THOMAS MORE

EINDOPDRACHT LYAM TEIJSSEN

Temperatuur data monitor en displays

2023 - 2024

Inhoud

1. Project beschrijving

1.1. Doel van het project

Het doel van dit project is de temperaturen van de lucht in en rond de pc te kunnen meten zodat je niet alleen weet hoe warm de componenten zelf zijn maar ook weet wat de invloed van temperatuur is op en door de componenten. Zo weet je dus hoeveel invloed gamen heeft op de temperatuur rondom je maar ook andersom, als het warm is weet je exact hoe warm het is en kan je dus ook goed zien wat voor effect dat exact heeft op je computer zijn componenten.

1.2. Extra's

Als een kleine toevoeging heb ik een extra display toegevoegd met knoppen waarop het mogelijk is de instellingen van het andere scherm te veranderen zonder je telefoon met de app steeds bij te moeten hebben.

2. Hoe ben ik tewerk gegaan?

Ik ben begonnen met het uittesten van mijn bestaande MQTT-installatie die gedaan was tijdens de lessen. Na het te testen wist ik dat deze ook werkt op het lokale netwerk (over ethernet i.p.v. Wi-Fi).

Daarna ben ik begonnen aan de code voor het OLED-schermpje op de grafische kaart. Ik ben op zoek moeten gaan voor een goede library om dit aan te sturen. Omdat het een goedkoop OLED-schermpje is van Amazon zaten er geen instructies of aanwijzingen bij, maar ben ik er uiteindelijk achter gekomen dat de 'U8g2' library er mee werkt omdat het een I2C communicatie bus heeft.
Uiteindelijk werkte dat schermpje en gaf het de tekst die ik manueel had ingegeven in de code. Daarna heb ik de code voor de MQTT-server toegevoegd zodat deze ook berichten kon aankrijgen en dus data kan weergeven en/of instellingen remote kan aanpassen.
Dat gebeurt aan de hand van berichten in verschillende topics zoals 'pc/temp/exhaust' voor de temperatuur achteraan mijn desktop pc en 'pc/screen/gpu' voor de instellingen van wat het scherm weergeeft aan te kunnen passen.

Daarna ben ik begonnen met het schrijven van de code voor het uitlezen van de temperatuur met een BMP280 sensor. Daar heb ik meteen de code voor de MQTT-verbinding in geschreven en de sensor de temperatuur laten publiceren.
Na dit even te testen met het schermpje was ik er zeker van dat het werkte en heb ik een pauze genomen om even van de vakantie te genieten.

Enkele dagen later ben ik weer aan de slag gegaan en heb ik een ESP32 gebruikt in combinatie met een 20x4 lcd-scherm met I2C om een besturingspaneeltje te maken. Daar ben ik dan de code voor beginnen schrijven en ben ik enkele dagen bezig geweest met de

bugs eruit te halen tot ik het uiteindelijk volledig herwerkt heb omdat het maar niet lukte. Dat heeft goed gewerkt want nu doet het besturingspaneeltje het. Met iets eenvoudigere besturing dan ervoor is nu het OLED-schermpje in te stellen met dit besturingspaneeltje en is er de mogelijkheid meer menu's toe te voegen omdat dit modulair is opgebouwd.

Daarna heb ik de vertaling voor de Blynk app proberen toevoegen op een aparte D1 mini. Dit is uiteindelijk niet gelukt omdat de berichten simpelweg niet werden ontvangen op de D1 mini en heb ik beslist om deze code te integreren in de bestaande D1 mini die de temperatuursensor leest. Dat heb ik dan weer uitgetest en na meerdere kleine fouten en problemen op te lossen werkte dit ook zeer goed en kon ik het remote schermpje aanpassen vanuit de Blynk app.

Daarna moest ik nog even de python code die al deels in de lessen gemaakt was herwerken om zo de nieuwe topics in de database op te slaan zodat ook op de Grafana dashboard de data te lezen was. Dit was vrij snel opgelost maar daar zat ik toch even aan om er dan achter te komen dat het te maken had met de regex expressie die niet meer juist was.

Uiteindelijk werkte dit allemaal en heb ik nog een tweede D1 mini met een BMP280 sensor aangesloten en daar ongeveer dezelfde code opgezet als de eerste temperatuursensor maar dan zonder de vertaling van de app (die hoeft geen 2 keer te gebeuren) en met als onderwerp 'pc/temp/intake' in de plaats van 'pc/temp/exhaust' zodat deze twee gescheiden blijven in de database en op het display.

3. Beschrijving van de code

3.1. Code van de D1 mini temperatuursensor en Blynk vertaling

De code van de D1 mini (te vinden op de GitHub pagina onder `/ESP8266/WeMos D1 mini Lite/ExhaustTemperatureSensorBMP280/`) begint zoals alle anderen met het specifiëren van de libraries en in dit geval ook de Blynk template ID en name.
Vervolgens zetten we alle belangrijke benodigde dingen in 'char' variabelen zodat deze hier gemakkelijk kunnen aangepast worden indien nodig en voor anderen die deze code downloaden en zelf willen gebruiken.
Vervolgens komen er nog enkele definities zoals de BMP280 sensor en de 'millis' voor het bijhouden wanneer er data moet gepubliceerd worden.

Daarna komen de definities van de custom functies zoals 'reconnect' en 'callback' om met de MQTT-server verbonden te blijven en de berichten bij het ontvangen correct om te zetten en dergelijke.

Daarna komt de setup fase waarin effectief met de wifi verbonden wordt, de connecties met de MQTT-server en Blynk worden gestart voor de eerste keer, er op de juiste topics wordt 'gesubscribed' en als laatste wordt gecheckt of de BMP280 sensor correct is aangesloten en gevonden wordt.

Daarna in de loop worden simpelweg de vorige functies zoals reconnect opgeroepen wanneer er geen verbinding meer is en de temperatuur wordt uitgelezen en opgeslagen om door te sturen.

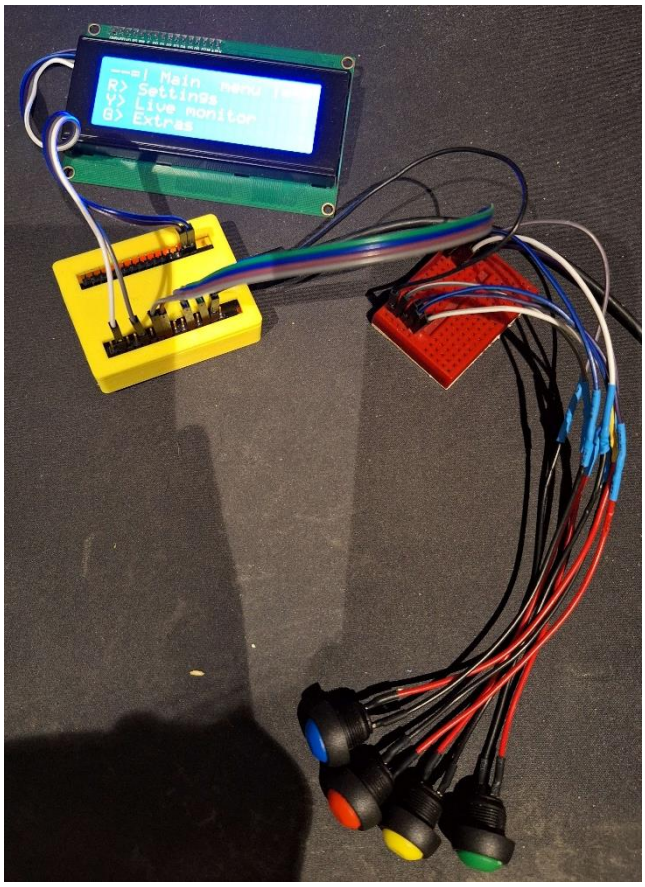
Onder de loop staat nog code om de knoppen in de Blynk app te kunnen uitlezen, deze stukjes worden elks uitgevoerd enkel wanneer er een verandering is van de virtuele waardes (V1, V2 en V3).

3.2. Code van de D1 mini temperatuursensor (standalone)

Deze code is zo goed als exact hetzelfde als de bovenstaande code maar dan zonder de Blynk app onderdelen. Het is dus een stukje korter maar erg gelijkaardig in werking. Deze code kan je vinden onder `/ESP8266/WeMos D1 mini Lite/GeneralTemperatureSensorBMP280/`

3.3. Code van de ESP32 ‘command center’

Een foto van deze opstelling:



De code voor de ESP32 is gemaakt om te werken met een 20 tekens 4 rijen lcd-display met een I2C module. (Deze code is te vinden onder `Temperature-Monitoring-System/ESP32/DOIT ESP32 DEVKIT V1/Settings and Monitor Module/`)

Ook deze code begint met het definiëren van de benodigde libraries en vervolgens de belangrijke variabelen zoals de Wi-Fi naam, wachtwoord, etc.
Daarna initialiseren we de lcd en de ‘PubSubClient’ ofwel de MQTT-verbinding.

Dan komen er enkele definities van lijsten, deze bevatten de menukeuzes zodat deze later nog altijd gemakkelijk aan te passen zijn.

Alles in deze code is zo modulair mogelijk gemaakt om het gemakkelijk uitbreidbaar te maken.

Daarna definiëren we de pinnen waarop onze knoppen zitten (knoppen A, B, X en Y zijn Groen, Rood, Blauw en Geel respectievelijk, dat heb ik gebaseerd op de kleuren van een klassieke Xbox controller). Ook definiëren we variabelen waarin we bij gaan houden of de knoppen ingedrukt zijn en of ze ervoor al ingedrukt waren of niet om zo het probleem van meerdere ‘drukken’ in een keer te vermijden.

Daarna zetten we in de setup alle knoppen als ‘INPUT_PULLUP’ om de interne pullup weerstanden van de ESP te gebruiken om te kijken of de knoppen ingedrukt zijn of niet. Ook starten we de lcd en de andere benodigde verbindingen en printen we het start menu voor de eerste keer op het lcd.

De loop bevat best weinig omdat ik gebruik gemaakt heb van functies om alles een beetje net te houden. We beginnen met het checken of we nog verbonden zijn met de MQTT-server, is dat niet dan herverbinden we met de reconnect functie.
Daarna voeren we de buttonchecks uit, deze functie kijkt of de knop nog niet ingedrukt was en nu wel, houdt dat dan bij en voert de code 1 keer uit. Deze code kijkt dan in welk menu we momenteel zitten en voert afhankelijk daarvan uit wat er moet gebeuren in dat menu.
Elk menu heeft ook een eigen functie om de titel te printen en de opties van elk menu worden allemaal met eenzelfde functie getoond op het scherm.
Na de loop is er een functie ‘menuPrint’ die dat uitvoert, afhankelijk van het menu roept deze de juiste titel op en print die de juiste opties of waardes.

3.4. Code van de Arduino Uno R4 Wifi (remote) display in de desktop

Een foto van dit display:

Ook deze code begint weer met het definiëren van de libraries en de variabelen zoals Wi-Fi naam en wachtwoord. Maar ook definiëren we meteen het OLED-scherm met de U8g2 library.

We definiëren erna een variabele om bij te houden in welke modus het scherm staat en om de laatst aangekregen temperaturen bij te houden en te displayen.

Ook hier hebben we weer deze custom functies om met de Wi-Fi te verbinden, etc.
Deze worden dan ook weer opgeroepen in de setup en in de loop om te herverbinden indien nodig.

De ‘display’ functie verwerkt alle input en zet het om naar wat er op het display getoond moet worden.
De eerdere ‘callback’ functie krijgt effectief de data binnen en zet deze in de variabelen die de ‘display’ functie dan gebruikt om de data te tonen op het scherm.

4. Controle van de werking van MQTT

Dit heb ik gedaan door een eenvoudige versie van de code op een microcontroller te uploaden en gewoon de aangekregen berichten op de seriële monitor te tonen, om vervolgens een keer manueel een publish te doen en eens dat werkte enkele automatische met de D1 mini temperatuursensor.

Als dit niet zou werken is het belangrijk na te kijken dat de microcontroller op de juiste onderwerpen gesubscribed is en met de Wi-Fi verbonden is. Als dat het geval is moeten de permissies van de aangemaakte user gecontroleerd worden. Meer info daarover kan je online vinden.

5. Links

YouTube Video: https://www.youtube.com/watch?v=ICydfGtEoto
GitHub Pagina: https://github.com/SubzeV/Temperature-Monitoring-System

CONTACT

Lyam Teijssen | Student
r0983995@student.thomasmore.be

VOLG MIJ

https://github.com/SubzeV
https://www.youtube.com/@trainspotters

THOMAS MORE