# Individual Project

## IT essentials 1ITF

Mathias Wouters 1ITF-13
r-number: r0879842

CAMPUS

**Geel**

**Technology**
**Elektronics-ICT / Applied informatics**

**IT essentials**

Course unit: IT essentials

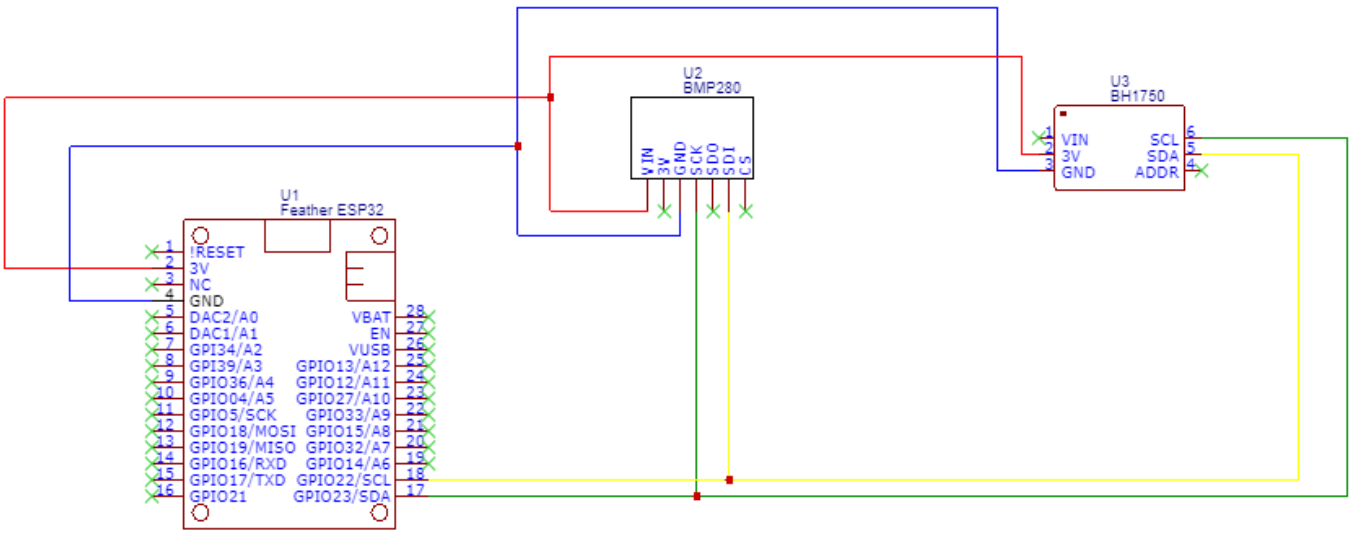
Educational activity: IT essentials

First tier

**Academiejaar 2021-2022**

# Table of Contents

# 1 Hardware




Het ESP32-bordje (U1) wordt gevoed door een transfo via een USB kabel en levert een 3V DC spanning af tussen de 3V (+) en GND (grond) pins.

Omdat de 2 gebruikte sensoren (U2 en U3) zowel op 3V als op 5V kunnen werken, gebruik ik de 3V DC uitgang van het ESP32-bordje om beide sensoren van spanning te voorzien en met de grond te verbinden.

Vervolgens zorg ik ervoor dat meetgegevens van beide sensoren doorgestuurd worden naar het ESP32-bordje. Deze datacommunicatie gebeurt via de I2C-bus. I2C werkt op basis van twee buslijnen: SDA (serial data) en SCL (serial clock). Via de SDA-lijn wordt de data verstuurd en via de SCL-lijn wordt het kloksignaal verzonden om het circuit te synchroniseren. Bij deze communicatie gedraagt de ESP32 zich als master en zijn de sensoren de slaves.

De master heeft de controle over de I2C-bus, genereert het kloksignaal en verstuurd de startbit en de stopbit. De slaves kunnen enkel dan communiceren, als de master daartoe een verzoek verstuurt.

# 2 Software

```
// Load Wi-Fi library
#include <BH1750.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <Wire.h>
#include <WiFi.h>

//uncomment the following lines if you're using SPI
//#include <SPI.h>
#define BME_SCK 18
#define BME_MISO 19
#define BME_MOSI 23
#define BME_CS 5

#define SEALEVELPRESSURE_HPA (1013.25)

BH1750 lightMeter;

Adafruit_BMP280 bme;

// Replace with your network credentials
const char* ssid     = "POCO_X3";
const char* password = "12345678";

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;
```

Eerst heb ik alle nodige libraries toegevoegd, zoals de library voor de 2 sensoren (BMP280 en BH1750) die ik gebruik en de library om de data op een website te krijgen.

Omdat ik het I2C protocol gebruik, moet ik de GPIO-pins op de standaard waarden instellen.

Hierna heb ik een variabele sealevelpressure (druk op 0 m) gemaakt waarmee ik de druk die wordt gemeten door mijn sensor vergelijk om zo de gemiddelde hoogte te kennen.

Om de website te kunnen weergeven moet ik mijn netwerk account gegevens doorgeven (naam + passwoord).

Dan zet ik mijn web server naar poort 80 en maak ik een variabele aan waar de header van het HTTP verzoek in wordt opgeslagen.

```
void setup() {
  Serial.begin(9600);
  Wire.begin();
  lightMeter.begin();
  delay(2000);
  // default settings
  // (you can also pass in a Wire library object like &Wire2)
  //status = bme.begin();
  if (!bme.begin()) {
    Serial.println("Could not find a valid BMP280 sensor, check wiring!");
    while (1);
  }

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}
```

De eerste stap in de setup is de seriale communicatie snelheid op 9600 baud zetten; dit is de snelheid waarmee de data wordt verzonden.

Dan kijk ik of de sensor is geïnitialiseerd en begin ik de wifi connectie om het ip address in de serial monitor te printen

```
67  void loop(){
68    WiFiClient client = server.available();   // Listen for incoming clients
69
70    if (client) {                             // If a new client connects,
71      currentTime = millis();
72      previousTime = currentTime;
73      Serial.println("New Client.");          // print a message out in the serial port
74      String currentLine = "";                // make a String to hold incoming data from the client
75      while (client.connected() && currentTime - previousTime <= timeoutTime) {  // loop while the client's connected
76        currentTime = millis();
77        if (client.available()) {             // if there's bytes to read from the client,
78          char c = client.read();             // read a byte, then
79          Serial.write(c);                    // print it out the serial monitor
80          header += c;
81          if (c == '\n') {                    // if the byte is a newline character
82            // if the current line is blank, you got two newline characters in a row.
83            // that's the end of the client HTTP request, so send a response:
84            if (currentLine.length() == 0) {
85              // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
86              // and a content-type so the client knows what's coming, then a blank line:
87              client.println("HTTP/1.1 200 OK");
88              client.println("Content-type:text/html");
89              client.println("Connection: close");
90              client.println();
91
92              // Display the HTML web page
93              client.println("<!DOCTYPE html><html>");
94              client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
95              client.println("<link rel=\"icon\" href=\"data:,\">");
96              // CSS to style the table
97              client.println("<style>body { text-align: center; font-family: \"Trebuchet MS\", Arial;}");
98              client.println("table { border-collapse: collapse; width:35%; margin-left:auto; margin-right:auto; }");
99              client.println("th { padding: 12px; background-color: #0043af; color: white; }");
100             client.println("tr { border: 1px solid #ddd; padding: 12px; }");
101             client.println("tr:hover { background-color: #bcbcbc; }");
102             client.println("td { border: none; padding: 12px; }");
103             client.println(".sensor { color:white; font-weight: bold; background-color: #bcbcbc; padding: 1px; }");
104
105             // Web Page Heading
106             client.println("</style></head><body><h1>ESP32 with BME280</h1>");
107             client.println("<table><tr><th>MEASUREMENT</th><th>VALUE</th></tr>");
108             client.println("<tr><td>Temp. Celsius</td><td><span class=\"sensor\">");
109             client.println(bme.readTemperature());
110             client.println(" *C</span></td></tr>");
111             client.println("<tr><td>Light</td><td><span class=\"sensor\">");
112             client.println(lightMeter.readLightLevel());
113             client.println(" lux</span></td></tr>");
114             client.println("<tr><td>Pressure</td><td><span class=\"sensor\">");
115             client.println(bme.readPressure() / 100.0F);
116             client.println(" hPa</span></td></tr>");
117             client.println("<tr><td>Approx. Altitude</td><td><span class=\"sensor\">");
118             client.println(bme.readAltitude(SEALEVELPRESSURE_HPA));
119             client.println(" m</span></td></tr>");
120
121             // The HTTP response ends with another blank line
122             client.println();
123             // Break out of the while loop
124             break;
125           } else { // if you got a newline, then clear currentLine
126             currentLine = "";
127           }
128         } else if (c != '\r') {  // if you got anything else but a carriage return character,
129           currentLine += c;      // add it to the end of the currentLine
130         }
131       }
132     }
133     // Clear the header variable
134     header = "";
135     // Close the connection
136     client.stop();
137     Serial.println("Client disconnected.");
138     Serial.println("");
139   }
140 }
```

De eerste lijn in de loop kijkt of er eventueel nieuwe clients zijn

Zodra er een nieuwe client is verbonden wordt de binnenkomende data opgeslagen.

Deze while loop blijft lopen tot de client niet meer geconnecteerd is.

In dit stuk code stuur ik mijn html code door naar de client waardoor deze een web pagina kan genereren.

In die html code zitten ook de gegevens die mijn sensoren meten.

Zodra de client stopt, wordt de variabele met de html header gewist en wordt de connectie stop gezet.

# 3 Aanpak

Als eerste stap ben ik gaan kijken of mijn ESP32-bordje werkte door het maken,laden en uitvoeren van een programma dat een ledje op het bordje laat flikkeren. Zodra dat werkte, heb ik mijn schakeling op het breadboard uitgebreid met beide sensoren. Vervolgens heb ik het gekregen programma voor de licht sensor geladen en getest. Daarna heb ik zelf de code ontwikkeld en toegevoegd om ook de andere sensor te laten werken (11/20 → werkt).

Als volgende stap heb ik dan geprobeerd om een web server te maken met het ESP bordje. Het vinden van de juiste instellingen voor de communicatiesnelheid bleek een probleem, maar eenmaal dit was opgelost, verwachtte ik dat het zou moeten werken. Dit gebeurde initieel niet omdat ik verbonden was met de wifi van de school. Daarna heb ik hetzelfde gedaan met mijn GSM als hotspot en toen werkte het wel. Vervolgens heb ik met HTML en CSS een tabel gemaakt waarin de gegevens van beide sensorsen komen (14/20 → werkt).

## Besluit

Omdat ik voordien (school en vrije tijd) reeds bezig ben geweest met Arduino en Raspberry-Pi was alles niet volledig nieuw voor mij. Toch heb ik tijdens dit projectje nieuwe dingen geleerd en de nodige problemen moeten aanpakken en oplossen. Ik vind het plezant om een deel van de leerstof beter te begrijpen via zo'n praktisch project.

**YouTube Filmpje:**

https://youtu.be/zEYegPzMROw

# Literatuurlijst


- Santos, S. (2020, 30 juli). *ESP32 Web Server with BME280 – Advanced Weather Station*. Random Nerd Tutorials. Geraadpleegd op 23 december 2021, van https://randomnerdtutorials.com/esp32-web-server-with-bme280-mini-weather-station/
- I2C.pptx (Thomas More)
- PlatformIO.pptx (Thomas More)
- Project_v3.pptx (Thomas More)