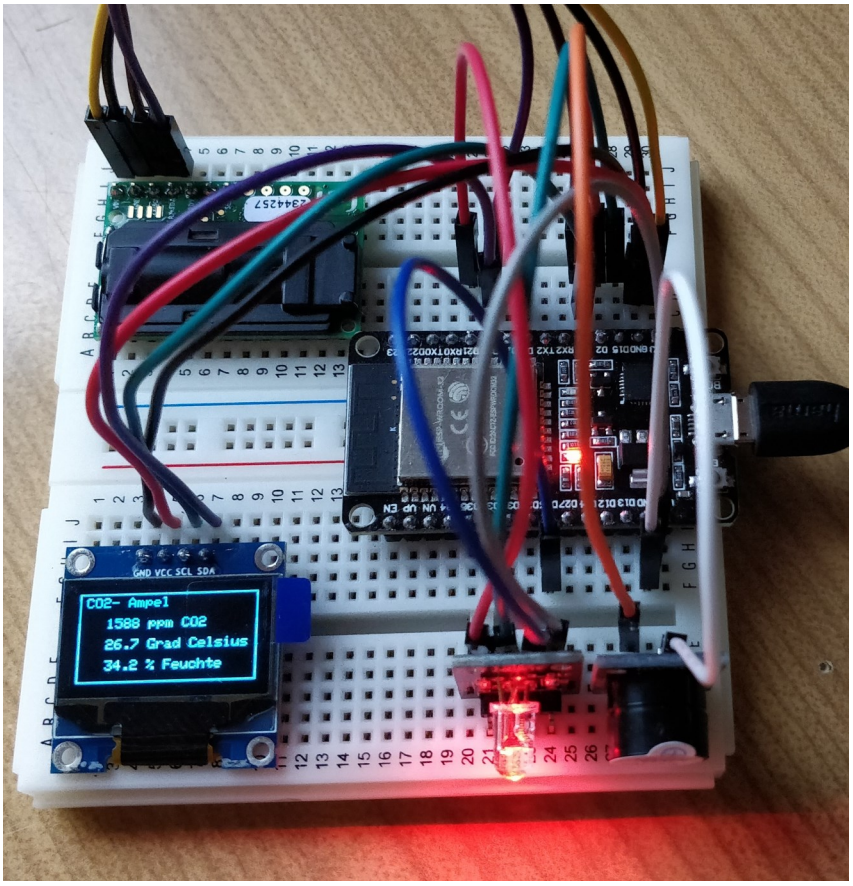
CO2 AMPEL

Kit für den Zusammenbau einer CO2 Ampel



1. ALLGEMEINE INFORMATIONEN

Sehr geehrter Kunde,
vielen Dank, dass Sie sich für unser Produkt entschieden haben. Im Folgenden zeigen wir Ihnen, was bei der Inbetriebnahme und der Verwendung zu beachten ist.

Sollten Sie während der Verwendung unerwartet auf Probleme stoßen, so können Sie uns selbstverständlich gerne kontaktieren.

2. EINFÜHRUNG

Aerosole können ein möglicher Übertragungsweg von Corona sein. Aerosole, die sich in geschlossenen oder schlecht belüfteten Innenräumen schnell verteilen, können die Ansteckung mit Corona erhöhen.

Das Risiko einer Infektion mit SARS-CoV-2 kann durch Lüften deutlich reduziert werden. Das ist das Ergebnis der Kommission Innenraumlufthygiene (IRK) am Umweltbundesamt.

Da die CO2- Konzentration ein guter Indikator für Aerosole ist, kann man über die CO2 Messung einen Rückschluss auf die Aerosole im Raum ziehen und erhält so einen Index, wann durch Lüften die Konzentration wieder gesenkt werden sollte.

So sind bei einer CO2 Konzentration von weniger als 1000 ppm keine weiteren Maßnahmen notwendig, aber bei einem Wert zwischen 1000 und 2000 ppm sollte gelüftet werden. Bei einer Konzentration von 2000 ppm und mehr sollte unbedingt verstärkt gelüftet werden, notfalls sogar die Anzahl der Personen im Raum reduziert werden.

Bei der Lüftung durch Öffnen der Fenster wäre eine Stoßlüftung mit Querlüftung optimal. Es wurde festgestellt, dass das Kippen der Fenster auch dauerhaft kaum wirksam ist! Somit könnte aber auch ein übermäßiges Lüften und Auskühlen der Klassenzimmer vermieden werden.

In Räumen bis 50 m^2 sollte ein Sensor angebracht werden, in größeren Räumen sollten entsprechend mehr Sensoren vorhanden sein.

Sensoren die den CO2- Äquivalenzwert messen (VOC - CO2 Äquivalent) sind auch ein guter Indikator für Aerosole, allerdings reagieren diese auch auf andere Quellen wie z.B. Lebensmittel. Für Schüler, die auch im Klassenzimmer essen, sind diese daher nur bedingt geeignet.

ZIEL

Das Ziel von diesem Bausatz ist der Bau und das Programmieren einer “CO2 Ampel” im Rahmen des Unterrichts.
Alle Tätigkeiten könnten in Gruppen ausgeführt werden (was momentan natürlich nicht möglich ist), aber natürlich auch mit entsprechendem Abstand voneinander oder allein. Sobald die Geräte fertig und funktionstüchtig sind (überprüft von der Lehrkraft) können diese für immer im Klassenzimmer verbleiben und dienen so dem Schutz der Anwesenden.
Alle “fertigen Geräte” können aber auch jederzeit wieder zerlegt werden und danach von anderen Schülern neu aufgebaut und neu programmiert werden (Nachhaltigkeit / Mehrfachnutzen).

- der Bausatz kann in der Schule im Unterricht zusammengebaut werden

- er (fördert und) erfordert handwerkliches Geschick

- das Programm für die NodeMCU ist enthalten, kann aber auch selbst programmiert werden (Informatik, MINT)

- wenn der Bausatz fertiggestellt wurde kann er für immer im Klassenzimmer verbleiben

- ab dann wird jeden Tag nur noch gelüftet, wenn es notwendig ist

- es stärkt die Gemeinschaft durch Teamwork

- der Bausatz ist nachhaltig: er kann jederzeit wieder zerlegt und mit anderen Schülern neu aufgebaut werden

AUFWAND / SCHWIERIGKEITSGRAD

Anbei unsere Empfehlung der Altersstufen für die einzelnen Tätigkeiten.

Komponenten auf einer Steckplatine positionieren

ab der 6. Klasse (handwerkliche Tätigkeit)

Komponenten verbinden mit Jumper- Brücken gemäß Anleitung

ab der 6. Klasse (handwerkliche Tätigkeit)

Programmierung der NodeMCU

ab der 10. Klasse aufwärts (Informatik)

Eine Musterlösung wird mitgeliefert.

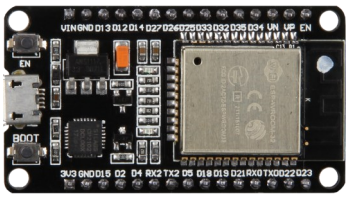
Nur einzelne Teile selbst programmieren lassen wie die RGB-LED oder den Buzzer.

Oder nur die Programmteile der Musterlösung erklären

3. PROGRAMMIERUNG

Der Bausatz besteht aus fünf Komponenten: dem Display, einer RGB-LED, einem Buzzer, einem CO2 Sensor und der NodeMCU mit einem ESP32.

CO2 Sensor



NodeMCU

RGB-LED

Buzzer

Display

SENSOR / DISPLAY
Die Messwerte vom CO2 Sensor sollen ausgelesen und am Display ausgeben werden.

RGB- LED
Die RGB- LED sollte über verschiedene Farben einen Index für die Konzentration anzeigen. Dafür würden die Farben rot, grün und blau zur Verfügung stehen. Mischfarben können aber auch ausprobiert werden. Eventuell könnte die LED auch z.B. ab einem Grenzwert von 2000 zu blinken beginnen.

BUZZER
Zusätzlich kann bei Überschreiten des Grenzwertes kurz der Buzzer aktiviert werden. Dieser sollte jedoch nicht zu lange aktiviert werden, da dies den Unterricht zu sehr stören würde. Der Zustand ist durch die LED jederzeit sichtbar. So könnte der Buzzer z.B. dreimal kurz piepsen, wenn der Grenzwert überschritten wurde und erst dann wieder einen Alarm auslösen, wenn der Grenzwert unterschritten und danach wieder erneut überschritten wurde.

ZUSÄTZLICHE FEATURES

SENSOR
Der Sensor braucht am Anfang ein wenig Zeit bis die Daten gültig sind. In dieser Wartezeit könnte man z.B. einen Countdown am Display ausgeben (in dieser Zeit sollte auch kein Alarm ausgegeben werden, selbst wenn der Wert zu hoch ist).

WEBSERVER
Einen Webserver erstellen, also einen Access Point “AP”: man verbindet das WLAN seines Smartphones oder vom Laptop mit diesem Netz und kann dann die Daten am Display ablesen. Man könnte z.B. auch den Alarm vom Buzzer ausschalten und wieder aktivieren über diese Weboberfläche.

INTERNET
Einbinden der Daten in die adafruit- Cloud: damit kann man die Daten im Internet einsehen und den Verlauf der Daten graphisch anzeigen.
Datum und Uhrzeit aus dem Internet holen und mit am Display ausgeben (um z.B. eine Pausenglocke mit einzubauen)

4. AUFBAU

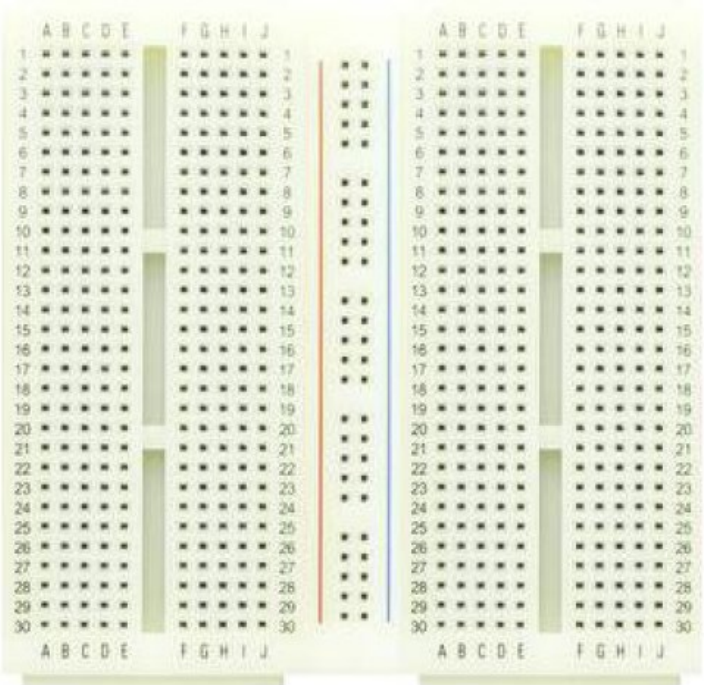
1. Steckplatinen zusammenbauen
2. Bauteile darauf platzieren
3. Steckbrücken einsetzen
4. NodeMCU programmieren
5. Gerät ins Gehäuse kleben

STECKPLATINE

Die Steckplatinen sind immer reihenweise durchkontaktiert: also ist in jeder Reihe 1-30 jeweils der Kontakt A bis E (also A,B,C,D und E) miteinander verbunden und die Kontakte F bis J sind verbunden (also F mit G, H, I und mit J).

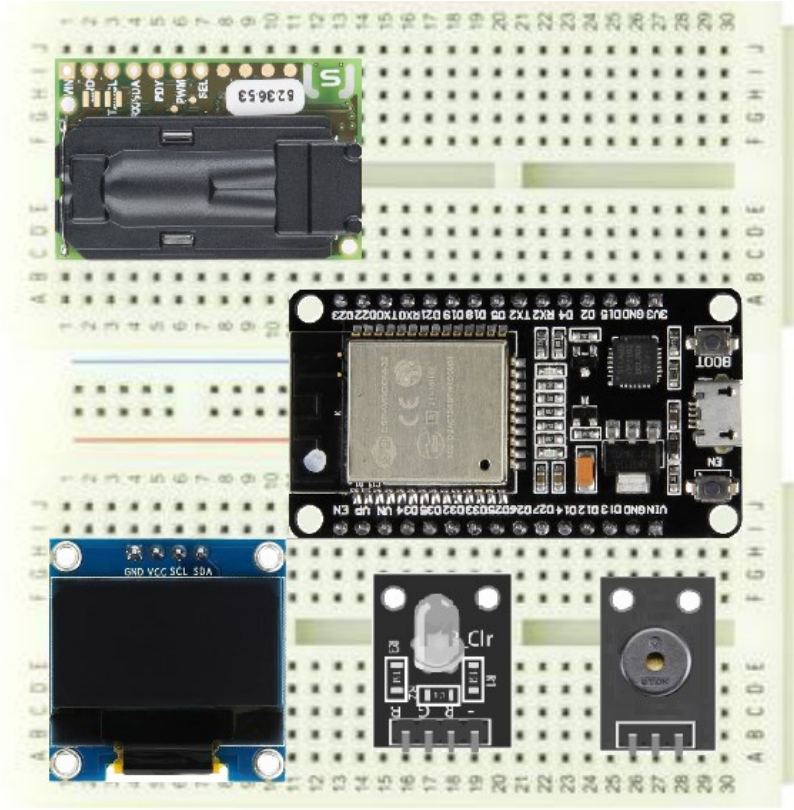
Bei den roten und blauen Streifen ist das anders: dort sind die Spalten untereinander verbunden. Die rote Spalte verbindet man normalerweise mit Plus und die blaue Spalte mit Minus: so kann man dann in jeder Zeile die Versorgungsspannung abgreifen wo man diese benötigt.

Das Problem ist die Breite der NodeMCU: wenn man den ESP auf eine normale Steckplatine aufbringt, dann ist auf einer Seite der NodeMCU keine Steckreihe mehr frei, also die NodeMCU reicht entweder von Spalte “A” bis “I” oder von Spalte “B” bis “J”). Daher scheidet eine einfache Steckplatine aus.
Am besten steckt man zwei Platinen zusammen und einen schmalen Streifen dazwischen (die mit den blau-roten Streifen). Wenn man dann die NodeMCU mittig aufsteckt, ist die Kontaktreihe fast in der Mitte (also entweder auf Spalte “A” und “I” oder auf Spalte “B” und “J”): somit hat man auf einer Seite drei und auf der anderen Seite vier Reihen frei für die Steckbrücken.



BAUTEILE PLATZIEREN

Positionieren Sie nun die Bauteile auf dem Steckbrett. Anbei unsere Empfehlung für die Anordnung. Natürlich können die fünf Bauteile anders angeordnet werden. Sie sollten nur beachten, dass Sie keinen Kurzschluss verursachen. Außerdem sollten am Display und der LED keine Kabel vorbeilaufen, damit man sie gut ablesen kann.



STECKBRÜCKEN EINSETZEN

Die 4 Komponenten müssen mit der NodeMCU verbunden werden:

- CO2 Sensor an den I^2C Bus anklemmen
- Display am I^2C Bus anklemmen
- LEDs an 3 Ports anschließen
- Buzzer an einen Port anschließen (keine PWM erzeugen, nur High für Ausgabe)

Es werden 14 Steckbrücken benötigt zur Verbindung der NodeMCU mit dem Sensor, dem Display, der RGB- LED und dem Buzzer.

Bauteil	Anschlüsse					Gesamtanzahl
CO2-Sensor	VCC	GND	SCL	SDA		4 Leitungen
Display	VCC	GND	SCL	SDA		4 Leitungen
LED		GND	R	G	B	4 Leitungen
Buzzer		GND		S		2 Leitungen

Nun werden die Steckbrücken nach Signalen eingesetzt. Zur Übersicht ist weiter unten das Pinout der NodeMCU dargestellt.

2 mal Plus
Das Display und der CO2-Sensor benötigen eine Versorgungsspannung Vcc von 3,3V (für Plus werden meist rote Kabel verwendet). Beim CO2 Sensor ist dies der Pin (VIN) ganz links, am Display ist es der zweite Pin von links (VCC). Diese beiden Pins werden mit dem Kontakt “3V3” der NodeMCU verbunden (in der Nähe vom BOOT Taster). Aber NICHT mit dem Vin der NodeMCU verbinden: das sind die 5V am Eingang vom USB Anschluss!

In unserem Beispiel ist der Pin “3V3” der NodeMCU auf dem Steckbrett am Kontakt “A27”. So könnten Sie die beiden Steckbrücken z.B. in “D27” und in “E27” stecken, aber natürlich auch in “B27” oder “C27”.

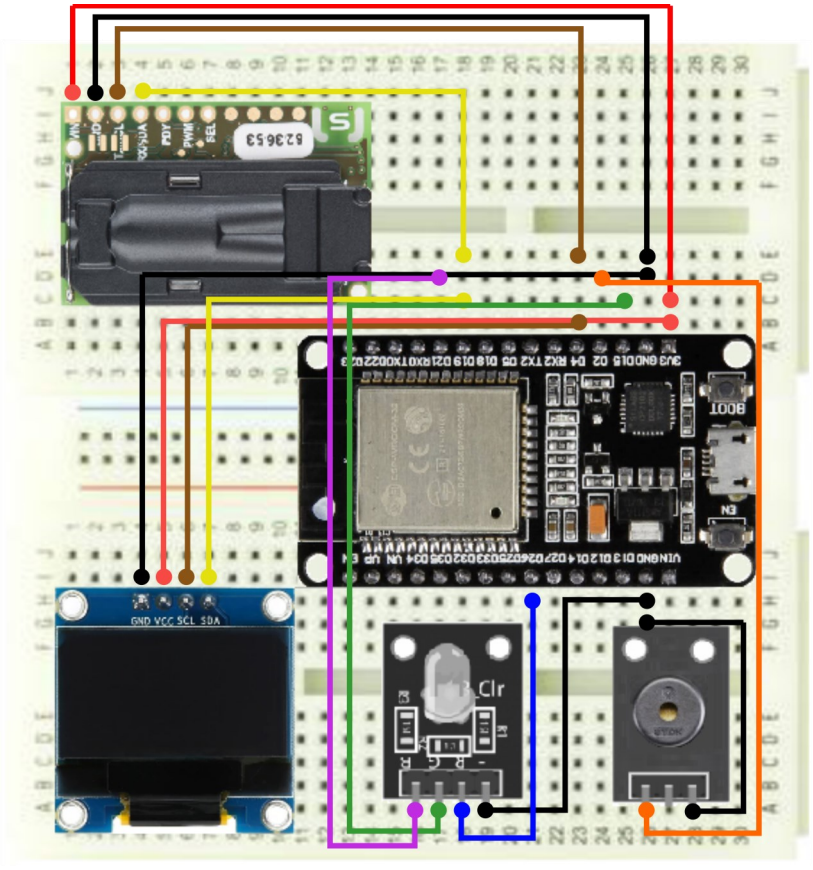
4 mal Minus
Alle 4 Bauteile benötigen einen Ground (für GND werden meist schwarze Kabel verwendet). Beim CO2 Sensor ist GND der 2. Pin von links, am Display ist es der ganz linke Pin, bei der RGB- LED und beim Buzzer ist es der Pin ganz rechts (-). Die NodeMCU hat 2 Anschlüsse für GND, und zwar jeweils am 2. Pin beim USB vorne (in unserem Beispiel ist der untere an “I26” und der obere an “A26”). Neben diesen beiden Pins kann man alle 4 GND anschließen.
Hinweis: die beiden GND sind intern verbunden und auch mit dem GND am USB Stecker verbunden. Daher muss man diese nicht extra verbinden.

2 SCL
Die beiden SCL Leitungen vom Display und dem Sensor (jeweils der 3. Pin von links) gehen an Port D4.

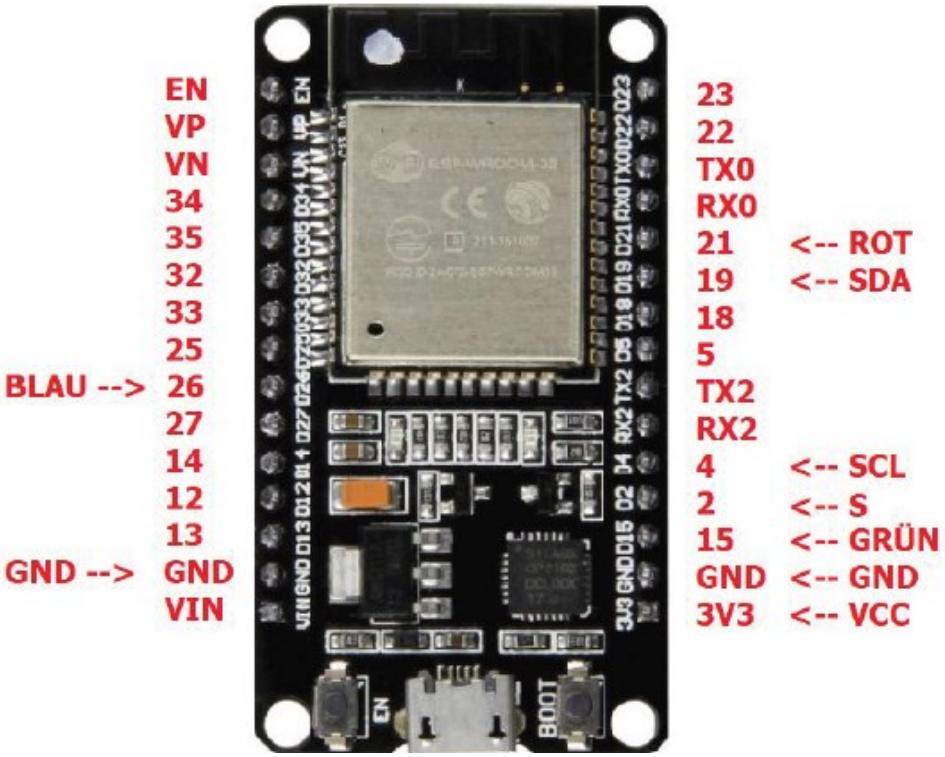
2 SDA
Die beiden SDA Leitungen vom Display und dem CO2- Sensor (jeweils der 4. Pin von links) werden mit D19 verbunden.

3 Farben RGB
Die 3 Farben der RGB LED gehen an drei verschiedene Pins: rot an D21, grün an D15 und blau an D26.

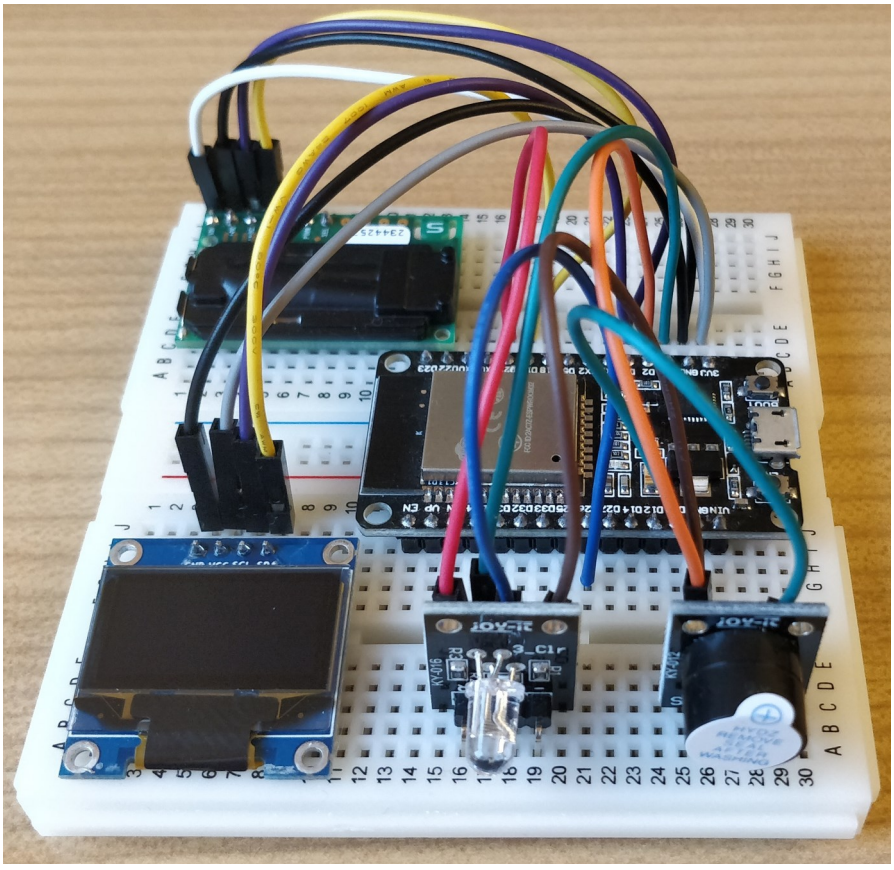
1 Sound
Vom Buzzer wird der Anschluss “S” ganz links mit D2 verbunden.



PINOUT NODEMCU



Am Ende könnte der Aufbau dann ungefähr so aussehen:



Danach wird von der Steckplatine unten die Schutzfolie abgezogen und die Platine auf den Boden im Gehäuse festgeklebt. Somit ist alles bereit für die Programmierung des ESP32. Im Gehäuse muss nur noch ein Loch an der Seite gebohrt werden, um das USB-Kabel durchzustecken.

5. PROGRAMMIERUNG

NODEMCU

Auf der NodeMCU befindet sich ein ESP32- Chip der Firma Espressif mit WLAN Funktionalität. Dieser Chip ist mit seinem bis zu 240 MHz- Takt sehr schnell.

Den ESP32- Chip der NodeMCU muss man programmieren: bei einer neuen NodeMCU ist nur der Bootloader auf dem Chip. Programmieren kann man die NodeMCU über die Arduino IDE Plattform. Aber das geht natürlich auch z.B. mit dem SDK von Espressif. Wir wollen hier die sehr beliebte Arduino IDE verwenden.

ARDUINO IDE INSTALLIEREN

Als erstes muss man die Arduino IDE herunterladen unter

https://www.arduino.cc/en/software

Dort wird rechts unter “Download Options” das Betriebsssystem ausgewählt.

ESP32 EINBINDEN

Nach der Installation wird die IDE gestartet.

Zuerst geht man unter “File” auf “Preferences” und gibt bei “Additional Boards Manager URLs:” die folgende Zeile ein:

https://dl.espressif.com/dl/package_esp32_index.json

Dann wird unter “Tools” / “Board” / “Boards Manager...” mit “esp32” die NodeMCU eingebunden. Dazu wird rechts von “Type” einfach esp32 eingegeben und wenn dieser angezeigt wird auf “install” gedrückt.

Nach dieser Installation kann man unter “Tools” / “Board” unter “ESP32 Arduino” das “ESP32 Dev Module” auswählen.

Jetzt kann die NodeMCU per USB mit dem Rechner verbunden werden. Unter “Tools” / “Port” wird der im Gerätemanager aufgeführte ComPort ausgewählt.

Hilfreich:

Unter “File / Preferences” kann man unter “Sketchbook location:” den Pfad eingeben, im dem man seine Dateien abspeichern will.

Und unter “File / Preferences” könnte man auch die “Editor language: ” auf Deutsch umstellen.

Dort müssen zusätzlich einige Einstellungen vorgenommen und Bibliotheken geladen werden. Eine Beschreibung (auf Englisch) findet man auch unter

https://docs.makerfactory.io/development-boards/esp32/install/

BIBLIOTHEKEN INSTALLIEREN

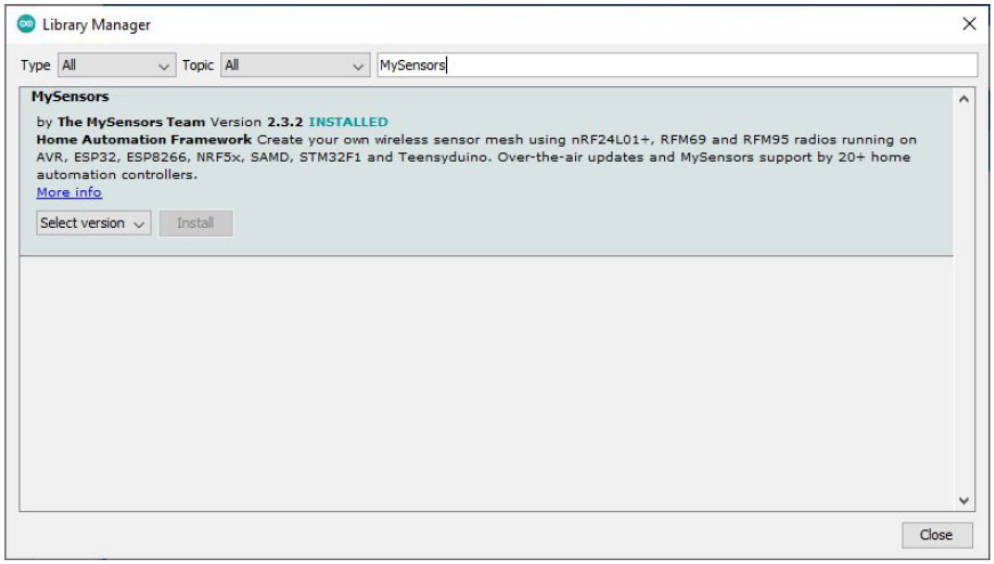
Bevor es losgeht müssen noch ein paar Bibliotheken installiert werden: Für das Display, für den CO2- Sensor, für die Schnittstelle und für die WiFi - Anbindung verwenden wir Bibliotheken.

Diese Bibliotheken sind der Riesenvorteil von Arduino, die die Arbeit erleichtern. Beim Display ist z.B. die "fillRect" Methode drin, um den Rahmen zu zeichnen und “display.setCursor” um den Cursor gezielt an eine Position zu bringen.

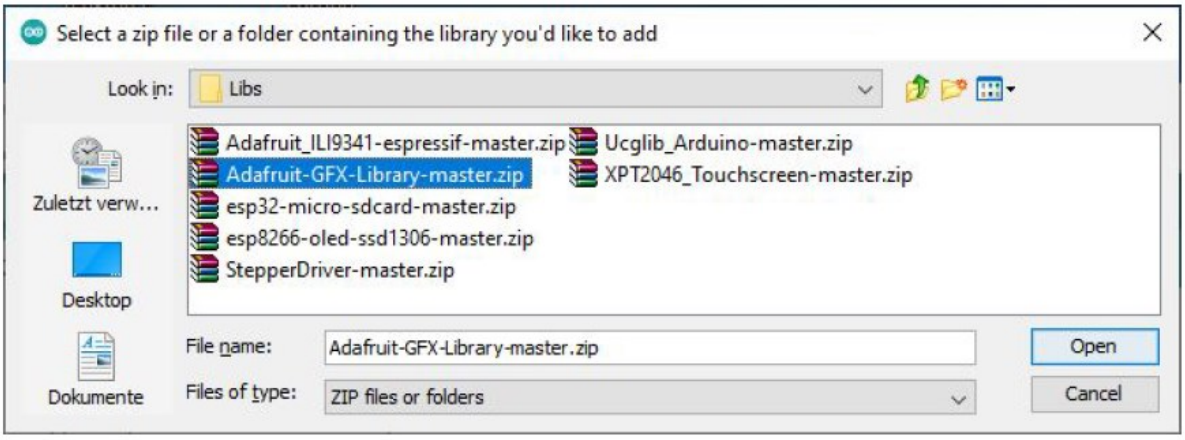
Es gibt verschiedene Arten, wie Bibliotheken eingebunden werden. Bei Problemen findet man hier eine Anleitung, wie Bibliotheken installiert werden:

https://www.arduino.cc/en/guide/libraries

Bibliotheken installiert man normalerweise mit dem “Library Manager”: den findet man unter “Sketch / Include Library / Manage Libraries…”. Dort wird die gewünschte Bibliothek gesucht oder direkt eingegeben und auf “install” gedrückt. Wenn alles geklappt hat sollte jeweils in der ersten Zeile der Beschreibung rechts “INSTALLED” stehen.



Bibliotheken aus ZIP- Dateien werden nicht entpackt sondern nur heruntergeladen: Diese ZIP- Dateien werden unter "Sketch - Include Library - Add .ZIP Library..." hinzugefügt und so mit eingebunden.



Bibliotheken, die in dem Ordner libraries sind, werden im Sketch in eckigen Klammern <> angegeben, Bibliotheken die sich im gleichen Ordner wie der Sketch befinden, werden in Anführungszeichen "" angegeben.

#include <LibraryFile.h> muss sich im libraries- Ordner befinden und

#include "LocalFile.h" muss sich lokal im gleichen Verzeichnis befinden

Und Bibliotheken werden immer ohne einen Strichpunkt am Ende angegeben.

Und wenn man eine spezielle Bibliothek verwendet, dann gibt man auch einen Link an, woher man die Bibliothek hat, damit das jeder ausprobieren kann.

Hier die Liste der verwendeten Bibliotheken:

```
#include <SPI.h> ist bei “MySensors” enthalten
#include <Wire.h> ist auch bei MySensors enthalten
#include <Adafruit_GFX.h>
// https://github.com/adafruit/Adafruit-GFX-Library
#include "Adafruit_SSD1306.h"
// https://github.com/adafruit/Adafruit_SSD1306
#include "SparkFun_SCD30_Arduino_Library.h"
// http://librarymanager/All#SparkFun_SCD30
#include <WiFi.h> Bibliothek WiFi
#include <WiFiClient.h>
#include <WiFiAP.h>
```

Wenn man sich nicht sicher ist ob das geklappt hat oder ob man eine Bibliothek vergessen hat, kann man den Sketch einfach kompilieren (das Häkchen in der Menüleiste anklicken): wenn eine Bibliothek nicht gefunden wurde, dann erhält man eine Fehlermeldung.

EINE ARDUINO DATEI ERSTELLEN

Alle Arduino- Programme werden “Sketch” genannt. Und die Arduino Dateien (*.ino) müssen sich IMMER in einem gleichnamigen Ordner befinden: der Sketch “test.ino” muss sich in einem Ordner “...\test” befinden. Wenn man eine neue Datei erstellt oder mit “Save As...” abspeichert, dann macht das die IDE automatisch.

Nun erstellen wir unter “File - New” einen neuen Sketch: der bekommt automatisch einen Namen und heißt z.B. “sketch_dec09a”. Unter ”File - Save as” kann man einen neuen Namen vergeben, der automatisch das Verzeichnis mit anlegt. Dieser Sketch besitzt nur zwei Teile: das setup und die loop:

```
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
    }
```

void setup() ist der Programmteil, der nur einmal beim Start ausgeführt wird

void loop() ist das Hauptprogramm: das läuft nach dem Setup in einer Endlos- Schleife

REIHENFOLGE BEIM PROGRAMMIEREN DER NODEMCU

Buzzer

kurz einschalten und wieder ausschalten lassen:
Beispiel aus File/Examples/Built-In Examples/01.Basics/Blink laden und anpassen

RGB- LED

Ebenso die drei Farben ein- und ausschalten (rot, blau, grün, und die Mischfarben testen)

Ausgabe

Texte seriell am PC ausgeben (z.B. “CO2- Ampel”)

Display

Texte am Display ausgeben (z.B. “CO2- Ampel”)

Sensor

CO2- Sensor auslesen
Werte vom Sensor seriell und am Display ausgeben

RGB-LED

Farbe der RGB- LED an den Messwert anpassen

Buzzer

Alarm- Ausgabe über den Messwert steuern

Arduino IDE

die Werte nicht nur am “Serial Monitor” ausgegeben lassen, sondern auch unter Tools am “Serial Plotter” grafisch anzeigen lassen.

Webserver

einen Webserver erstellen (Access Point AP)
Beispiel aus File/Examples/Examples for ESP32 Dev Module/WiFi/WiFiAccessPoint laden und abändern

Cloud

Einbinden der Daten in die adafruit- Cloud: damit kann man die Daten im Internet einsehen und den Verlauf der Daten graphisch anzeigen bzw. abspeichern (Trendanalyse)

BUZZER

Zuerst den Buzzer aktivieren - der Erfolg ist hörbar…

Dazu laden wir das Beispiel “Blink” aus File/Examples/Built-In Examples/01.Basics/Blink und ersetzen LED_BUILTIN durch die Zahl 2 für den Port D2, an dem der Buzzer angeschlossen ist und speichern das als Datei ”test2”. Die sieht dann so aus:

```
/*
Test2
Buzzer
*/
void setup() {
pinMode(2, OUTPUT);
}
void loop() {
digitalWrite(2, HIGH);
delay(1000); // wait for a second
digitalWrite(2, LOW);
delay(1000); // wait for a second
}
```

/* Kommentare werden mit einem Schrägstrich (Slash) und einem Sternchen (Asterisk) eingegrenzt */

// oder mit zwei Schrägstrichen (Slash): der gilt aber nur für den Rest dieser Zeile. Deshalb braucht man kein Ende- Zeichen

Der Aufruf delay(1000) bedeutet, dass das Programm 1000 Millisekunden pausieren soll.

UPLOAD AUF DIE NODEMCU

Nun wollen wir unser erstes Programm auf die NodeMCU laden.

Files werden mit “Upload” auf den ESP geladen (im Menü das zweite Symbol mit dem Pfeil nach rechts).

Zuvor sollte nochmals kontrolliert werden, ob unter “Tools” das richtige “Board” eingestellt wurde: Dort muss “ESP32 Dev Module” ausgewählt sein. Nach dem Anstecken muss dort auch der richtige “Port” angezeigt werden.

Beim Programmieren z.B. eines “Arduino Uno” wird über die DTR- Leitung am Atmel Chip ein Reset durchgeführt, der den Bootloader startet. So kann man den Chip ohne weiteres programmieren.

Das geht bei der NodeMCU leider nicht ganz so leicht. Hier braucht man zum Programmieren einen kleinen Trick: Sobald in der IDE unten das “Connecting….” erscheint, muss man auf die Taste “BOOT” drücken und diese halten, bis das flashen beginnt. Das ist alles.

Nun sollte man den Buzzer im Sekundentakt hören. Das Programm ist fest auf dem Chip, d.h. wenn man den USB abklemmt und wieder ansteckt, dann wird dieses Programm starten.

RGB- LED

Auf die gleiche Art wird die RGB- LED angesteuert. Die neue Datei soll als “test3” abgespeichert werden. Dort soll erst die rote, dann die blaue und zuletzt die grüne LED aufleuchten. Grün ist an D15, Blau an D26 und Rot ist an D21 angeschlossen. Diese wollen wir am Anfang definieren, z.B. die Grün als “RGB_GREEN”. Und Konstanten werden mit “const int” deklariert. Das Programm test3 könnte dann so aussehen:

```
/*
Test 3
RGB- LED
*/
const int RGB_GREEN = 15; // GREEN
const int RGB_BLUE = 26; // BLUE
const int RGB_RED = 21; // RED
void setup() {
pinMode(RGB_RED, OUTPUT);
pinMode(RGB_GREEN, OUTPUT);
pinMode(RGB_BLUE, OUTPUT);
}
void loop() {
digitalWrite(RGB_RED, HIGH);
delay(1000);
digitalWrite(RGB_RED, LOW);
delay(1000);
digitalWrite(RGB_BLUE, HIGH);
delay(1000);
digitalWrite(RGB_BLUE, LOW);
delay(1000);
digitalWrite(RGB_GREEN, HIGH);
delay(1000);
digitalWrite(RGB_GREEN, LOW);
delay(1000);
}
```

Jetzt sollte erst die rote, die blaue und zuletzt die grüne LED aufleuchten. Danach startet das Programm wieder von vorne (Endlosschleife).

SERIELLE AUSGABE

Das Programm “test4” soll einen Text seriell am PC ausgeben (z.B. “CO2-Alarm”). Dazu wird im setup() die serielle Schnittstelle initialisiert mit:

```
Serial.begin(115200);
```

In der Klammer wird die Geschwindigkeit der seriellen Schnittstelle festgelegt. Ein PC kennt normal diese Geschwindigkeiten: 1200, 2400, 4800, 9600, 19200, 38400, 57600 oder 115200 Bits pro Sekunde.

Der Aufruf "Serial.print" gibt nur den Text aus und "Serial.println" erzeugt zusätzlich einen Zeilenvorschub: so wird danach eine neue Zeile angefangen (das geht auch mit der Anweisung "\n").

```
Serial.println("CO2- Ampel");
Serial.println();
```

Dieser Teil wird in die void loop() eingetragen, damit der Text zum Test dauernd ausgegeben wird.

Um die Ausgabe zu sehen wird der “Serial Monitor” gestartet: der ist im Menü unter “Tools” oder man nimmt die Lupe oben rechts im Menü. Dann öffnet sich das Fenster mit der Ausgabe. Dort wird unten die Geschwindigkeit auf “115200 baud” eingestellt.

DISPLAY

Im Programm “test5” wollen wir einen Text am Display ausgeben (z.B. “CO2- Alarm”). Dazu wird die Bibliothek deklariert:

```
#include <Adafruit_GFX.h>
// https://github.com/adafruit/Adafruit-GFX-Library
#include "Adafruit_SSD1306.h"
// https://github.com/adafruit/Adafruit_SSD1306
#define OLED_RESET -1
    Adafruit_SSD1306 display(OLED_RESET);
```

Hier müssen sich die Dateien “Adafruit_SSD1306.h” und die “Adafruit_SSD1306.cpp” im gleichen Verzeichnis befinden.

Und im setup() wird das eingefügt:

```
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
display.setTextSize(1);
display.setTextColor(WHITE, BLACK);
display.setCursor(5, 3);
display.print("CO2- Ampel");
display.display();
```

Nun sieht man zum Text zusätzlich das Logo von adafruit. Das könnte mit display.clearDisplay(); gelöscht werden.

Sie können das Programm OLED_1 zum testen des Displays verwenden. Dieses wie alles weiteren Beispielprogramme kann man unter folgendem Link herunterladen:

https://joy-it.net/public/CO2Ampel.zip

Dieses Beispiel zeigt eine schöne Demo am Display (auch hier müssen sich die beiden Dateien im gleichen Verzeichnis befinden).

CO2 - SENSOR

Im Beispiel “test6” wird der CO2- Sensor ausgelesen.

Dazu tragen wir oben die Bibliothek ein und definieren den airSensor:

```
#include "SparkFun_SCD30_Arduino_Library.h"
SCD30 airSensor;
```

Im setup wird der Sensor gesucht:

```
Wire.begin();
if (airSensor.begin() == false) {
    Serial.println("CO2 Sensor wurde nicht gefunden!");
    while (1);
} else Serial.println("CO2 Sensor gefunden");
```

und in der loop werden die Daten seriell ausgegeben:

```
if (airSensor.dataAvailable()) {
    Serial.print("CO2 = ");
    Serial.print(airSensor.getCO2());
    Serial.println(" ppm ");
}
```

Am Display kann man den CO2 Wert z.B. so ausgeben:

```
display.setCursor(25, 25);
display.setTextSize(3);
display.print(airSensor.getCO2());
display.display();
```

Nun kann man die Grenzwerte für die Ampel und für den Alarm festlegen und die RGB- LED und den Buzzer entsprechend schalten.

Zum Test kann man den Sensor anhauchen: so wird getestet ob der Sensor funktioniert, und man kann zum Spaß testen, wer einen höheren Wert erzielt. So kann man auch prüfen, ob beim jeweiligen Grenzwert die LED-Anzeige stimmt und der Buzzer aktiviert wird.

SERIAL PLOTTER

Die Werte kann man nicht nur am “Serial Monitor” seriell ausgeben, sondern auch am “Serial Plotter” grafisch anzeigen. Diesen startet man einfach unter “Tools - Serial Plotter” - dazu muss aber der “Serial Monitor” geschlossen werden, da nur einer der beiden geöffnet sein kann. So erhält man einen Kurvenverlauf vom CO2- Wert.

WEBSERVER

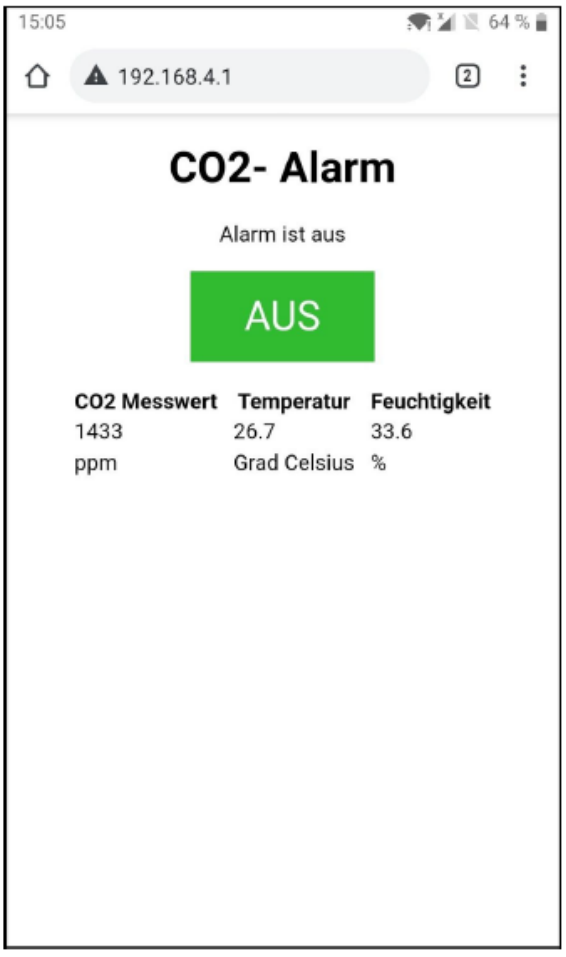
Zusätzlich könnte man einen Webserver erstellen (einen eigenen Access Point AP): Mit dem Smartphone loggt man sich dann in dieses WLAN-Netz ein und ruft mit dem Browser die angezeigte IP- Adresse auf. Auf dieser “Webseite” wird der aktuelle CO2- Wert dargestellt. Auf der Webseite kann man auch einen Button platzieren, mit dem man z.B. den Buzzer ein- und ausschalten kann.

Dazu lädt man am Besten das Beispiel im Ordner “File/Examples/ Examples for ESP32 Dev Module/WiFi/WiFiAccessPoint” und passt es an.

Das fertige Programm könnte so aussehen: “CO2_Ampel_AP” Wenn das Programm gestartet hat, wählt man mit dem Smartphone oder dem Laptop das Netzwerk “CO2-AP” aus und gibt als Passwort “password” ein (das Smartphone zeigt dann so etwas ähnliches an: “verbunden, kein Internet”). Wenn man dann mit dem Browser die angezeigte IP- Adresse (wahrscheinlich die 192.168.4.1) eingibt, sieht man die Oberfläche und kann mit dem Button den Buzzer ein- und ausschalten (bei Problemen evtl. die Mobilen Daten ausschalten, da das Smartphone sonst nur im Internet sucht).

Die Daten werden aber nicht von selbst aktualisiert.

Ansicht am Smartphone



CLOUD

Oder man verbindet die NodeMCU mit der adafruit cloud. Damit kann man die Daten von überall aus im Internet einsehen und den Verlauf der Daten graphisch anzeigen lassen.

Dazu muss man sich bei Adafruit (kostenlos) registrieren. Dann erhält man zu seinem Usernamen einen “key”. Außerdem benötigt man die Zugangsdaten zum WLAN.

Eine Erklärung findet man unter

https://learn.adafruit.com/adafruit-io-basics-dashboards

Bei github muss man die ZIP Datei “Adafruit_IO_Arduino-master.zip” herunterladen und installieren:

https://github.com/adafruit/Adafruit_IO_Arduino

Dann findet man Beispiele unter “File - Examples” im Ordner “Examples from custom Libraries” im Ordner “Adafruit IO Arduino” mehrere Beispiele, wie die “adafruitio_00_publish”. Diese speichert man unter einem anderen Namen (test8) und passt das File an.

Dort ist auch die Datei “config.h” enthalten, in der man seine Zugangsdaten für adafruit und für das WLAN eingibt. Der Vorteil von diesem Prinzip ist, dass man das Arduino- File (*.ino) jederzeit weitergeben oder auch veröffentlichen kann, ohne seine privaten Daten offenzulegen.

In der config.h müssen die vier “credentials” eingetragen werden:

```
#define IO_USERNAME "xxx"
#define IO_KEY "yyy"
#define WIFI_SSID "ssid"
#define WIFI_PASS "pass"
```

Im Hauptprogramm wird die “config.h” eingebunden und die “Feeds”:

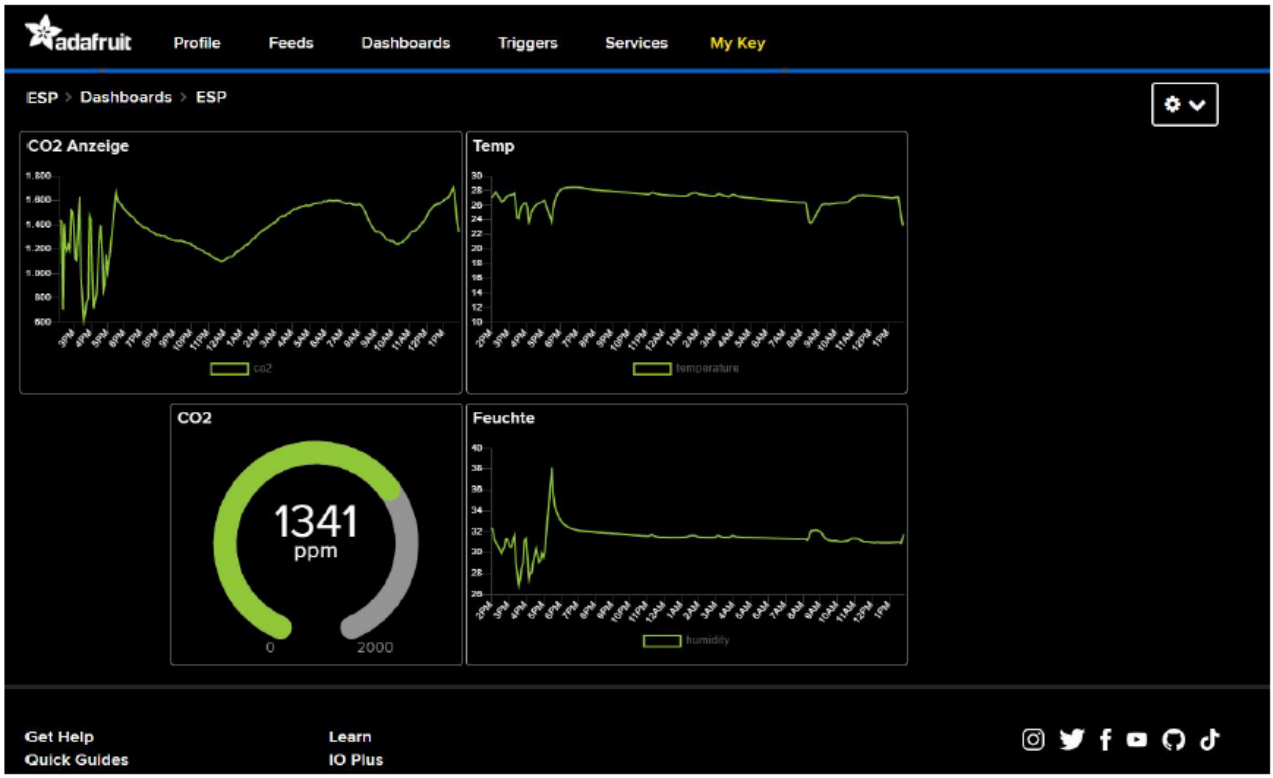
```
#include "config.h"
AdafruitIO_Feed *CO2 = io.feed("co2");
AdafruitIO_Feed *temperature = io.feed("temperature");
AdafruitIO_Feed *humidity = io.feed("humidity");
```

und mit “save” werden die Daten an die Cloud gesendet:

```
CO2->save(airSensor.getCO2());
```

In der freien Version von adafruit darf man nicht so viele Daten in die Cloud senden. Deshalb muss man nach jedem Senden eine Pause einbauen von 5 oder 10 Sekunden (und es genügt ja auch, wenn man nur alle 10 Sekunden einen neuen Wert einliest). Oder man “gruppiert” die Daten, wie es im Beispiel “adafruitio_11_group_pub” erklärt wird.

ANSICHT UNTER ADAFRUIT



KOMBINATION

Natürlich könnte man beide Programme für den Webserver und die Adafruit Cloud zusammenbauen, so dass der Webserver im WLAN- Netz erreichbar ist. Dann funktioniert das Programm aber nicht mehr ohne WLAN, z.B. wo man keinen WLAN Empfang hat oder keine Zugangsdaten - die Webserver- Variante funktioniert da aber, oder sogar mit einer Powerbank mitten im Wald.

Alle vorbereiteten Beispielprogramme können unter folgendem Link heruntergeladen werden:

https://joy-it.net/public/CO2Ampel.zip

6. SONSTIGE INFORMATIONEN

Unsere Informations- und Rücknahmepflichten nach dem Elektrogesetz (ElektroG)

Symbol auf Elektro- und Elektronikgeräten:

Diese durchgestrichene Mülltonne bedeutet, dass Elektro- und Elektronikgeräte **nicht** in den Hausmüll gehören. Sie müssen die Altgeräte an einer Erfassungsstelle abgeben. Vor der Abgabe haben Sie Altbatterien und Altakkumulatoren, die nicht vom Altgerät umschlossen sind, von diesem zu trennen.

Rückgabemöglichkeiten:
Als Endnutzer können Sie beim Kauf eines neuen Gerätes, Ihr Altgerät (das im Wesentlichen die gleiche Funktion wie das bei uns erworbene neue erfüllt) kostenlos zur Entsorgung abgeben. Kleingeräte bei denen keine äußere Abmessungen größer als 25 cm sind können unabhängig vom Kauf eines Neugerätes in haushaltsüblichen Mengen abgeben werden.

Möglichkeit Rückgabe an unserem Firmenstandort während der Öffnungszeiten:
SIMAC Electronics GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

Möglichkeit Rückgabe in Ihrer Nähe:
Wir senden Ihnen eine Paketmarke zu mit der Sie das Gerät kostenlos an uns zurücksenden können. Hierzu wenden Sie sich bitte per E-Mail an Service@joy-it.net oder per Telefon an uns.

Informationen zur Verpackung:
Verpacken Sie Ihr Altgerät bitte transportsicher, sollten Sie kein geeignetes Verpackungsmaterial haben oder kein eigenes nutzen möchten kontaktieren Sie uns, wir lassen Ihnen dann eine geeignete Verpackung zukommen.

7. SUPPORT

Wir sind auch nach dem Kauf für Sie da. Sollten noch Fragen offen bleiben oder Probleme auftauchen stehen wir Ihnen auch per E-Mail, Telefon und Ticket-Supportsystem zur Seite.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Telefon: +49 (0)2845 98469 – 66 (10 - 17 Uhr)

Für weitere Informationen besuchen Sie unsere Website:
www.joy-it.net

Veröffentlicht: 18.12.2020