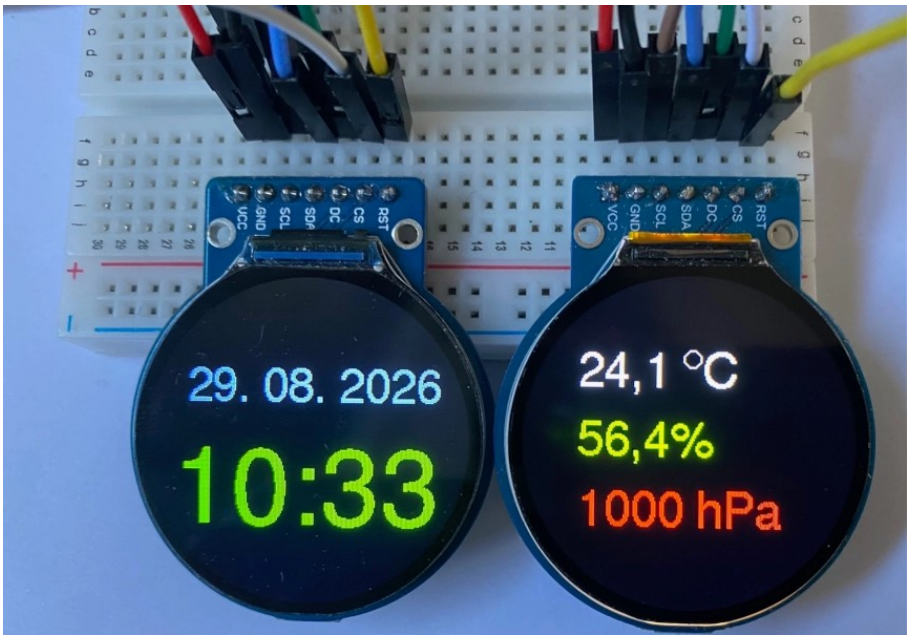
Ziel des Projekts

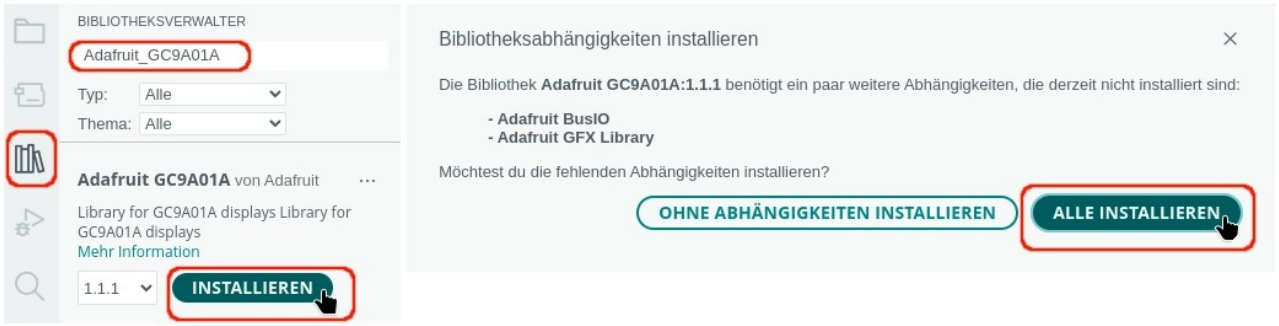
Auf einem TFT wird eine digitale Uhr angezeigt. Auf dem anderen TFT werden die Messwerte eines BME280-Sensors dargestellt. Der Bildschirmaufbau auf dem linken TFT wird mit der GFXcanvas der Bibliothek Adafruit GFX realisiert.



Benötigte Bauteile

- Zwei runde TFT-Module
- BME280-Sensor
- Leitungsdrähte

Benötigte Bibliotheken



Anschluss der TFT-Module

Der Unterschied besteht lediglich in der Zuordnung des CS-Pins.

gelb -> 5 (RST)
weiß -> 6 (TFT_CS_Uhr)
grün -> 4 (DC)
blau -> 8 (SDA -> COPI)
braun -> 10 (SCL -> SCK)
schwarz -> GND
rot -> 5V

gelb -> 5 (RST)
weiß -> 7 (TFT_CS_BME)
grün -> 4 (DC)
blau -> 8 (SDA -> COPI)
braun -> 10 (SCL -> SCK)
schwarz -> GND
rot -> 5V

Das Programm

Bibliotheken und Variable

- Neben den Bibliotheken für das TFT und den BME280 werden die Schriftarten der Bibliothek Adafruit GFX eingebunden.
  Allerdings können sie keine Umlaute und Sonderzeichen darstellen und sind nur bis zur Größe von 24 Punkten verfügbar.
  Auf der Seite ↳https://rop.nl/truetype2gfx/ kannst du von den verfügbaren freien Schriftarten neue Schriftgrößen erstellen. Allerdings verwenden diese Schriften auch nur den einfachen ASCII-Zeichensatz und beinhalten keine Umlaute und Sonderzeichen.
  Die neu erstellte Schrift befindet sich anschließend in deinem Download-Ordner. Jetzt musst du sie noch in den Ordner **Arduino/libraries/Adafruit_GFX_Library/Fonts** kopieren.

- Der SPI-Bus kann mehrere Module gleichzeitig bedienen, allerdings wird dann auf beiden TFTs der gleiche Inhalt angezeigt.
  Neben den eigentlichen SPI-Pins (SLCK, COPI, CIPO), die festen Anschlüssen zugeordnet sind, kann der ChipSelect-Pin (CS) der TFTs unterschiedlichen Pins zugeordnet:
  ```
  #define TFT_CS_Uhr
  #define TFT_CS_BME
  ```
  Für die beiden TFTs werden verschiedene Objekte der Bibliothek definiert:
  ```
  Adafruit_GC9A01A tft_Uhr(TFT_CS_Uhr, TFT_DC);
  Adafruit_GC9A01A tft_BME(TFT_CS_BME, TFT_DC);
  ```
- Der ESP32-C5 Zero weist ebenso wie der ESP32-C6 Zero und der ESP32-S3 eine Besonderheit auf.
  beliebige Pins können als I^2C-Pins definiert werden:
  ```
  #define SDA_PIN 2
  #define SCL_PIN 3
  ```
  Im setup-Teil muss Wire mit den entsprechenden Pins gestartet werden:

`Wire.begin(SDA_PIN, SCL_PIN);`

Bei den anderen Mikrocontrollern entfallen diese Schritte.

```
#include "WiFi.h"
#include "WiFiUdp.h"
#include "Adafruit_GFX.h"
#include "Adafruit_GC9A01A.h"
#include "Adafruit_BME280.h"

// interne Schriften
#include "Fonts/FreeSans24pt7b.h"
#include "Fonts/FreeSans18pt7b.h"
#include "Fonts/FreeSans12pt7b.h"

// mit https://rop.nl/truetype2gfx/ erstellte Schriftgrößen
#include "Fonts/FreeSans32pt7b.h"
#include "Fonts/FreeSans40pt7b.h"

#include "NTP3.h"

// SSID und Passwort des Routers
char Router[] = "Router_SSID";
char Passwort[] = "xxxxxxxx";

WiFiUDP wifiUdp;
NTP3 ntp(wifiUdp);

// Objekt der Bibliothek erstellen
Adafruit_BME280 bme;

// ESP32-WROOM
// #define TFT_CS_Uhr    5
// #define TFT_CS_BME   17
// #define TFT_RST       4
// #define TFT_DC        2

// ESP32-S3
// #define TFT_CS_Uhr   10
// #define TFT_CS_BME   14
// #define TFT_RST       5
// #define TFT_DC        4

// ESP32-C6/ESP32-C6-Zero
// #define TFT_CS_Uhr   18
// #define TFT_CS_BME    6
// #define TFT_RST       3
// #define TFT_DC        2

// ESP32-C3 Zero/ESP32-C3 Super Mini
// #define TFT_CS_Uhr     7
// #define TFT_CS_BME     3
// #define TFT_RST        3
// #define TFT_DC         2

// ESP32-C5 Zero
#define TFT_CS_Uhr     6
```

```
#define TFT_CS_BME     7
#define TFT_RST        5
#define TFT_DC         4

// Pins anpassen
#define SDA_PIN 2
#define SCL_PIN 3

// 16-Bit Canvas definieren
GFXcanvas16 Bereich(320, 240);

Adafruit_GC9A01A tft_Uhr(TFT_CS_Uhr, TFT_DC);
Adafruit_GC9A01A tft_BME(TFT_CS_BME, TFT_DC);

// Farben
#define SCHWARZ     0x0000
#define WEISS       0xFFFF
#define BLAU        0x001F
#define ROT         0xF800
#define GRUEN       0x07E0
#define CYAN        0x07FF
#define MAGENTA     0xF81F
#define GELB        0xFFE0
#define BRAUN       0x9A60
#define GRAU        0x7BEF
#define GRUENGELB   0xB7E0
#define DUNKELCYAN  0x03EF
#define ORANGE      0xFDA0
#define PINK        0xFE19
#define BORDEAUX    0xA000
#define HELLBLAU    0x867D
#define VIOLETT     0x915C
#define SILBER      0xC618
#define GOLD        0xFEA0

// beim Start des loop-Teils wird die Zeit einmalig synchronisiert
volatile bool Start = true;

unsigned long Zeitmessung = 0;
int Sekunden;
```

Der setup-Teil

- Jedes TFT-Modul muss seperat gestartet werden:
  ```
  tft_Uhr.begin();
  tft_BME.begin();
  ```
- die Netzwerkverbindung wird im Stationsmodus hergestellt, setSleep schaltet den Powersafe-Modus des WiFi-Moduls aus, setAutoReconnect versucht eine abgebrochene Verbindung neu aufzubauen:
  ```
  WiFi.begin(Router, Passwort);
  WiFi.mode(WIFI_MODE_STA);
  WiFi.setSleep(false);
  WiFi.setAutoReconnect(true);
  ```
- jede Anweisung zur Konfiguration und zur Bildschirmausgabe muss für jedes TFT individuell definiert werden:
  Beispiele:
  ```
  tft_Uhr.setRotation(2);
  tft_BME.setRotation(2);
  tft_Uhr.setCursor(40, 90);
  tft_Uhr.setFont(&FreeSans12pt7b);
  tft_Uhr.print("WiFi verbinden ...");
  tft_BME.setCursor(40, 120);
  tft_BME.setFont(&FreeSans12pt7b);
  tft_BME.print("BME gestartet");
  ```
- der BME280 muss unter Angabe seiner HEX.Adresse gestartet werden
  bme.begin(0x76)
  Wenn die Initialisierung scheitert, wird eine Meldung ausgegeben und das Proramm beendet.

Wenn der Start erfolgreich war, werden Statusmeldungen angezeigt:
IP-Adresse, verbundener Router und die Signalstärke des WiFi-Netzes

```
void setup()
{
  Serial.begin(9600);
  delay(2000);

  // Wire mit den I2C-Pins starten
  Wire.begin(SDA_PIN, SCL_PIN);
  tft_Uhr.begin();
  tft_BME.begin();

  // Rotation anpassen
  tft_Uhr.setRotation(2);
  tft_Uhr.fillScreen(SCHWARZ);
  tft_Uhr.setCursor(40, 120);
  tft_Uhr.setFont(&FreeSans12pt7b);
  tft_Uhr.print("WiFi verbinden ...");

  tft_BME.setRotation(2);
  tft_BME.fillScreen(SCHWARZ);
  tft_BME.setFont(&FreeSans12pt7b);

  // BME280 starten, bei Misserfolg Meldung anzeigen
  if(!bme.begin(0x76))
  {
    Serial.println("BME280 nicht verbunden");
    Serial.println("Verkabelung und/oder HEX-Adresse prüfen!");
    Serial.println("Start mit möglichen HEX-Adressen:");
    Serial.println("bme.begin(0x76);");
    Serial.println("bme.begin(0x77);");
    Serial.println("Programm wird beendet!");
    tft_BME.fillScreen(SCHWARZ);
    tft_BME.setCursor(20, 90);
    tft_BME.print("BME nicht gestartet");
    tft_BME.setCursor(20, 120);
    tft_BME.print("HEX-Adresse");
    tft_BME.setCursor(20, 150);
    tft_BME.print("korrigieren");
    tft_BME.setCursor(20, 180);
    tft_BME.print("Programm beendet");
    while(1);
  }
  else
  {
    Serial.println("BME280 erfolgreich gestartet!");
    tft_BME.setCursor(40, 120);
    tft_BME.print("BME gestartet");
  }

  delay(1000);

  // WiFi starten DHCP
  WiFi.begin(Router, Passwort);

  WiFi.mode(WIFI_MODE_STA);
  WiFi.setSleep(false);
  WiFi.setAutoReconnect(true);
  ntp.updateInterval(30000);
```

```
  while(WiFi.status() != WL_CONNECTED)
  {
    delay(200);
    Serial.print(".");
  }

  /*
    Zeitzone
    CEST: Central European Summertime
    Beginn europäische Sommerzeit letzter Sonntag im März 2 Uhr GMT + 2 Stunden
  */
  ntp.ruleDST("CEST", Last, Sun, Mar, 2, 120);

  // CET: Central European Time
  // Beginn Normalzeit letzter Sonntag im Oktober 3 Uhr GMT + 1 Stunde
  ntp.ruleSTD("CET", Last, Sun, Oct, 3, 60);

  // ntp starten
  ntp.begin();

  // WiFi-Daten anzeigen
  tft_Uhr.fillScreen(SCHWARZ);
  tft_Uhr.setCursor(40, 70);
  tft_Uhr.print("WiFi verbunden ...");
  tft_Uhr.setCursor(40, 100);
  tft_Uhr.print(WiFi.localIP());
  tft_Uhr.setCursor(40, 130);
  tft_Uhr.print(WiFi.SSID());
  tft_Uhr.setCursor(40, 160);
  tft_Uhr.print(WiFi.RSSI());
  delay(5000);
  Zeitmessung = millis() + 1000;
}
```

Die Funktion Zeitanzeigen()

- die Farben der Bildschirmausgabe können mit der Anweisung setTextColor individuell gestaltet werden
- der Rückgabewert der Aktualisierung der Zeit mit ntp.update() ist im Erfolgsfall true.
  Bei einem Misserfolg versucht das Programm die WiFi-Verbindung neu aufzubauen. Der Wert der Variable Start wird auf true gesetzt, im loop-Teil wird die Zeit sofort aktualisiert.
- ntp.isValid() gibt true zurück, wenn die ermittelte Zeit gültig ist. Ist das nicht der Fall, versucht das Programm die WiFi-Verbindung neu aufzubauen. Die Variable Start wird auf true gesetzt und die Zeit wird sofort aktualisiert.
- Die Messwerte des BME280 werden auf eine Nachkommastelle gekürzt, der Luftduck wird durch die Umwandlung zu int als ganze Zahl dargestellt.
  Im String wird mit replace der Punkt durch ein Komma ersetzt.

```
String Temperatur = String(bme.readTemperature(), 1);
String Luftfeuchtigkeit = String(bme.readHumidity(), 1);;
String Luftdruck = String(int(bme.readPressure() / 100.0));
Temperatur.replace(".", ",");
Luftfeuchtigkeit.replace(".", ",");
```

```
void ZeitAnzeigen()
{
   // Zeit aktualisieren
  if (ntp.update())
  {
    // überprüfen ob die Zeit gültig ist
    if (ntp.isValid())
    {
      Sekunden = ntp.seconds();

      // Canvas erstellen
      Bereich.fillScreen(SCHWARZ);

      // Datum
      Bereich.setCursor(30, 90);
      Bereich.setFont(&FreeSans18pt7b);
      Bereich.setTextColor(HELLBLAU);
      Bereich.print(ntp.formattedTime("%d. %m. %Y"));

      // Zeit
      Bereich.setCursor(20, 180);
      Bereich.setFont(&FreeSans40pt7b);
      Bereich.setTextColor(GRUEN);
      Bereich.print(ntp.formattedTime("%H:%M"));

      // Canvas anzeigen
      tft_Uhr.drawRGBBitmap(1,1, Bereich.getBuffer(), Bereich.width(), Bereich.height());
    }

    // Zeit ist nicht gültig
    else
    {
      // prüfen, ob WiFi verbunden ist
      while (WiFi.status() != WL_CONNECTED)
      {
        delay(200);
        Serial.print(".");
      }
      Start = true;
    }
  }
  // WiFi ist nicht verbunden
  else
  {
    // prüfen, ob WiFi verbunden ist
    while (WiFi.status() != WL_CONNECTED)
    {
      delay(200);
      Serial.print(".");
    }
    Start = true;
  }

  // Messwerte BME280 1 Nachkommastelle
  String Temperatur = String(bme.readTemperature(), 1);
  String Luftfeuchtigkeit = String(bme.readHumidity(), 1);
```

```
  // Luftdruck als ganze Zahl
  String Luftdruck = String(int(bme.readPressure() / 100.0));

  // . durch , ersetzen
  Temperatur.replace(".", ",");
  Luftfeuchtigkeit.replace(".", ",");

  tft_BME.fillScreen(SCHWARZ);
  tft_BME.setCursor(40,80);
  tft_BME.setFont(&FreeSans18pt7b);
  tft_BME.setTextColor(HELLBLAU);
  tft_BME.print(Temperatur);

  // Grad-Zeichen als Kreis
  tft_BME.drawCircle(122, 60, 6, HELLBLAU);
  tft_BME.setCursor(130,80);
  tft_BME.print("C");

  tft_BME.setCursor(40,130);
  tft_BME.setTextColor(GRUEN);
  tft_BME.print(Luftfeuchtigkeit + "%");

  tft_BME.setCursor(40,180);
  tft_BME.setTextColor(ROT);
  tft_BME.print(Luftdruck + " hPa");
}
```

Der loop-Teil

```
void loop()
{
  // Start = true
  // -> Zeit einmalig beim Start synchronisieren
  // danach zur vollen Minute
  if (Start)
  {
    Start = false;
    ZeitAnzeigen();
  }

  // Sekunden weiter zählen
  if (Zeitmessung < millis())
  {
    Zeitmessung += 1000;
    Sekunden++;

    // nach 60 Sekunden Zeit und Messwerte aktualisieren
    if (Sekunden == 60) ZeitAnzeigen();
  }
}
```