# Temperatuur, vochtigheid en co2 meten

A device with a wire

Description automatically generated

Voor deze oefening maken we gebruik van Visual studio code en de PlatformIO extensie. Dit is de simpelste manier van werken, omdat je alles vanuit een grafische omgeving kan doen.

Op de afbeelding hierboven kan je zie dat de sensor CO2 (936 ppm), temperatuur (39 °C) en vochtigheid (33 %) meet.

Het doel is om een dashboard te maken, waar je de metingen van de sensor mooi kan opvolgen:

A screenshot of a computer

Description automatically generated

1. **Voorbereiding van de ontwikkelomgeving**

Download Visual studio code via de blauwe knop van de website: <https://code.visualstudio.com>. Er is voor elk veelvoorkomend besturingssysteem wel een download.

Als je dit al staan hebt op je computer moet je dit natuurlijk niet opnieuw installeren. Dan moet je juist nog maar de extensie installeren.

Wanneer Visual Studio code geïnstalleerd is ga je naar “Extensions” (het icoontje met de 4 vierkantjes in de linker balk).

En daar zoek je voor “PlatformIO IDE” en klik op Install. Deze extensie heeft het symbool van een alien. Het project dat wij zullen aanmaken is gebaseerd op deze extensie.

A screen shot of a logo

Description automatically generated

Je kan dit ook zonder Visual studio maken, alleen heb je dan geen grafische omgeving. In dat geval moet je PlatformIO Core (CLI) downladen (<https://docs.platformio.org/en/latest/core/index.html>).

1. **Het project zelf aanmaken**

Als dit gebeurd is komt er in de linker balk onder “Extentions” het icoon van PlatformIO bij. Dit is het icoon met een alien op. Klik hierop en dan kom je op de homepagina.

A screenshot of a computer

Description automatically generated

Geef je project een duidelijke naam zoals bv. ControlCO2. Dit kan eender wat zijn zolang dit maar voor jouw duidelijk is.

Als board neem je “SparkFun ESP32 Thing (SparkFun Electronics) en Framework “Arduino”. Dit is belanrijk dat dit juist is, want op basis hiervan weel PlatformIO welke tools en frameworks er nodig is om het project correct te laten werken. **Controleer dit dus dat dit is zoals in deze screenshot!**

A screenshot of a computer program

Description automatically generated

Het vinkje “Use default location” mag je aan laten staan. Dan wordt dit aangemaakt in de standaard map: *C:\Users\<Gebruikersnaam>\Documents\PlatformIO\Projects* Maar ik heb de voorkeur om dit op mijn Desktop te zetten, omdat dit gemakkelijker werkt. En je kan dan snel aan alle bestanden.

Als je dit hebt aangemaakt krijg je een leeg project. Het kan zijn dat dit een tijdje duurt, dit is normaal. PlatformIO moet immers nog die tools en frameworks installeren.

Het belangrijkste bestand is platformio.ini. Hier staat de configuratie van dit project. **Dit bestand kan je zien als de motor van je auto, hierin staat wat er allemaal nodig is om dit project uit te voeren. Zonder dit start de auto niet.**

Omdat we met Visual studio code werken kan je alles via een grafische omgeving doen en wordt dit automatische geüpdatet. Je kan hier echter wel eens in kijken uit nieuwsgierigheid, maar telkens je hier iets aan veranderd moet je het project opnieuw bouwen. Indien we zonder Visual studio werken zouden we dit idere keer manueel moeten aanpassen.

Aan deze structuur moeten we niks veranderen. Dit is de standaard structuur, zoals elk PlatformIO project is opgebouwd.

Het enige wat we nodig hebben is de broncode van deze sensor. Maak een nieuwe map met naam operame aan.

Gaan nu in de browser naar de GitHub repository van deze sensor: <https://github.com/controlco2/operame>.

Download de ino-file “operame.ino”. En zet dit in de operame map die je net hebt aangemaakt. Doordat dit een ino-file is moeten we dit in een map met diezelfde naam zetten, anders wordt dit ino bestand niet goed uitgelezen.

Ga nu in de browser, in diezelfde repository naar het bestand “operame\_string.h” en “logo.h” en sla dit op in de *include* directory van jouw project.

Nu zou je dit moeten hebben. Let daarbij op de ino extensie van operame.ino en de h extensie van logo.h en operame\_strings.h. Als de extensie niet klopt kan je dit eventueel nog hernoemen in Visual studio code.

A screenshot of a computer

Description automatically generatedDe rest van deze mappen hebben we in principe niet nodig en kan je verwijderen. Maar ik laat dit staan ter referentie. Als je wil kan je uit nieuwsgierigheid hier ook eens in kijken.

Je kan ook eens kijken naar wat er in dit ino bestand staat, maar in principe moet je hier niks aan aanpassen.

A black background with white text

Description automatically generatedWel interessant is om te zien hoe dit de meetwaarden verstuurt. Dit kan met MQTT of met REST.

A screen shot of a computer program

Description automatically generated

1. **Het project configureren**

Nu heb je in principe alles om dit te kunnen bouwen. Alleen moet je nog 2 parameters configureren omdat de code in de “src” map anders wordt uitgevoerd. PlatformIO voert standaard de “src” map uit en niet de “operame” map.

Klik op hat alien icoon, ga naar de Project pagina, en klik op “configure”.

A black rectangular object with a black stripe

Description automatically generated

Op het tabblad PIO Configuration, voeg daar een nieuwe optie toe: src\_dir met als waarde het pad naar de operame map. Bij mij is dit *C:\Users\jarne\Desktop\ControlCO2\operame*, maar bij jullie kan dit uiteraard anders zijn.

**Vergeet niet op opslaan te drukken (= “Save”). Anders wordt dit niet bewaard.**

A black rectangular object with a black border

Description automatically generated

Om het project uit te voeren hebben we nog een aantal bibliotheken nodig. Ga naar de sectie common configuratie en voeg de optie lib\_deps toe.

A black screen with white text

Description automatically generated

Klik nu op Add library en voeg deze toe:

A black screen with white text

Description automatically generated

Als je deze niet direct vindt, kan je dit ook rechtstreeks in platfromio.ini plakken. A computer screen shot of white text

Description automatically generated

**Vergeet ook nu niet op te slaan.** Of dit manueel in platformio.ini aan te passen.

1. **Het project uploaden naar de sensor.**

Sluit nu de sensor met een usb-kabel aan op je computer, we gaan dit nu uploaden naar de sensor. Als je dit hebt gedaan komt er op het devices tabblad een device bij:

A screenshot of a computer

Description automatically generated

In mijn geval heb ik de USB-kabel op COM3 aangesloten. Bij jullie kan dit een andere poort zijn. Maar voor USB-poorten is altijd van het formaat: COM + getal.

Als je dit niet ziet kan je eventueel een andere USB-kabel gebruiken, of de drivers van je computer up te daten.

Ga nu terug naar de configuratie van het project. En voeg op het tabblad Env nog de parameter upload\_port toe. Deze moet hetzelfde zijn als wat er bij Devices staat.

A screenshot of a computer

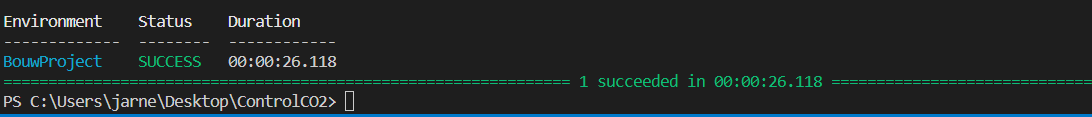
Description automatically generated

Open een terminal (rechter muisklik -> Open in Integrated terminal) in diezelfde directory als het project en voer het commando “pio run” uit.

A black background with white text

Description automatically generated

Dit zal veel output produceren, maar hier hoef je niks van aan te trekken. Het belangrijkste is dat er op het einde een succes status te zien krijgt. Nu heb je dit geüpload naar de sensor.



Je kan ook in de output zien dat dit effectief in het geheugen van de sensor is geschreven:

A screen shot of a computer screen

Description automatically generated

1. **De instellingen van de sensor**

Druk nu de 2 knoppen op de sensor in. Dit is om het te resetten.

Neem nu je smartphone erbij en verbind met het wifi netwerk zoals aangegeven op het scherm. Voor deze sensor is dit het WiFi netwerk “operame-3ce41d”.

A close up of a device

Description automatically generated

Ga nu in de browser van je smartphone naar 192.168.4.1 en je komt op het configuratie menu van de sensor.

Voeg daar de instellingen voor het wifi netwerk en voor MQTT in.

Zorg er zeker voor dat het vinkje “WiFi verbinding gebruiken” en “Activeer MQTT publicatie is aangevinkt”, anders zal dit niet werken.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedOok het vinkje “Activeer MQTT publicatie …” moet aanstaan, anders zendt dit alleen maar de CO2 door.

In het veld Broker heb ik het IP-adres van een mosquitto MQTT-broker ingevoerd. Deze heb ik geïnstalleerd op een raspberry pi.

Klik vervolgens op opslaan en op restart. Eenmaal dit herstart is zal dit waarden versturen naar de mosquitto broker.

1. **Het dashboard maken in Zabbix**

Nu moeten we dit nog in een dashboard krijgen. Het is immers de bedoeling dat we dit kunnen opvolgen.

Hiervoor heb ik Zabbix op de raspberry pi geïnstalleerd. Dit is een monitoringsysteem waarmee je ook dashboards kan maken. Ga naar het IP-adres weer dit draait en voeg /zabbix eraan toe. In mijn geval is dit http:/192.168.0.226/zabbix. Maar je moet mij maar vragen voor het correcte IP-dress, op elk WiFi-netwerk is dit anders.

A screenshot of a login form

Description automatically generated

Log in met gebruikersnaam “Admin” en wachtwoord “zabbix”. Opgelet dit is hoofdlettergevoelig! In Zabbix kan je ook wat rondsnuffelen uit nieuwsgierigheid. Zo kan je bijvoorbeeld bij Templates zien wat er allemaal mogelijk is.

Bij latest Data kan je de waarden voor deze sensor zien.

A white line on a white surface

Description automatically generated

Naar het dashboard CoderDojo gaan. Dit is het dashboard voor deze sensor.

A screenshot of a computer

Description automatically generated

Als alles goed gaat kan je nu hier waarden voor Temperatuur, CO2 en vochtigheid zien. Je kan hier ook wat spelen met dit dashboard door dit te bewerken en andere widgets erop te zetten.