CO2-Ampel

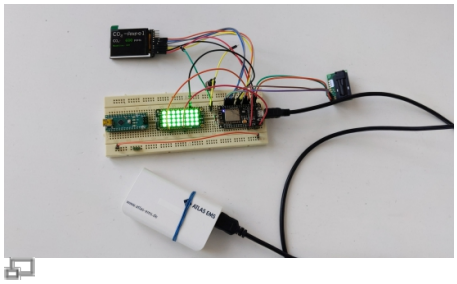
Quellen

Grundlagen und Projekthintergrund

- https://www.heise.de/select/make/2020/5/2022015381334973804 MAKE Artikel zum Thema CO2-Ampel und die Grundlage meines Bastelprojektes
- https://www.heise.de/select/make/2020/5/softlink/xyrg?wt_mc=pred.red.make.make052020.010.softlink.softlink Downloads und weitere Informationen zum MAKE Artikel
- https://www.umwelt-campus.de/forschung/projekte/iot-werkstatt Webseite der IoT-Werkstatt
- https://www.umwelt-campus.de/forschung/projekte/iot-werkstatt/ideen-zur-corona-krise-1 Hintergrundinformationen zum Projekt CO2-Ampel der IoT-Werkstatt

Informationen zur verwendeten Technik

- https://joy-it.net/de/products/SBC-NodeMCU Der verwendete Mikrocontroller
- https://joy-it.net/files/files/Produkte/SBC-NodeMCU/SBC-NodeMCU_pinout.png Pinbelegung des verwendeten Mikrocontrollers
- https://forbiddenbit.com/en/arduino-projects/esp8266/nodemcu-and-1-8-spi-st7735-display/ Verdrahtung und Ansteuerung meines Displays
- https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library-use Dokumentation zur Verwendung der Adafruit NeoPixel Bibliothek
- https://learn.adafruit.com/adafruit-gfx-graphics-library/coordinate-system-and-units Dokumentation zur Verwendung der Adafruit TFT Grafikdisplay Bibliothek
- http://javl.github.io/image2cpp/http://javl.github.io/image2cpp/ Code-Generator für Byte Grafiken (für die Smiley Bilder im Dirsplay)
- https://github.com/realholgi/Encoder Dokumentation zur verwendeten Bibliothek für den Dreh-Encoder (Einstellung der LED Helligkeit)
- https://www.pjrc.com/teensy/td_libs_Encoder.html Homepage des Entwicklers der Encoder Bibliothek
- https://www.instructables.com/Select-Color-Display-for-ESP32/ Übersicht verschiedener TFT-Displays
- https://circuits4you.com/2019/01/08/esp8266-turn-off-wifi-save-power/ Konfiguration desWIFI am ESP8266
- https://makesmart.net/esp8266-d1-mini-mqtt Grundlagen zur Einbindung von MQTT im ESP8266



Prototyp auf Steckbrett



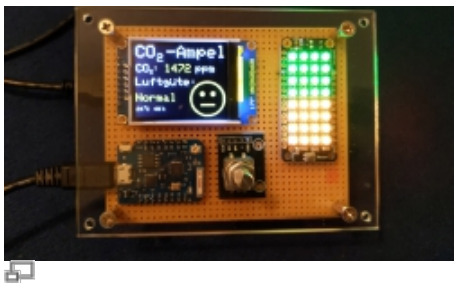
fertige Co2-Ampel an Powerbank



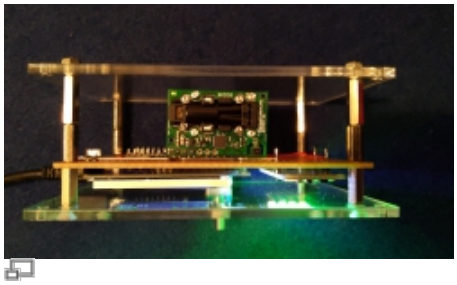
alles im grünen Bereich



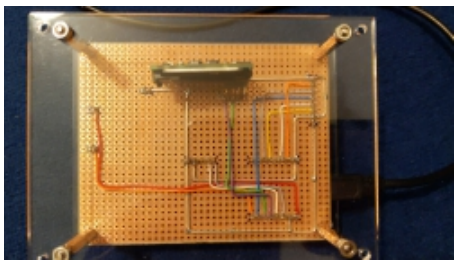
schlechte Luft



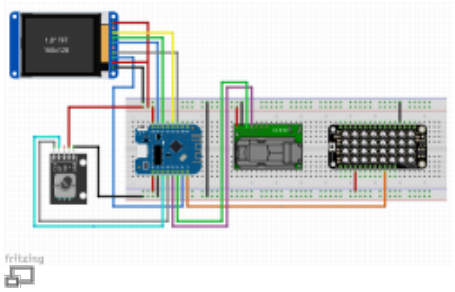
bals mal Lüften



Ansicht von oben



Verdrahtung von hinten



Verdrahtung via Fritzing

Arduino IDE

IDE herunterladen und installieren:

```
wget
https://downloads.arduino.cc/arduino-1.8.16-
linux64.tar.xz
tar xJz arduino-1.8.16-linux64.tar.xz
cd arduino-1.8.16
./install.sh
```

ESP8266 Unterstützung installieren:
`Arduino IDE –> Datei –> Voreinstellungen –> zusätzliche Boardverwalter URLs:`

```
https://arduino.esp8266.com/stable/package_es
p8266com_index.json
https://raw.githubusercontent.com/espressif/a
rduino-esp32/gh-
pages/package_esp32_dev_index.json
```

Einkaufsliste

Material:	Quelle:	ca. Preis	Link
D1 Mini Pro - ESP8266	Reichelt	7,30 €	https://www.reichelt.de/d1-mini-pro-esp8266-cp2104-d1-mini-pro-p266066.html
Sensirion SCD30	Mouser	43,46 €	https://www.mouser.de/ProductDetail/Sensirion/SCD30/?qs=rrS6PyfT74fdywu4FxpYjQ%3D%3D
NeoPixel FeatherWing - 4×8 RGB LED	Mouser	12,66 €	https://www.mouser.de/ProductDetail/Adafruit/2945/?qs=ivJcBTDythXnjfcjjhuHaQ%3D%3D
DEBO ROT SWITCH	Reichelt	2,65 €	https://www.reichelt.de/entwicklerboards-kodierter-drehschalter-debo-rot-switch-p239260.html
Display LED, 1,8„, 128×160, ST7735R	Reichelt	10,50 €	https://www.reichelt.de/entwicklerboards-display-led-1-8-128x160-st7735r-debo-tft-1-8-p192139.html
H25PR075 Lochrasterplatine, Hartpapier, 75x100mm	Reichelt	1,15 €	https://www.reichelt.de/lochrasterplatine-hartpapier-75x100mm-h25pr075-p8269.html

Quellcode

CO2-Ampel.ino

```
/*
Author:   Christoph von Thülen
Version:  1.0 10/23/2020

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.

Dieses Programm ist Freie Software: Sie können es unter den Bedingungen
der GNU General Public License, wie von der Free Software Foundation,
Version 3 der Lizenz oder (nach Ihrer Wahl) jeder neueren
veröffentlichten Version, weiter verteilen und/oder modifizieren.

Dieses Programm wird in der Hoffnung bereitgestellt, dass es nützlich
sein wird, jedoch
OHNE JEDE GEWÄHR,; sogar ohne die implizite
Gewähr der MARKTFÄHIGKEIT oder EIGNUNG FÜR EINEN BESTIMMTEN ZWECK.
Siehe die GNU General Public License für weitere Einzelheiten.

Sie sollten eine Kopie der GNU General Public License zusammen mit
diesem
Programm erhalten haben. Wenn nicht, siehe
<https://www.gnu.org/licenses/>.
-------------------------------------------------------------------------
-------------------------*/

// Libs to controll the 1.8" 120x160 px Display with ST7735 Controller:
#include <Adafruit_GFX.h>
#include <Adafruit_ST7735.h>
#include <SPI.h>

// Libs for the MCU itself
#include <ESP8266WiFi.h>  // Wifi part

// Lib for quadrature encoder like KY-040 or similar:
#include <Encoder.h>
#define ENCODER_OPTIMIZE_INTERRUPTS
```

```
// Libs to control the Sensirion CO2 sensor:
#include <SparkFun_SCD30_Arduino_Library.h>
#include <Wire.h>

// Adafruit NeoPixel FeatherWing
#include <Adafruit_NeoPixel.h>

// Include the Smiley grafics as byte array...
#include "graphics.c"

// Define some important things ...
// Used Pins to control the Display:
#define TFT_CS         15      // Use GPIO15 for Chip Select (CS)
#define TFT_RST         3      // Use GPIO0 for RESET
#define TFT_DC         16      // Use GPIO2 for DC

// Used pins to control the NeoPixel LEDs:
#define NEOPIXEL_DATA_PIN   1 // Use GPIO 1 for NeoPixel DIN (Data In)

// time intervall in seconds to read out the CO2 sensor
#define INTERVALL 5 // seconds
#define CO2_BAD_LOWER_LIMIT    2000
#define CO2_GOOD_UPPERLIMIT    1000

// Used rotary ecoder Pins:
#define ENC_DT  2        // use GPI/O 2 for DT
#define ENC_CLK 0        // use GPI/O 0 for CLK
#define ENC_BUTTON 12    // use GPI/O 12 for button click

// some color definitions:
#define BLACK    0x0000
#define BLUE     0x001F
#define RED      0xF800
#define GREEN    0x07E0
#define CYAN     0x07FF
#define MAGENTA  0xF81F
//#define ORANGE   0xFC20
#define ORANGE   0xFC00
#define YELLOW   0xFFE0
#define WHITE    0xFFFF

// some (font) size definitions
#define SMALL  1
#define MEDIUM 2
#define LARGE  3

// Internal state definitions
#define STATE_UNDEFINED 0
#define STATE_GOOD      1
#define STATE_NORMAL    2
```

```
#define STATE_BAD       3

// Lets call our 1.8" TFT Display (Controller Type: ST7735) simply:
"tft" ;-)
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST); //Those
things are for the display

// Objekt SCD30 environment sensor
SCD30 airSensorSCD30;

// Declare some usefull variables ...
// ... for the CO2 Sensor:
int CO2 = 0 ;
char CO2_BUFFER[5] = {};
int Temperature = 0;
int Humidity = 0;

// ... for the LEDs:
#define MAX_BRIGHTNESS 38      // set maximum brightness value
#define MAX_LED_ROWS    8
#define LEDS_PER_ROW    4
#define NEOPIXEL_NO_OF_LEDS (MAX_LED_ROWS * LEDS_PER_ROW)
//NEOPIXEL_NO_OF_LEDS 32   // NeoPixel PCB has n LEDs
#define LED_OFF 0              // set corresponding color brightness to
zreo --> LED is switched of
int Brightness = 16;           // set default brightness to 16
int count = 0;                 // set counter to 0
int Row = 0;

// ... for the display graphics:
int graphics_width = 64;
int graphics_height = 64;
int Title_x_pos = 2;
int Title_y_pos = 1;
int CO2_Value_x_pos = 2;
int CO2_Value_y_pos = 34;
int InfoText_x_pos = 1;
int InfoText_y_pos = 58;
int State_y_Offset = 28;
int gfx_x_pos = 96;
int gfx_y_pos = 64;
int Temp_Humid_x_pos = 2;
int Temp_Humid_y_pos = 110;

extern const unsigned char smiley_good [];
extern const unsigned char smiley_normal [];
extern const unsigned char smiley_bad [];

// internal states
int last_state = STATE_UNDEFINED;       // set startup state to
undefined
```

```
int new_state = STATE_UNDEFINED;        // set startup state to
undefined

// Init Adafruit NeoPixel Matrix:
Adafruit_NeoPixel neoWing = Adafruit_NeoPixel(NEOPIXEL_NO_OF_LEDS,
NEOPIXEL_DATA_PIN, NEO_GRB + NEO_KHZ800);

// Init encoder lib with uses interrupt pins connected to encoder
Encoder encoder(ENC_CLK, ENC_DT);

void FillLEDMatrix(int, int, int);

// Some things to set up before the main loop starts:
void setup() {

// Display Setup
  //tft.initR(INITR_GREENTAB);   // use for displays with green tab
aka. Joy-IT
  tft.initR(INITR_BLACKTAB);   //  use for displays with red or black
tab
  tft.setRotation(1);          // Displayinhalt um 90° (1) gedreht
(Landscape Mode); (2) = 180°, (3) = 270°

// Configure serial interface (UART) for debug output:
  Serial.begin(115200);        // Note: if GPI/O Pin 1 is used other
than UART TxD
                               // then debug output prints via
"serial.print" will not work correctly in loop()
  Serial.println("Co2 Monitor is starting ...");

 //Turn off WiFi
  WiFi.mode(WIFI_OFF);    //This also works

// Initialize the I2C-Bus:
  Wire.setClock(100000L);            // 100 kHz SCD30
  Wire.setClockStretchLimit(200000L);// CO2-SCD30
  Wire.begin();
  if (Wire.status() != I2C_OK) Serial.println("Something wrong with
I2C");

// Initialize CO2 Sensor:
  if (airSensorSCD30.begin() == false) {
    Serial.println("The SCD30 did not respond. Please check wiring.");
    while(1) {
      yield();
      delay(1);
    }
  }

  airSensorSCD30.setAutoSelfCalibration(false); // Sensirion no auto
calibration
```

```
  airSensorSCD30.setMeasurementInterval(INTERVALL);     // CO2
measurement every 5 seconds

// Initialize NeoPixel LEDs:
  neoWing.begin();
  delay(1);

// Print static elemets on display screen:
// Title:
  tft.fillScreen(ST77XX_BLACK);
  tft.setTextWrap(false);
  tft.setCursor(Title_x_pos, Title_y_pos);    //Horiz (ma. 160
px)/Vertical (max. 128 px)
  tft.setTextSize(LARGE);
  tft.setTextColor(ST77XX_WHITE,ST77XX_BLACK);
  tft.print("CO -Ampel");
  tft.setCursor(Title_x_pos+37, Title_y_pos+14);    //Horiz (ma. 160
px)/Vertical (max. 128 px)
  tft.setTextSize(MEDIUM);
  tft.setTextColor(ST77XX_WHITE,ST77XX_BLACK);
  tft.print("2");

// Text for CO2 Value:
  tft.setCursor(CO2_Value_x_pos, CO2_Value_y_pos);
  tft.setTextSize(MEDIUM);
  tft.setTextColor(ST77XX_WHITE,ST77XX_BLACK);
  tft.print("CO");
  tft.setCursor(CO2_Value_x_pos+24, CO2_Value_y_pos+10);
  tft.setTextSize(SMALL);
  tft.print("2");
  tft.setCursor(CO2_Value_x_pos+29, CO2_Value_y_pos);
  tft.setTextSize(MEDIUM);
  tft.print(":");
  tft.setCursor(CO2_Value_x_pos+105, CO2_Value_y_pos);
  tft.setTextSize(MEDIUM);
  tft.print("ppm");

// Info Text
  tft.setCursor(InfoText_x_pos, InfoText_y_pos);
  tft.setTextSize(MEDIUM);
  tft.setTextColor(ST77XX_WHITE,ST77XX_BLACK);
  tft.print("Luftg""\201""te:");
}

// Begin main loop:
void loop() {

// Read new CO2 measurement from sensor:
  CO2 = airSensorSCD30.getCO2();                  // get new CO2 value
from sensor
  Temperature = airSensorSCD30.getTemperature();  // get new
```

```
temperature value from sensor
  Humidity = airSensorSCD30.getHumidity();        // get new humidity
value from sensor

// Debug output via serial interface:
//  Serial.println("CO2: "+String(String(CO2)));
//  Serial.println();

// display temperatur & humidity
  tft.setCursor(Temp_Humid_x_pos, Temp_Humid_y_pos);
  tft.setTextSize(SMALL);
  tft.setTextColor(ST77XX_WHITE,ST77XX_BLACK);
  tft.print(Temperature);
  tft.setCursor(Temp_Humid_x_pos+12, Temp_Humid_y_pos-4);
  tft.print((char)9);
  tft.setCursor(Temp_Humid_x_pos+17, Temp_Humid_y_pos);
  tft.print("C");
  tft.setCursor(Temp_Humid_x_pos+30, Temp_Humid_y_pos);
  tft.print(Humidity);
  tft.setCursor(Temp_Humid_x_pos+44, Temp_Humid_y_pos);
  tft.print("%");

  tft.setCursor(InfoText_x_pos, InfoText_y_pos+State_y_Offset);
  tft.setTextSize(MEDIUM);

// Compare CO2 value to three ranges:
  if(CO2 < CO2_GOOD_UPPERLIMIT) {                          // we are in
state GOOD
    new_state = STATE_GOOD;
    if ( last_state != new_state) {        // when the state has
changed, draw new text and simley
      tft.setTextColor(ST77XX_GREEN,ST77XX_BLACK);
      tft.print("Gut     ");
      tft.drawBitmap(gfx_x_pos, gfx_y_pos, smiley_bad, graphics_width,
graphics_height, BLACK);
      tft.drawBitmap(gfx_x_pos, gfx_y_pos, smiley_normal,
graphics_width, graphics_height, BLACK);
      tft.drawBitmap(gfx_x_pos, gfx_y_pos, smiley_good, graphics_width,
graphics_height, GREEN);
    }
    FillLEDMatrix(1, MAX_LED_ROWS, LED_OFF, Brightness, LED_OFF);
  }
  else if (CO2 >= CO2_BAD_LOWER_LIMIT) {                 // we are in
state BAD!
    new_state = STATE_BAD;
    if ( last_state != new_state) {       // when the state has
changed, draw new text and simley
      tft.setTextColor(ST77XX_RED,ST77XX_BLACK);
      tft.print("Schlecht");
      tft.drawBitmap(gfx_x_pos, gfx_y_pos, smiley_good, graphics_width,
graphics_height, BLACK);
```

```
      tft.drawBitmap(gfx_x_pos, gfx_y_pos, smiley_normal,
graphics_width, graphics_height, BLACK);
      tft.drawBitmap(gfx_x_pos, gfx_y_pos, smiley_bad, graphics_width,
graphics_height, RED);

    }
    FillLEDMatrix(1, MAX_LED_ROWS, Brightness, LED_OFF, LED_OFF);
  }
  else {                                  // we are in state NORMAL
    new_state = STATE_NORMAL;
    if ( last_state != new_state) {     // when the state has changed,
draw new text and simley
      tft.setTextColor(ST77XX_YELLOW,ST77XX_BLACK);
      tft.print("Normal  ");
      tft.drawBitmap(gfx_x_pos, gfx_y_pos, smiley_bad, graphics_width,
graphics_height, BLACK);
      tft.drawBitmap(gfx_x_pos, gfx_y_pos, smiley_good, graphics_width,
graphics_height, BLACK);
      tft.drawBitmap(gfx_x_pos, gfx_y_pos, smiley_normal,
graphics_width, graphics_height, YELLOW);
    }
    Row = (((CO2-1000)/125)+1);                 // calculate how many
LED rows we have to fill
    FillLEDMatrix(1, Row, (Brightness*2), Brightness, LED_OFF);

    // fill the rest with green color to get a rolling traffic light
effect
    if (Row < MAX_LED_ROWS){
      FillLEDMatrix((Row+1), MAX_LED_ROWS, LED_OFF, Brightness,
LED_OFF);
    }
  }
  last_state = new_state;  // set new state

  // set color to update CO2 value on display
  switch(last_state) {
  case 1: tft.setTextColor(ST77XX_GREEN,ST77XX_BLACK); break;
  case 2: tft.setTextColor(ST77XX_YELLOW,ST77XX_BLACK); break;
  case 3: tft.setTextColor(ST77XX_RED,ST77XX_BLACK); break;
  }

  // refresh CO2 Value on display:
  tft.setCursor(CO2_Value_x_pos+50, CO2_Value_y_pos);
  tft.setTextSize(MEDIUM);
  sprintf(CO2_BUFFER, "%4d", CO2);
  tft.print(String(CO2_BUFFER));

// read encoder:
  count = encoder.read();

// Debug output via serial terminal
```

```
  //Serial.print("Count: ");
  //Serial.print(count);
  //Serial.print(", Brightness: ");
  //Serial.print(Brightness);

// if encoder was turned left or right, set new brightness values:
// don't go higher than MAX_BRIGHTNESS linit
  if ((Brightness + count) > MAX_BRIGHTNESS ) {
    Brightness = MAX_BRIGHTNESS;
  }
// don't go negative!
  if ((Brightness + count) < 0 ) {
    Brightness = 0;
  }else {
    // increase or decrease brightness value according to positiv or
negative count value
    Brightness += count;
  }
  Serial.println(Brightness);

// Debug output via serial terminal
//    Serial.print(", Red: ");
//    Serial.print(Led_red);
//    Serial.print(", Green: ");
//    Serial.print(Led_green);
//    Serial.print(", Blue: ");
//    Serial.print(Led_blue);
//    Serial.println();

// reset encoder value for next read cycle:
    encoder.write(0);

 // delay(1000);  // for debug and test purpose only
}

// Function to fill LED matrix row by row
void FillLEDMatrix(int row_start, int row_end, int LED_red, int
LED_green, int LED_blue){

// use for filling the LED matrix from left to right
//  for (uint16_t j=row_start-1; j < row_end; j++) {
//    for(uint16_t i=0+j; i < NEOPIXEL_NO_OF_LEDS  ; i+= MAX_LED_ROWS)
{
//      neoWing.setPixelColor(i,LED_red,LED_green,LED_blue);
//    }
//  }

// use for filling the LED matrix from right to left
  for (uint16_t j=row_start-1; j < row_end; j++) {
    for(uint16_t i=0+j; i < NEOPIXEL_NO_OF_LEDS  ; i+= MAX_LED_ROWS) {
      neoWing.setPixelColor((NEOPIXEL_NO_OF_LEDS-1-
```

```
i),LED_red,LED_green,LED_blue);
    }
  }
  neoWing.show();
}
```

graphics.c

```
#include <avr/pgmspace.h>

// 'Simley_good_bw_64x64px', 64x64px
const unsigned char smiley_good [] PROGMEM = {
  0x00, 0x00, 0x00, 0x1f, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
0xff, 0xff, 0x80, 0x00, 0x00,
  0x00, 0x00, 0x0f, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x3f,
0xff, 0xff, 0xfe, 0x00, 0x00,
  0x00, 0x00, 0xff, 0xc0, 0x03, 0xff, 0x00, 0x00, 0x00, 0x03, 0xfe,
0x00, 0x00, 0x3f, 0xc0, 0x00,
  0x00, 0x07, 0xf0, 0x00, 0x00, 0x0f, 0xe0, 0x00, 0x00, 0x0f, 0xc0,
0x00, 0x00, 0x03, 0xf8, 0x00,
  0x00, 0x1f, 0x00, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x00, 0x7e, 0x00,
0x00, 0x00, 0x00, 0x7e, 0x00,
  0x00, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x00, 0x00, 0xf8, 0x00,
0x00, 0x00, 0x00, 0x1f, 0x80,
  0x01, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x07, 0x80, 0x03, 0xe0, 0x00,
0x00, 0x00, 0x00, 0x07, 0xc0,
  0x07, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x03, 0xe0, 0x07, 0x80, 0x00,
0x00, 0x00, 0x00, 0x01, 0xf0,
  0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x0f, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x78,
  0x1e, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x78, 0x1e, 0x00, 0x0f,
0x80, 0x01, 0xf8, 0x00, 0x3c,
  0x3c, 0x00, 0x1f, 0xc0, 0x03, 0xfc, 0x00, 0x3c, 0x3c, 0x00, 0x3f,
0xe0, 0x03, 0xfc, 0x00, 0x3c,
  0x3c, 0x00, 0x3f, 0xe0, 0x07, 0xfe, 0x00, 0x1e, 0x78, 0x00, 0x3f,
0xe0, 0x07, 0xfe, 0x00, 0x1e,
  0x78, 0x00, 0x3f, 0xe0, 0x07, 0xfc, 0x00, 0x1e, 0x78, 0x00, 0x1f,
0xc0, 0x03, 0xfc, 0x00, 0x0f,
  0x70, 0x00, 0x1f, 0xc0, 0x01, 0xf8, 0x00, 0x0f, 0xf0, 0x00, 0x07,
0x00, 0x00, 0xe0, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x07,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x07,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x70, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
```

```
  0x78, 0x00, 0x60, 0x00, 0x00, 0x07, 0x00, 0x0f, 0x78, 0x00, 0xf0,
0x00, 0x00, 0x0f, 0x00, 0x1e,
  0x78, 0x00, 0xf8, 0x00, 0x00, 0x1f, 0x00, 0x1e, 0x38, 0x00, 0x7c,
0x00, 0x00, 0x3f, 0x00, 0x1e,
  0x3c, 0x00, 0x3f, 0x00, 0x00, 0xfe, 0x00, 0x3c, 0x3c, 0x00, 0x1f,
0xc0, 0x03, 0xfc, 0x00, 0x3c,
  0x1e, 0x00, 0x0f, 0xfc, 0x3f, 0xf0, 0x00, 0x3c, 0x1e, 0x00, 0x03,
0xff, 0xff, 0xe0, 0x00, 0x78,
  0x1f, 0x00, 0x01, 0xff, 0xff, 0x80, 0x00, 0x78, 0x0f, 0x00, 0x00,
0x3f, 0xfc, 0x00, 0x00, 0xf0,
  0x07, 0x80, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf0, 0x07, 0xc0, 0x00,
0x00, 0x00, 0x00, 0x03, 0xe0,
  0x03, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x03, 0xc0, 0x01, 0xf0, 0x00,
0x00, 0x00, 0x00, 0x07, 0xc0,
  0x01, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x80, 0x00, 0xfc, 0x00,
0x00, 0x00, 0x00, 0x1f, 0x00,
  0x00, 0x7e, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x00, 0x00, 0x3f, 0x00,
0x00, 0x00, 0x00, 0xfc, 0x00,
  0x00, 0x1f, 0xc0, 0x00, 0x00, 0x01, 0xf8, 0x00, 0x00, 0x07, 0xf0,
0x00, 0x00, 0x07, 0xf0, 0x00,
  0x00, 0x03, 0xfc, 0x00, 0x00, 0x3f, 0xc0, 0x00, 0x00, 0x00, 0xff,
0x80, 0x01, 0xff, 0x80, 0x00,
  0x00, 0x00, 0x7f, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x1f,
0xff, 0xff, 0xf8, 0x00, 0x00,
  0x00, 0x00, 0x03, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00,
0x3f, 0xfe, 0x00, 0x00, 0x00
};

// 'Simley_normal_bw_64x64px', 64x64px
const unsigned char smiley_normal [] PROGMEM = {
    0x00, 0x00, 0x00, 0x1f, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
0xff, 0xff, 0x80, 0x00, 0x00,
  0x00, 0x00, 0x0f, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x3f,
0xff, 0xff, 0xfe, 0x00, 0x00,
  0x00, 0x00, 0xff, 0xc0, 0x03, 0xff, 0x00, 0x00, 0x00, 0x03, 0xfe,
0x00, 0x00, 0x3f, 0xc0, 0x00,
  0x00, 0x07, 0xf0, 0x00, 0x00, 0x0f, 0xe0, 0x00, 0x00, 0x0f, 0xc0,
0x00, 0x00, 0x03, 0xf8, 0x00,
  0x00, 0x3f, 0x00, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x00, 0x7e, 0x00,
0x00, 0x00, 0x00, 0x7e, 0x00,
  0x00, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x00, 0x00, 0xf8, 0x00,
0x00, 0x00, 0x00, 0x1f, 0x80,
  0x01, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x80, 0x03, 0xe0, 0x00,
0x00, 0x00, 0x00, 0x07, 0xc0,
  0x07, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x03, 0xe0, 0x07, 0x80, 0x00,
0x00, 0x00, 0x00, 0x01, 0xe0,
  0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x1f, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0xf8,
  0x1e, 0x00, 0x02, 0x00, 0x00, 0x40, 0x00, 0x78, 0x1e, 0x00, 0x0f,
0x80, 0x01, 0xf8, 0x00, 0x7c,
  0x3c, 0x00, 0x1f, 0xc0, 0x03, 0xfc, 0x00, 0x3c, 0x3c, 0x00, 0x3f,
```

```
0xe0, 0x03, 0xfc, 0x00, 0x3c,
  0x3c, 0x00, 0x3f, 0xe0, 0x07, 0xfc, 0x00, 0x1e, 0x78, 0x00, 0x3f,
0xe0, 0x07, 0xfc, 0x00, 0x1e,
  0x78, 0x00, 0x3f, 0xe0, 0x07, 0xfc, 0x00, 0x1e, 0x78, 0x00, 0x1f,
0xc0, 0x03, 0xfc, 0x00, 0x0e,
  0x70, 0x00, 0x1f, 0xc0, 0x01, 0xf8, 0x00, 0x0f, 0xf0, 0x00, 0x07,
0x00, 0x00, 0xe0, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x70, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
  0x70, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x78, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x1e,
  0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1e, 0x38, 0x00, 0x0f,
0xff, 0xff, 0xf0, 0x00, 0x1e,
  0x3c, 0x00, 0x1f, 0xff, 0xff, 0xf8, 0x00, 0x3c, 0x3c, 0x00, 0x1f,
0xff, 0xff, 0xfc, 0x00, 0x3c,
  0x3e, 0x00, 0x1f, 0xff, 0xff, 0xf8, 0x00, 0x3c, 0x1e, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x78,
  0x1f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x0f, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0xf0,
  0x07, 0x80, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf0, 0x07, 0xc0, 0x00,
0x00, 0x00, 0x00, 0x03, 0xe0,
  0x03, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x03, 0xc0, 0x01, 0xf0, 0x00,
0x00, 0x00, 0x00, 0x07, 0xc0,
  0x01, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x80, 0x00, 0xfc, 0x00,
0x00, 0x00, 0x00, 0x1f, 0x00,
  0x00, 0x7e, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x00, 0x00, 0x3f, 0x00,
0x00, 0x00, 0x00, 0xfc, 0x00,
  0x00, 0x1f, 0xc0, 0x00, 0x00, 0x01, 0xf8, 0x00, 0x00, 0x07, 0xf0,
0x00, 0x00, 0x07, 0xf0, 0x00,
  0x00, 0x03, 0xfc, 0x00, 0x00, 0x3f, 0xc0, 0x00, 0x00, 0x00, 0xff,
0x80, 0x01, 0xff, 0x80, 0x00,
  0x00, 0x00, 0x7f, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x1f,
0xff, 0xff, 0xf8, 0x00, 0x00,
  0x00, 0x00, 0x03, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00,
0x7f, 0xfe, 0x00, 0x00, 0x00
};

// 'Simley_bad_bw_64x64px', 64x64px
const unsigned char smiley_bad [] PROGMEM = {
  0x00, 0x00, 0x00, 0x1f, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
0xff, 0xff, 0x80, 0x00, 0x00,
  0x00, 0x00, 0x0f, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x7f,
0xff, 0xff, 0xfe, 0x00, 0x00,
```

```
  0x00, 0x00, 0xff, 0xc0, 0x03, 0xff, 0x00, 0x00, 0x00, 0x03, 0xfe,
0x00, 0x00, 0x3f, 0xc0, 0x00,
  0x00, 0x07, 0xf0, 0x00, 0x00, 0x0f, 0xe0, 0x00, 0x00, 0x0f, 0xc0,
0x00, 0x00, 0x03, 0xf8, 0x00,
  0x00, 0x3f, 0x00, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x00, 0x7e, 0x00,
0x00, 0x00, 0x00, 0x7e, 0x00,
  0x00, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x00, 0x00, 0xf8, 0x00,
0x00, 0x00, 0x00, 0x1f, 0x80,
  0x01, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x80, 0x03, 0xe0, 0x00,
0x00, 0x00, 0x00, 0x07, 0xc0,
  0x07, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x03, 0xe0, 0x07, 0x80, 0x00,
0x00, 0x00, 0x00, 0x01, 0xe0,
  0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x1f, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0xf8,
  0x1e, 0x00, 0x02, 0x00, 0x00, 0x40, 0x00, 0x78, 0x3e, 0x00, 0x1f,
0x80, 0x01, 0xf8, 0x00, 0x7c,
  0x3c, 0x00, 0x3f, 0xc0, 0x03, 0xf8, 0x00, 0x3c, 0x3c, 0x00, 0x3f,
0xe0, 0x07, 0xfc, 0x00, 0x3c,
  0x38, 0x00, 0x3f, 0xe0, 0x07, 0xfc, 0x00, 0x1e, 0x78, 0x00, 0x3f,
0xe0, 0x07, 0xfc, 0x00, 0x1e,
  0x78, 0x00, 0x3f, 0xe0, 0x07, 0xfc, 0x00, 0x1e, 0x70, 0x00, 0x3f,
0xc0, 0x03, 0xfc, 0x00, 0x0e,
  0xf0, 0x00, 0x1f, 0xc0, 0x01, 0xf8, 0x00, 0x0f, 0xf0, 0x00, 0x07,
0x00, 0x00, 0xe0, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
  0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f,
  0x70, 0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00, 0x0e, 0x78, 0x00, 0x00,
0xff, 0xff, 0x00, 0x00, 0x1e,
  0x78, 0x00, 0x03, 0xff, 0xff, 0xc0, 0x00, 0x1e, 0x78, 0x00, 0x07,
0xff, 0xff, 0xf0, 0x00, 0x1e,
  0x3c, 0x00, 0x1f, 0xe0, 0x07, 0xf8, 0x00, 0x3c, 0x3c, 0x00, 0x3f,
0x80, 0x00, 0xfc, 0x00, 0x3c,
  0x1e, 0x00, 0x7e, 0x00, 0x00, 0x7e, 0x00, 0x7c, 0x1e, 0x00, 0xf8,
0x00, 0x00, 0x1f, 0x00, 0x78,
  0x1f, 0x00, 0xf0, 0x00, 0x00, 0x0f, 0x00, 0x78, 0x0f, 0x00, 0xe0,
0x00, 0x00, 0x07, 0x00, 0xf0,
  0x07, 0x80, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf0, 0x07, 0xc0, 0x00,
0x00, 0x00, 0x00, 0x03, 0xe0,
  0x03, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x03, 0xc0, 0x01, 0xe0, 0x00,
0x00, 0x00, 0x00, 0x07, 0xc0,
  0x01, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x80, 0x00, 0xfc, 0x00,
0x00, 0x00, 0x00, 0x1f, 0x00,
  0x00, 0x7e, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x00, 0x00, 0x3f, 0x00,
```

```
0x00, 0x00, 0x00, 0xfc, 0x00,
  0x00, 0x1f, 0x80, 0x00, 0x00, 0x01, 0xf8, 0x00, 0x00, 0x07, 0xf0,
0x00, 0x00, 0x07, 0xf0, 0x00,
  0x00, 0x03, 0xfc, 0x00, 0x00, 0x3f, 0xc0, 0x00, 0x00, 0x01, 0xff,
0x80, 0x01, 0xff, 0x80, 0x00,
  0x00, 0x00, 0x7f, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x1f,
0xff, 0xff, 0xf8, 0x00, 0x00,
  0x00, 0x00, 0x03, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00,
0x7f, 0xfe, 0x00, 0x00, 0x00
};
```

From:
https://von-thuelen.de/ - **Christophs DokuWiki**

Permanent link:
https://von-thuelen.de/doku.php/wiki/projekte/co2-ampel/uebersicht

Last update: **2021/10/21 19:04**