**evaluatie week 7: Freestyle IoT project:**

Maak zelf een **ORIGINELE IoT applicatie** naar keuze.

**Minimum vereisten:**

* Het project moet ergens **data verzamelen via sensoren.**
* Het project moet deze **data verwerken** en op basis van de ingelezen data iets regelen of uitvoeren.
* Het project moet **externe hardware aansturen** (lampjes, motoren, geluid, deurslot, verwarming,…)
* Het project heeft bij voorkeur zowel een lokale display ter controle van de parameters als een **remote control via webserver en/of Blynk app**.
* Het project moet **volledig gedocumenteerd zijn op je GitHub pagina** zodat anderen het kunnen nabouwen.
* Het project moet door jezelf voorgesteld worden in een **video met demo.**

**Op te leveren:**

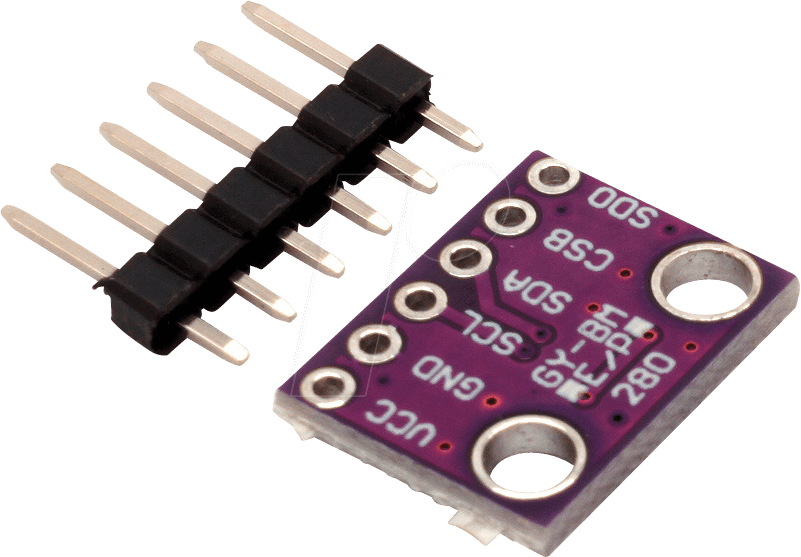
Een beschrijving van je project (wat gaat het doen, welke hardware gebruikt het)

Ik heb een verwarming gemaakt die ik kan regelen lokaal en op afstand via mijn app op mijn gsm en door behulp van een lampje laat zien wanneer de airco werkt, de hardware vind u hier beneden.

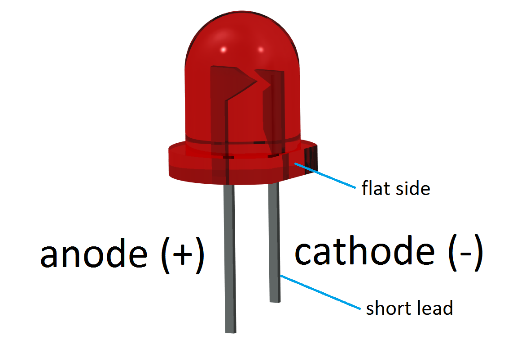
Esp32



BMP 280



LED



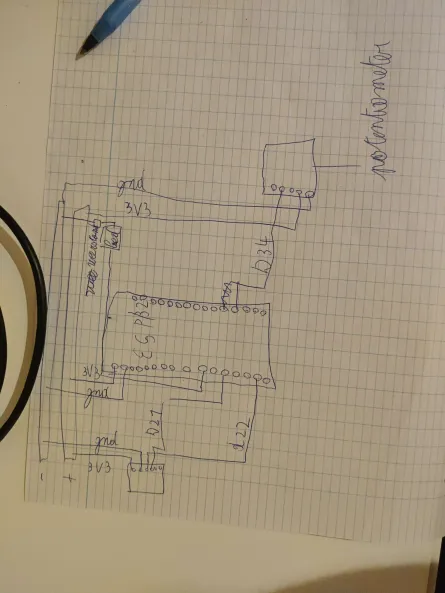
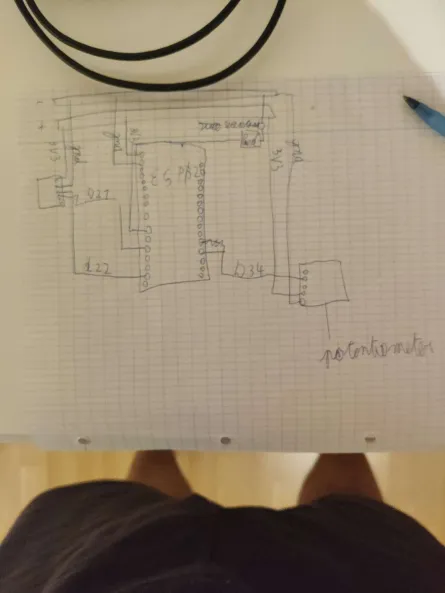
POTENTIOMETER



Link naar je Github pagina. Vergeet deze niet open te zetten !

Gedetailleerde beschrijving van het project op je github. Schema’s, software, foto’s van de opstelling screenshots van de app of webbrowser en documentatie. Schrijf veel commentaar in je software zelf !! Elke lezer moet je redenering kunnen volgen.

1. Schema



1. Uitleg +code:

**#define BLYNK\_TEMPLATE\_ID "user20"**

**#define BLYNK\_TEMPLATE\_NAME "user20@server.wyns.it"**

**#define BLYNK\_PRINT Serial**

Deze regels definiëren en configureren de Blynk-template-ID en -naam, die worden gebruikt voor het delen van Blynk-projecten. Ook wordt **BLYNK\_PRINT** geconfigureerd om seriële uitvoer te gebruiken voor het debuggen.

**#include <WiFi.h>**

**#include <WiFiClient.h>**

**#include <BlynkSimpleEsp32.h>**

Deze regels voegen de nodige bibliotheken toe voor WiFi-verbindingen en communicatie met het Blynk-platform.

**#include <Wire.h>**

**#include <SPI.h>**

**#include <Adafruit\_BMP280.h>**

Hier worden bibliotheken ingevoegd voor het communiceren met de BMP280-temperatuursensor.

**Adafruit\_BMP280 bmp;**

**BlynkTimer timer;**

**WidgetLED led1(V6);**

Hier worden objecten geïnstantieerd voor de BMP280-sensor, de Blynk-timer en een LED-widget op het Blynk-dashboard.

**char auth[] = "QddllR7eEuSt7dBJjC6xwHAccHH\_4frN";**

**char ssid[] = "telenet-B115D97";**

**char pass[] = "wkd3ucpeXfch";**

Deze regels definiëren de authenticatietoken voor Blynk en de WiFi-gegevens (SSID en wachtwoord) voor de ESP32 om verbinding te maken met het WiFi-netwerk.

**#define BMP\_SCK (13)**

**#define BMP\_MISO (12)**

**#define BMP\_MOSI (11)**

**#define BMP\_CS (10)**

**#define led 18**

Dit zijn de pindefinities voor de communicatie met de BMP280-sensor en de LED.

**int gewensteWaarde;**

**int pot2 = 0.0;**

**int ap1 = 0.0;**

**int ap2 = 0.0;**

**int POT1 = 0.0;**

**float Val = 0;**

**float pot = 34;**

Hier worden verschillende variabelen gedeclareerd voor het opslaan van sensorwaarden, instellingen en communicatie met het Blynk-platform.

**void sendSensor() {**

**float t = bmp.readTemperature();**

**float tt = int(t \* 10) / 10.0;**

**Blynk.virtualWrite(V5, tt);**

**}**

Dit is een functie genaamd **sendSensor()**, die de temperatuur van de BMP280-sensor leest en naar het virtuele pin V5 op het Blynk-dashboard stuurt.

**#define gewensteWaarde V7**

Hier wordt een virtuele pin V7 gedefinieerd als gewensteWaarde.

**BLYNK\_WRITE(gewensteWaarde) {**

**int gt = param.asInt();**

**Val = gt;**

**Serial.print("Gewenste waarde ingesteld op: ");**

**Serial.println(gt);**

**}**

Dit is een functie die wordt aangeroepen wanneer de waarde van het virtuele pin V7 op het Blynk-dashboard wordt gewijzigd. Het ontvangt de nieuwe waarde en drukt deze af op de seriële monitor.

**void setup() {**

**Serial.begin(115200);**

**Serial.println(F("BMP280 test"));**

**pinMode(led, OUTPUT);**

**if (!bmp.begin(0x76)) {**

**Serial.println(F("Could not find a valid BMP280 sensor, check wiring or try a different address!"));**

**while (1)**

**;**

**}**

**delay(10);**

**Serial.print("Connecting to ");**

**Serial.println(ssid);**

**WiFi.begin(ssid, pass);**

**int wifi\_ctr = 0;**

**while (WiFi.status() != WL\_CONNECTED) {**

**delay(500);**

**Serial.print(".");**

**}**

**Serial.println("WiFi connected");**

**Blynk.begin(auth, ssid, pass, "server.wyns.it", 8081);**

**timer.setInterval(1000L, sendSensor);**

**}**

Dit is de **setup()** functie, die wordt uitgevoerd bij het opstarten van de ESP32. Het initialiseert seriële communicatie, de BMP280-sensor, de LED-pin, WiFi-verbinding en de Blynk-communicatie. Ook wordt een periodieke timer ingesteld om de **sendSensor()** functie elke seconde uit te voeren.

**void loop() {**

**Blynk.run();**

**timer.run();**

**float t = bmp.readTemperature();**

**float potvalue = analogRead(pot);**

**POT1 = map(potvalue, 0, 4095, 0, 30);**

**if (POT1 != pot2) {**

**pot2 = POT1;**

**Blynk.virtualWrite(V7, pot2);**

**t = pot2;**

**}**

**if (ap2 != ap1) {**

**ap2 = ap1;**

**Blynk.virtualWrite(V7, ap2);**

**t = ap2;**

**}**

**if (t <= Val) {**

**digitalWrite(led, HIGH);**

**led1.on();**

**}**

**else {**

**digitalWrite(led, LOW);**

**led1.off();**

**}**

**Serial.print(F("Temperature = "));**

**Serial.print(bmp.readTemperature());**

**Serial.println(" \*C");**

**Serial.println();**

**delay(1000);**

**}**

Dit is de **loop()** functie, die continu wordt uitgevoerd na de **setup()** functie. Het beheert de communicatie met het Blynk-platform, voert periodieke taken uit, zoals het lezen van de temperatuur van de sensor en het aanpassen van de LED-status op basis van de temperatuur. Ook worden de huidige temperatuurwaarden afgedrukt op de seriële monitor.

**BLYNK\_WRITE(V6) {**

**float value = param.asFloat();**

**Serial.print("float value received from Blynk app: ");**

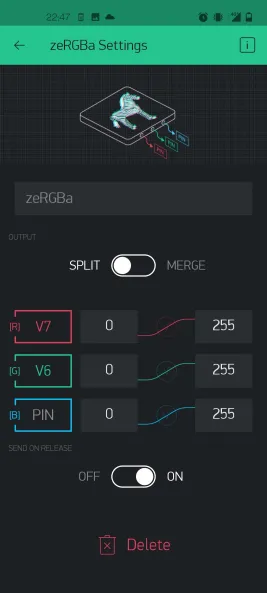
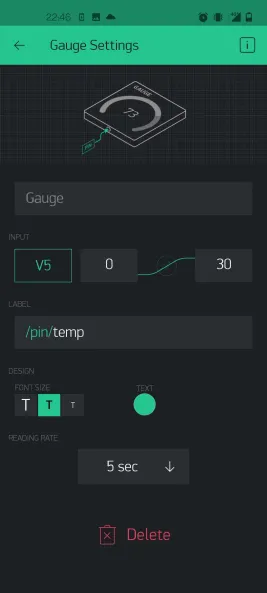
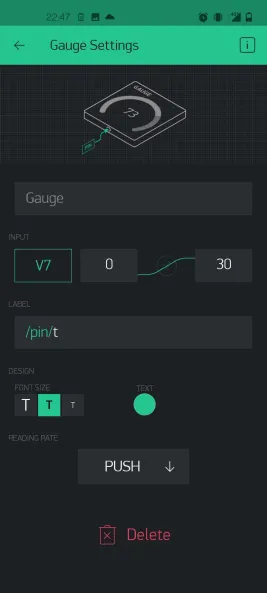
**Serial.print(value);**

**}**

Dit is een functie die wordt aangeroepen wanneer een waarde wordt ontvangen op het virtuele pin V6 op het Blynk-dashboard. Het ontvangt de waarde en drukt deze af op de seriële monitor.

1. Uitleg + blynk:

Hier vind u mijn blynk aplicatie die alles van op afstand regelt aan de linker bovenkant vind u de temperatuur die sensor meet en onvangt en doorstuurt op deze gauge. Op de rechterbovenkant vind u de koeling die verbonden is met de slider daaronder. De slider kan de gauge aanpassen en is ook verbonden met de zebra daaronder dat het led lampje representeerd wanneer het koud is het donkerblauw wanneer het te heet is dan is het groen en dit is mijn blynk aplicatie , als u de lokale potentiometer gebruikt dan past de slider op de app zich ook automatisch aan.

**Afbeelding met tekst, schermopname, Graphics, grafische vormgeving

Automatisch gegenereerde beschrijving**

Een filmpje waarin je kort het project uitlegt, even stilstaat bij de problemen die je bent tegengekomen en hoe je deze hebt opgelost en tot slot een demo waar je duidelijk toont hoe het werkt en of het werkt. Het filmpje publiceer je op je Github of youtube en je voorziet de link hieronder op **dit document**, dat je oplaadt in de uploadzone voor de deadline.

ik heb de video rechtstreeks geüpload want ik wouw niks op youtube zetten.

**Link naar Github :**

<https://github.com/zaidben32/eindprojectweek7>

**Puntenverdeling:**

Hardware design en documentatie hiervan 10 punten

App design en screenshots hiervan 10 punten

Software listing en documentatie 10 punten

Filmpje dat concept en werking toont 20 punten

**Timing:**

**Deadline maandag 15 april 2024 23:59**

Maandag 25 maart kan je terecht in het labo IoT voor hulp.

Maandag 15 april is het labo gesloten maar kan je nog wel vragen stellen via email of chat.