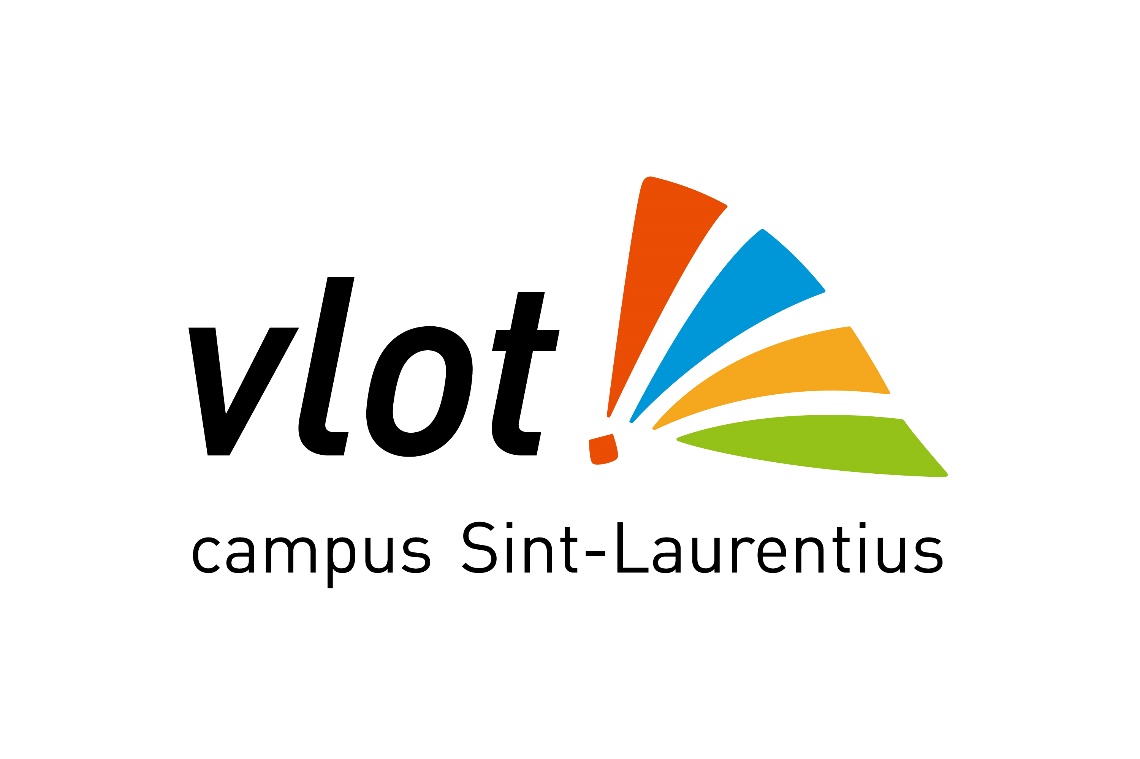
**Campus Sint-Laurentius**

VLOT!-vzw Prosper Thuysbaertlaan 1, 9160 Lokeren

Eindopdracht

Intelligent sensorarray voor binnenklimaatmetingen



Quinten Van Damme

Klas : 6 Industriële Informatie- en Communicatietechnologie (IT)

Schooljaar : 2023-2024

Mentor(en) : Wim Baert, Mieke Herreman, Sam Wesemael

Eindopdracht

**Intelligent sensorarray voor binnenklimaatmetingen**

# Inhoud

[Inhoud 2](#_Toc167210811)

[Lijst met afbeeldingen en tabellen 3](#_Toc167210812)

[Voorwoord 4](#_Toc167210813)

[Inleiding 5](#_Toc167210814)

[Realisatiedossier 5](#_Toc167210815)

[Hardware 5](#_Toc167210816)

[Raspberry Pi Pico W als microcontroller 5](#_Toc167210817)

[BME680 als sensor 6](#_Toc167210818)

[Sensor aansluitschema 8](#_Toc167210819)

[Behuizing sensoren 9](#_Toc167210820)

[Centrale computer 9](#_Toc167210821)

[Werkingsbeschrijving 10](#_Toc167210822)

[Quick start 10](#_Toc167210823)

[Gedetailleerde handleiding 13](#_Toc167210824)

[Software 15](#_Toc167210825)

[Raspberry Pi Pico W software 15](#_Toc167210826)

[Python (Computer) Software 20](#_Toc167210827)

[Besluit 26](#_Toc167210828)

[Bronnen 26](#_Toc167210829)

[„Getting Started with Raspberry Pi Pico W for IoT: Micropython and MQTT 27](#_Toc167210830)

[Bijlages 28](#_Toc167210831)

[Flowcharts 28](#_Toc167210832)

[Raspberry Pi Pico W 28](#_Toc167210833)

[Python (Computer) 29](#_Toc167210834)

[Software 34](#_Toc167210835)

[Databladen 64](#_Toc167210836)

# Lijst met afbeeldingen en tabellen

*Figuur 1: Raspberry Pi pico w*

*Figuur 2: BME 680 sensor*

Figuur 3: Bme 680 aangesloten op de pico w

Figuur 4: Bme 680 aansluitschema

*Figuur 5: Bodem*

*Figuur 6: Deksel*

*Figuur 7: Vaste computer als “server”*

*Figuur 8: Vaste computer “main.py”*

*Figuur 9: Sensor “main.py”*

*Figuur 10: Web interface*

*Figuur 11 : Flowchart functie “connect\_wifi”*

*Figuur 12 : Flowchart functie “connect\_mqtt”*

*Figuur 13 : Flowchart functie “publish\_meeting”*

*Figuur 14 : Flowchart klassen “Meeting” en “Sensor”*

*Figuur 15 : Flowchart functie “main”*

*Figuur 16 : Flowchart klassen “MQTTBROKER” en “MQTTCLIENT”*

*Figuur 17 : Flowchart methods “MQTTBROKER.run” en “MQTTCLIENT.run”*

*Figuur 18 : Flowchart klasse “Database”*

*Figuur 19: Flowchart methods “Database.create\_meetingen\_tabel” en “Database.create\_kalibreer\_table”*

*Figuur 20: Flowchart methods “Database.inser\_reading” en “Database.insert\_kalibratie”*

*Figuur 21 : Flowchart methods “Database.get\_readings” en “Database.get\_kalibratie”*

*Figuur 22: Flowchart methods “Database.clear\_meetingen” en “Database.clear\_kalibratie”*

*Figuur 23 : Flowchart klasse “WebServer”*

*Figuur 24 : Flowchart method “WebServer.do\_GET”*

*Figuur 25: Flowchart functie “run\_server”*

*Figuur 26: Flowchart functie “main”*

# Voorwoord

Ik heb voor de richting “Industriële informatie- en communicatietechnologie”  gekozen omdat ik sinds mijn 7 jaar al graag op de computer zat. Vanaf mijn 12 jaar wou ik altijd al IT doen. In de lagere school moest ik altijd de leerkrachten helpen als ze problemen hadden met de computer. Ik begon me toen meer een meer te interesseren in software.

Ik heb als onderwerp voor mijn eindproject gekozen “Intelligent sensorarray voor binnenklimaatmetingen” omdat ik altijd al veel data vanuit de omgeving wou verzamelen en op verschillende apparaten wil bekijken. Ik wou ook meer software doen dan hardware omdat software mij wat meer interesseert.

Ik wil graag meneer Baert bedanken voor het helpen kiezen van verschillende componenten. Mevrouw Herreman en Sam Wesemael voor het helpen bij moeilijk problemen.

# Inleiding

"Technologie verandert de wereld niet, mensen doen dat." Dit citaat van Steve Jobs benadrukt de rol van innovatie. In mijn geïntegreerde proef (GIP) ontwikkel ik een systeem dat de temperatuur op verschillende plaatsen in een ruimte meet en deze informatie naar een centrale computer stuurt. Dit systeem biedt de gebruiker inzicht in het binnenklimaat.

Het gebruik van meerdere sensoren geeft een nauwkeuriger beeld dan één enkele sensor. Bijvoorbeeld, een sensor bij een open raam in de winter zou een vertekend beeld geven. Door meerdere sensoren te gebruiken, kan ik een completer klimaatbeeld schetsen.

Mijn project past binnen de richting Informatietechnologie. Ik gebruik Raspberry Pi Pico W microcontroller vanwege zijn wifi- en bluetoothmogelijkheden en lage prijs. De data verstuur ik via het MQTT-protocol over een wifi-netwerk. De sensoren bevestig ik aan de kabelgoten in het klaslokaal, waar al verlengkabels aanwezig zijn.

Voor de verwerking en presentatie van de data gebruik ik Python. Met de vele beschikbare libraries kan ik de data binnenlezen via MQTT, opslaan in een database en visueel weergeven op een webpagina. Mijn desktop pc fungeert als centrale computer vanwege de benodigde rekenkracht en wifi-capaciteiten.

Naast temperatuur meet ik ook druk, luchtvochtigheid en gasniveaus met de BME680 sensor, die al deze waarden kan lezen. Dit project toont mijn technische vaardigheden en biedt een praktische oplossing voor binnenklimaatmonitoring.

# Realisatiedossier

## Hardware

### Raspberry Pi Pico W als microcontroller

Ik heb de raspberry pi pico w gekozen als microcontroller omdat deze zeer goedkoop is, een vorm van python kan draaien genaamd micropython en deze ook wifi en bluetooth mogelijkheden heeft. Je kunt de microcontroller als gebruiker zeer makkelijk programmeren met thonny.

|  |
| --- |
| RASPBERRY-PI RASPBERRY PI PICO W SBC, Raspberry Pi Pico W, RP2040, ARM  Cortex-M0+, 264kB RAM, 2MB Flash, Wifi, Micro-USB |
| *Figuur 1: Raspberry Pi pico w* |

### BME680 als sensor

Afbeelding met tekst, elektronica, flashgeheugen

Automatisch gegenereerde beschrijvingDe BME680 kan dankzij zijn i2c communicatie gemakkelijk zonder veel draden aangesloten worden op de microcontroller. Ook kan deze 4 meetwaarden lezen: temperatuur, luchtdruk, luchtvochtigheid en gas. Deze sensor kan zeer goedkoop aangeschaft worden en met een library ook gemakkelijk worden geprogrammeerd.

*Figuur 2: BME 680 sensor*

### Sensor aansluitschema

Omdat we i2c gebruiken hebben we maar 2 draden voor communicatie nodig en 2 voedingsdraden. SCL (groen) is het kloksignaal waarbij de sensor en de microcontroller op een vastgesteld tempo kunnen communiceren. SDA (blauw) is de draad waar de data wordt verzonden en ontvangen. In dit geval worden de meetwaarden die de sensor meet verzonden naar de microcontroller.

|  |
| --- |
| Afbeelding met Elektronisch onderdeel, Stroomkringonderdeel, Passief stroomkringonderdeel, Elektronische engineering  Automatisch gegenereerde beschrijving |
| Figuur 3: Bme 680 aangesloten op de pico w |
| Afbeelding met elektronica, Elektronisch onderdeel, stroomkring, Stroomkringonderdeel  Automatisch gegenereerde beschrijving |
| Figuur 4: Bme 680 aansluitschema |

### Behuizing sensoren

De behuizing bestaat uit 2 onderdelen, de bodem en het deksel. Het deksel bevat ook een tekst die zegt welke sensor het is. Ik hen deze bestanden met behulp van Autodesk inventor gemaakt en met de 3d printer geprint.

|  |  |
| --- | --- |
| Afbeelding met schermopname, Rechthoek, ontwerp  Automatisch gegenereerde beschrijving |  |
| *Figuur 5: Bodem* | *Figuur 6: Deksel* |

### Centrale computer

Voor de database en de mqtt server te draaien heb je een computer nodig, hiervoor heb ik gebruik gemaakt van een school computer. Niet onbelangerijk deze computer heeft een static IP-adres wat wil zeggen dat het IP-adres van deze computer niet veranderd, dit zorgt ervoor dat ik niet constant de code van de sensoren moet wij wijzigen omdat het IP-adres is veranderd.

|  |
| --- |
|  |
| *Figuur 7: Vaste computer als “server”* |

## Werkingsbeschrijving

### Quick start

Dit is de korte handleiding van jouw werk. Je hebt software geschreven die op hardware draait.

Hier leg je aan de gebruiker uit hoe hij met de hard- en software moet omgaan. Hoe wordt de toepassing opgestart? Welke voeding moet er worden voorzien? Je moet hier uitgaan van een *technisch onderlegd* gebruiker. Je voorziet een stappenplan in instructietaal. De gebruiker mag na het lezen van dit deel gaan twijfels meer hebben. Ook veiligheid wordt hier besproken.

Op de computer open je thonny daarna steek je je sensor in en selecteer je recht onder de pico om mee te verbinden. Daarna zal je een paar variabelen in main.py moeten veranderen. Zoals de wifi naam, wachtwoord, het server adres, de sensor naam en de plaats waar de pico staat. Daarna druk je linksboven op bestand -> opslaan als -> opslaan op pico w. Je geeft de naam p.py. Het programma staat nu op de pico en zal de software draaien wanneer je hem in een voeding stopt (usb voeding).

Het server adres kan je terugvinden door op je computer ipconfig in command prompt te openen. Of door het programma op de computer te runnen.

|  |
| --- |
|  |
| *Figuur 8: Vaste computer “main.py”* |

Nu dat we de sensor geprogrammeerd hebben moeten we de software draaien op de computer om de sensor met de computer te verbinden en die vervolgens die data openbaar maakt op een website. Installeer daarvoor eerst “mosquitto mqtt” en open daarna main.py en druk bovenaan op run.

|  |
| --- |
|  |
| *Figuur 9: Sensor “main.py”* |

Ga nu naar 172.0.0.1:8000 en dan zal je deze pagina te zien krijgen:

|  |
| --- |
| Afbeelding met tekst, schermopname, nummer, ontvangst  Automatisch gegenereerde beschrijving |
| *Figuur 10: Web interface* |

Daarna kan je de sensoren aansluiten aan een voeding, kan telfoon oplader zijn en wanneer de sensor verbonden is zal het lichte branden van de sensor en zal je op de web interface de grafiek zien veranderen.

### Gedetailleerde handleiding

Dit is de uitgebreide handleiding van jouw werk. Je hebt software geschreven die op hardware draait. Hier leg je aan de gebruiker uit hoe hij met de hard- en software moet omgaan. Je moet hier uitgaan van een *niet onderlegd* gebruiker. Je voorziet een stappenplan in instructietaal. De gebruiker mag na het lezen van dit deel gaan twijfels meer hebben. Ook veiligheid wordt hier besproken.

#### Voorbereiding

##### Benodigdheden

* Raspberry Pi Pico W
* Sensor
* USB-voeding (bijvoorbeeld een telefoonoplader)
* Computer met Thonny IDE geïnstalleerd
* Werkende internetverbinding
* Serveradres (te achterhalen via IP-configuratie)

#### Stappenplan

##### 1. Hardware Setup

1. BME680 aansluiten op de microcontroller:
   * Zie figuur 4 voor het aansluitschema.

##### 2. Software Setup op de Computer

1. Thonny IDE Openen:
   * Start Thonny IDE op je computer.
2. Sensor Aansluiten en Verbinden:
   * Steek de sensor in de juiste poort van de Raspberry Pi Pico W.
   * Selecteer rechtsonder in Thonny de Pico W als het apparaat om mee te verbinden.
3. Variabelen in main.py Aanpassen:
   * Open main.py in Thonny.
   * Pas de volgende variabelen aan naar jouw specifieke configuratie:
     + wifi\_naam: De naam van je Wi-Fi-netwerk.
     + wachtwoord: Het wachtwoord van je Wi-Fi-netwerk.
     + server\_adres: Het adres van de server waar de data naartoe gestuurd wordt.
     + sensor\_naam: De naam die je aan de sensor wilt geven.
     + plaats\_sensor: De locatie waar de sensor zich bevindt.
4. Bestand Opslaan op Pico W:
   * Klik linksboven op Bestand.
   * Selecteer Opslaan als en kies Opslaan op Pico W.
   * Geef het bestand de naam p.py.

##### 3. Serveradres Achterhalen

1. IP-adres Verkrijgen:
   * Open Command Prompt op je computer.
   * Typ ipconfig en druk op Enter.
   * Noteer het IPv4-adres. Dit is het serveradres dat je moet invoeren in main.py.

#### Sensor en Software in Gebruik Nemen

##### 4. Sensor Programmeren

1. Programma Opslaan en Runnen:
   * Sla het programma op de Pico W op zoals eerder beschreven.
   * Wanneer je de Pico W in de voeding steekt, zal deze automatisch de software draaien.

##### 5. Software op de Computer

1. Mosquitto MQTT Installeren:
   * Installeer Mosquitto MQTT via de officiële website.
2. main.py Runnen:
   * Open main.py op je computer.
   * Klik bovenaan op Run om het programma te starten.

##### 6. Web Interface Toegang

1. Web Interface Openen:
   * Ga naar 172.0.0.1:8000 in je webbrowser.
   * Je zou nu de web interface moeten zien waar de sensor data real-time wordt weergegeven.

## Software

Je legt hier stap voor stap uit hoe je voor de software vanuit de probleemstelling naar een oplossing bent gegaan. Dit hoofdstuk kan onderdelen bevatten zoals…

### Raspberry Pi Pico W software

#### Wifi.py

Om de gegevens van de microcontroller naar de server te brengen maken we gebruik van wifi. Ik kon niet het schoolwifi gebruiken dus moest ik mijn eigen modem gebruiken. Verbinding maken met wifi was heel simpel en het houd in dat je de wifi module van de microcontroller inschakeld en verbinding maakt met een bepaald wifi netwerk.

* Functie : connect\_wifi(ssid,password)
* Doel : Connectie maken met het wifi netwerk.
* Parameters : ssid (str), password (str)
* Returnwaarde : void
* Te installeren headers: network, time
* Flowchart:

|  |
| --- |
| Afbeelding met tekst, schermopname, diagram, Lettertype  Automatisch gegenereerde beschrijving |
| *Figuur 11 : Flowchart functie “connect\_wifi”* |

#### Mqtt.py

Om de gegevens te verzenden en te ontvagen moeten beide apparaten hetzelfde protocol spreken, in dit geval maken we gebruik van mqtt die het heel makkelijk maakt om data te verzenden naar een server of andere apparaten. Ik heb ervoor gekozen om lokaal een mqtt server te maken omdat ik niet kon verbinden met mqtt servers in de cloud.

* Functie : connect\_mqtt(client\_id,server,port)
* Doel : Connectie maken met de mqtt server.
* Parameters : client\_id (str), server (str), port (int)
* Returnwaarde : client (mqtt.Client)
* Te installeren headers: umqtt.simple
* Flowchart:

|  |
| --- |
| Afbeelding met tekst, schermopname, diagram, ontwerp  Automatisch gegenereerde beschrijving |
| *Figuur 12 : Flowchart functie “connect\_mqtt”* |

* Functie : publish\_meeting(client,meeting,topic)
* Doel : Vezend een meeting naar de mqtt server.
* Parameters : client (mqtt.Client), meeting (Meeting), topic (str)
* Returnwaarde : void
* Te installeren headers: umqtt.simple
* Flowchart:

|  |
| --- |
| Afbeelding met tekst, schermopname, Lettertype, symbool  Automatisch gegenereerde beschrijving |
| *Figuur 13 : Flowchart functie “publish\_meeting”* |

#### Bme680.py

Met de bme680 sensor had ik weinig problemen omdat er al een library voor geschreven is wat het heel eenvoudig maakt om met de sensor te werken. Wel heb ik ondervonden dat deze sensor makkelijk kapotgaat als je te lang soldeert.

* Klassen : Meeting, Sensor
* Doel : De Meeting klasse bevat de data van een meeting en de Sensor klasse bevat de data van een sensor.
* Parameters : Meeting: temperatuur (float), luchtvochtigheid (float), luchtdruk (float), gas (float), x (float), y (float), tijd (Time), sensor\_id (str)
* Parameters : Sensor: x (float), y (float), sensor\_id (str), i2c (i2c), bme (BME680\_I2C), uart0 (UART)
* Returnwaarde : Meeting, Sensor
* Te installeren headers: bme680lib, machine, utime, time, json
* Flowchart:

|  |
| --- |
| Afbeelding met tekst, schermopname, Lettertype  Automatisch gegenereerde beschrijving |
| *Figuur 14 : Flowchart klassen “Meeting” en “Sensor”* |

#### Main.py

In de main functie maken we gebruik van een aantal constante variabelen voor de wifi en mqtt. De main functie zelf maakt eerst verbinding met de wifi daarna met de mqtt server en zal daarna elke seconde de bme680 sensor inlezen en de meeting verzenden naar de mqtt server.

* Functie : main()
* Doel : De main functie van het programma.
* Parameters : void
* Returnwaarde : void
* Te installeren headers: time, wifi, mqtt, bme680
* Flowchart:

|  |
| --- |
| Afbeelding met tekst, schermopname, Lettertype, ontwerp  Automatisch gegenereerde beschrijving |
| *Figuur 15 : Flowchart functie “main”* |

### Python (Computer) Software

#### Mqtt.py

* Klassen : MQTTBROKER, MQTTCLIENT
* Doel : De MQTTBROKER klasse bevat de data van een mqtt broker en de MQTTCLIENT klasse bevat de data van een mqtt client.
* Parameters : void
* Returnwaarde : MQTTBROKER, MQTTCLIENT
* Te installeren headers: paho.mqtt, time, subprocess, os, json, constants, database
* Flowchart:

|  |
| --- |
|  |
| *Figuur 16 : Flowchart klassen “MQTTBROKER” en “MQTTCLIENT”* |

|  |
| --- |
|  |
| *Figuur 17 : Flowchart methods “MQTTBROKER.run” en “MQTTCLIENT.run”* |

#### Database.py

* Klassen : Database
* Doel : De Database klasse bevat de functie om om te communiceren met de database.
* Parameters : db\_file (str)
* Returnwaarde : Database
* Te installeren headers: time, sqlite3, json
* Flowchart:

|  |
| --- |
|  |
| *Figuur 18 : Flowchart klasse “Database”* |

|  |
| --- |
|  |
| *Figuur 19 : Flowchart methods “Database.create\_meetingen\_tabel” en “Database.create\_kalibreer\_table”* |
|  |
| *Figuur 20: Flowchart methods “Database.inser\_reading” en “Database.insert\_kalibratie”* |

|  |
| --- |
|  |
| *Figuur 21 : Flowchart methods “Database.get\_readings” en “Database.get\_kalibratie”* |
|  |
|  |
| *Figuur 22: Flowchart methods “Database.clear\_meetingen” en “Database.clear\_kalibratie”* |

#### Web.py

* Klassen : WebServer
* Doel : De WebServer klasse bevat functies om een webserver te maken.
* Parameters : BaseHTTPRequestHandler
* Returnwaarde : WebServer
* Te installeren headers: http.server, time, json, constants, database
* Flowchart:

|  |
| --- |
|  |
| *Figuur 23 : Flowchart klasse “WebServer”* |

|  |
| --- |
|  |
| *Figuur 24 : Flowchart method “WebServer.do\_GET”* |

* Functie : run\_server()
* Doel : Start de webserver.
* Parameters : void
* Returnwaarde : void
* Te installeren headers:
* Flowchart:

|  |
| --- |
|  |
| *Figuur 25: Flowchart functie “run\_server”* |

#### Main.py

* Functie : main()
* Doel : De main functie van het programma.
* Parameters : void
* Returnwaarde : void
* Te installeren headers: time, threading, web, mqtt
* Flowchart:

|  |
| --- |
|  |
| *Figuur 26: Flowchart functie “main”* |

# Besluit

Mijn eindproject is bijna compleet, maar er zijn nog enkele werkpunten die moeten worden aangepakt:

Werkpunten:

1. Kalibreren van de sensoren: De sensoren moeten nauwkeurig worden gekalibreerd voor betrouwbare metingen.
2. Verbinden met de sensoren: De betrouwbaarheid van de verbinding met de sensoren moet worden verbeterd.
3. Gebruik van een eigen router: Er moet een eigen router worden gebruikt om te verbinden.
4. MQTT-configuratie: De MQTT-instellingen moeten worden herzien. Momenteel stuurt het systeem alle data naar a103/sensors in plaats van naar specifieke sensortopics. Dit moet worden verbeterd.
5. Temperatuurmeting: De nauwkeurigheid van de temperatuurmetingen moet worden verbeterd. De huidige sensor meet, vanwege de behuizing, ongeveer 7°C hoger dan de werkelijke temperatuur. Dit probleem moet worden opgelost.
6. Tijdens het maken van mijn eindproject zijn er 2 bme680 sensors kapot gegaan waarschijnlijk door te lang te solderen.
7. De webinterface is niet aangepast voor smartphones.

Positieve punten:

1. Website: Het maken van de website verliep goed.
2. Database code: Het schrijven van de database code ging ook goed.
3. Behuizing: Het ontwerpen en printen van de behuizing verliep succesvol.

Wat ging er goed:

1. De waarden van elke sensor worden mooi op hun eigen grafiek weergeven.
2. Het maken van de website ging ook vrij vlot en javascript is voor de hand liggend.

Wat ging er niet goed:

1. Als uitbreiding had ik voorgesteld om een i2c display aan te sluiten op de pico w maar na weken proberen is het mij niet gelukt om te communiceren met de display.
2. Het aan de praat krijgen van een mqtt server is niet zo makkelijk en communiceren met de mqtt server heeft ook wat tijd in beslag genomen.
3. Communiceren met de database is moeilijk omdat sqllite gebruik maak van zijn eigen taal genaamd query's.

# Bronnen

Hier vermeld je alle bronnen die je bij de uitwerking van je opdracht hebt gebruikt. Dat kunnen website’s, boeken, catalogi,…zijn.

Een juiste bronvermelding is enorm belangrijk. Zo kun je gemakkelijk extra informatie opzoeken of controleren of je de bron op een correcte manier gebruikt hebt. Het toont bovendien aan dat je onderzoek gedaan hebt om je gip wetenschappelijk te onderbouwen. Telkens als je een bron gebruikt, vermeld je dat in je giptekst door vierkante haakjes te plaatsen met daarin een volgnummer. Alle bronvermeldingen verzamel je op één lijst met als titel *Bronnen*.

|  |  |
| --- | --- |
| [1] | „Getting Started with Raspberry Pi Pico W for IoT: Micropython and MQTT ” Available: https://www.hivemq.com/article/iot-reading-sensor-data-raspberry-pi-pico-w-micropython-mqtt-node-red/ |
| [2] | „Raspberry Pi Pico BME680 Web Server with MicroPython” Available: https://microcontrollerslab.com/raspberry-pi-pico-bme680-web-server-micropython/ | |
| [3] |

# Bijlages

## Flowcharts

### Raspberry Pi Pico W

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, diagram, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, diagram, ontwerp

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, Lettertype, symbool

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype, ontwerp

Automatisch gegenereerde beschrijving

### Python (Computer)

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, diagram, Lettertype

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, diagram, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype, diagram

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, Lettertype, diagram

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype, diagram

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, diagram

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype, ontwerp

Automatisch gegenereerde beschrijving

## Software

# (sensor) main.py

import time

from wifi import connect\_wifi

from mqtt import connect\_mqtt, publish\_meeting

from bme680 import Sensor

# wifi

wifi\_ssid = "bokkie"

wifi\_password = "papabokkie"

# mqtt

mqtt\_client\_id = "A103S2"

mqtt\_server = "10.16.0.12"

mqtt\_port = 1884

mqtt\_topic = "A103/picos"

def main():

connect\_wifi(wifi\_ssid,wifi\_password)

client = connect\_mqtt(mqtt\_client\_id,mqtt\_server,mqtt\_port)

sensor = Sensor(0,0)

while(True):

meeting = sensor.get\_sensor\_data()

publish\_meeting(client,meeting,mqtt\_topic)

print("test")

time.sleep(1)

main()

# (sensor) wifi.py

import network

import time

def connect\_wifi(ssid,password):

wlan = network.WLAN(network.STA\_IF)

wlan.active(True)

wlan.connect(ssid,password)

while(not wlan.isconnected()):

print(f"trying to connect to {ssid} wifi")

time.sleep(1)

# (sensor) mqtt.py

import time

from umqtt.simple import MQTTClient

def connect(client\_id,server,port):

client = MQTTClient(client\_id, server, port=port, keepalive=3600)

client.connect()

print(f'Connected to mqtt://{server}:{port} MQTT Broker as {client\_id}')

return client

def reconnect():

print('Failed to connect to the MQTT Broker. Reconnecting...')

time.sleep(5)

def connect\_mqtt(client\_id,server,port):

connected = False

while (connected is False):

try:

client = connect(client\_id,server,port)

connected = True

return client

except OSError as e:

reconnect()

def publish\_meeting(client,meeting,topic):

client.publish(topic,meeting)

# (sensor) bme680.py

from bme680lib import \*

import machine

from machine import Pin, I2C

import utime

import time

import json

class Meeting():

def \_\_init\_\_(self,bme680,x,y,sensor\_id):

self.temperatuur = str(round(bme680.temperature, 2))

self.luchtvochtigheid = str(round(bme680.humidity, 2))

self.luchtdruk = str(round(bme680.pressure, 2))

self.gas = str(round(bme680.gas/1000, 2))

self.x = x

self.y = y

self.tijd = utime.time()

self.sensor\_id = sensor\_id

def toJson(self):

meeting = {

"temperatuur": self.temperatuur,

"luchtvochtigheid": self.luchtvochtigheid,

"luchtdruk": self.luchtdruk,

"gas": self.gas,

"x": self.x,

"y": self.y,

"tijd": self.tijd,

"sensor\_id": self.sensor\_id

}

return json.dumps(meeting)

class Sensor():

def \_\_init\_\_(self,x,y,sensor\_id):

self.x = x

self.y = y

self.sensor\_id = sensor\_id

self.i2c=I2C(1,sda=Pin(2), scl=Pin(3), freq=400000) #initializing the I2C method

self.bme = BME680\_I2C(i2c=self.i2c)

uart0 = machine.UART(0, baudrate=115200)

def get\_sensor\_data(self):

return Meeting(self.bme,self.x,self.y,self.sensor\_id).toJson()

# (server) main.py

import time

import threading

from web import run\_server

from mqtt import MQTTBROKER, MQTTCLIENT

if \_\_name\_\_ == "\_\_main\_\_":

webServerThread = threading.Thread(target=run\_server)

mqttServerThread = threading.Thread(target=MQTTBROKER.run)

webServerThread.start()

mqttServerThread.start()

time.sleep(1)

MQTTCLIENT.run()

# (server) mqtt.py

from paho.mqtt import client as mqtt

import time

import subprocess

import os

import json

from constants import \*

from database import DataBase

database = DataBase()

class MQTTBROKER():

def run():

os.system("taskkill /f /im mosquitto.exe") # sluit eerst alle mqtt servers af

# maak een mqtt server aan

print(f"[SERVER] MQTT is running on mqtt://{IP}:{MQTT\_PORT}/")

subprocess.run(["C:\Program Files\mosquitto\mosquitto.exe", "-c", "C:\Program Files\mosquitto\mosquitto.conf", "-v", "-p", f"{MQTT\_PORT}"], shell=True)

class MQTTCLIENT():

def on\_message(client, userdata, message):

decoded\_message=str(message.payload.decode("utf-8"))

msg=json.loads(decoded\_message)

database.insert\_reading(msg)

def run():

client = mqtt.Client(MQTT\_CLIENT\_NAME) #create new instance

client.connect(IP, port=MQTT\_PORT, bind\_address="") #connect to broker

client.subscribe(MQTT\_TOPIC)

client.on\_message=MQTTCLIENT.on\_message

client.loop\_forever()

# (server) web.py

from http.server import BaseHTTPRequestHandler, HTTPServer

import time

import re

from constants import \*

from database import DataBase

database = DataBase()

class WebServer(BaseHTTPRequestHandler):

def do\_GET(self):

if re.search('/api/waarde/\*',self.path):

sensor\_id = self.path.split('/')[-1]

self.do\_meetingen(sensor\_id)

elif re.search('/api/kalibratie/\*',self.path):

sensor\_id = self.path.split('/')[-1]

self.get\_kalibratie(sensor\_id)

elif re.search('/api/clear',self.path):

self.clear()

else:

self.send\_response(200)

self.send\_header('Content-type', 'text/html')

self.end\_headers()

with open('index.html', 'rb') as file:

self.wfile.write(file.read())

def do\_meetingen(self,sensor\_id):

meetingen = database.get\_readings(20,sensor\_id)

self.send\_response(200)

self.send\_header('Content-type', 'application/json')

self.end\_headers()

self.wfile.write(meetingen.encode())

def get\_kalibratie(self,sensor\_id):

kalibratie = database.get\_kalibratie(sensor\_id)

self.send\_response(200)

self.send\_header('Content-type', 'application/json')

self.end\_headers()

self.wfile.write(kalibratie.encode())

def clear(self):

database.clear\_meetingen()

meetingen = database.get\_readings(20,sensor\_id)

self.send\_response(200)

def run\_server():

server\_address = ('', WEBSITE\_PORT)

httpd = HTTPServer(server\_address, WebServer)

print(f"[SERVER] Server is running on http://{IP}:{WEBSITE\_PORT}/")

try:

httpd.serve\_forever()

except KeyboardInterrupt:

pass

httpd.server\_close()

# (server) database.py

import time

import sqlite3

import json

class DataBase:

def \_\_init\_\_(self, db\_file='metingen.db'):

self.db\_file = db\_file

self.conn = sqlite3.connect(db\_file, check\_same\_thread=False)

self.create\_meetingen\_table()

self.create\_kalibreer\_table()

def create\_meetingen\_table(self):

query = '''

CREATE TABLE IF NOT EXISTS sensor\_data (

id INTEGER PRIMARY KEY AUTOINCREMENT,

sensor\_id TEXT,

temperatuur REAL,

lucht\_vochtigheid REAL,

lucht\_druk REAL,

gas REAL,

x REAL,

y REAL,

tijd INTEGER

)

'''

with self.conn:

self.conn.execute(query)

def create\_kalibreer\_table(self):

query = '''

CREATE TABLE IF NOT EXISTS kalibratie\_data (

id INTEGER PRIMARY KEY AUTOINCREMENT,

sensor\_id TEXT,

temperatuur REAL

lucht\_vochtigheid REAL,

lucht\_druk REAL,

gas REAL

)

'''

with self.conn:

self.conn.execute(query)

def insert\_reading(self, json):

gas = json['gas']

temperatuur = json['temperatuur']

sensor\_id = json['sensor\_id']

lucht\_vochtigheid = json['luchtvochtigheid']

lucht\_druk = json['luchtdruk']

x = json['x']

y = json['y']

tijd = json['tijd']

query = '''

INSERT INTO sensor\_data (sensor\_id, temperatuur, lucht\_vochtigheid, lucht\_druk, gas, x, y, tijd)

VALUES (?, ?, ?, ?, ?, ?, ?, ?)

'''

with self.conn:

self.conn.execute(query, (sensor\_id, temperatuur, lucht\_vochtigheid, lucht\_druk, gas, x, y, tijd))

def insert\_kalibratie(self, json):

sensor\_id = json['sensor\_id']

temperatuur = json['temperatuur']

sensor\_id = json['sensor\_id']

lucht\_vochtigheid = json['luchtvochtigheid']

lucht\_druk = json['luchtdruk']

# Controleren of een item met hetzelfde sensor\_id al in de database staat

existing\_query = '''

SELECT COUNT(\*) FROM kalibratie\_data WHERE sensor\_id = ?

'''

with self.conn:

existing\_count = self.conn.execute(existing\_query, (sensor\_id,)).fetchone()[0]

if existing\_count > 0:

# Overschrijven van de waarde als het item al bestaat

update\_query = '''

UPDATE kalibratie\_data SET waarde = ? WHERE sensor\_id = ?

'''

self.conn.execute(update\_query, (waarde, sensor\_id))

else:

# Toevoegen van een nieuw item als het item niet bestaat

insert\_query = '''

INSERT INTO kalibratie\_data (sensor\_id, waarde) VALUES (?, ?)

'''

self.conn.execute(insert\_query, (sensor\_id, waarde))

def get\_readings(self, aantal,sensor\_id):

readings = []

query = f'SELECT \* FROM sensor\_data WHERE sensor\_id == ? ORDER BY id DESC LIMIT ?'

with self.conn:

cursor = self.conn.execute(query, (sensor\_id, aantal))

r = cursor.fetchall()

for reading in r:

x = {

"sensor\_id": reading[1],

"temperatuur": reading[2],

"lucht\_vochtigheid": reading[3],

"lucht\_druk": reading[4],

"gas": reading[5],

"x": reading[6],

"y": reading[7],

"tijd": reading[8]

}

readings.append(x)

return json.dumps(readings)

def get\_kalibratie(self, sensor\_id):

query = '''

SELECT \* FROM kalibratie\_data WHERE sensor\_id = ?

'''

with self.conn:

result = self.conn.execute(query, (sensor\_id)).fetchone()

if result:

# Gegevens gevonden voor de sensor\_id

x = {

"sensor\_id": reading[1],

"temperatuur": reading[2],

"lucht\_vochtigheid": reading[3],

"lucht\_druk": reading[4],

"gas": reading[5],

}

return json.dumps(x)

else:

# Geen gegevens gevonden voor de sensor\_id

x = {

"sensor\_id": sensor\_id,

"temperatuur": 0,

"lucht\_vochtigheid": 0,

"lucht\_druk": 0,

"gas": 0,

}

return json.dumps(x)

def clear\_meetingen(self):

query = '''delete from sensor\_data'''

with self.conn:

self.conn.execute(query)

def clear\_kalibratie(self):

query = '''delete from kalibratie\_data'''

with self.conn:

self.conn.execute(query)

def close\_connection(self):

self.conn.close()

# (server) constants.py

import socket

IP = socket.gethostbyname\_ex(socket.gethostname())[-1][1]

WEBSITE\_PORT = 8000

MQTT\_PORT = 1884

MQTT\_CLIENT\_NAME = "SERVER"

MQTT\_TOPIC = "A103/picos"

# (server) index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="icon" type="image/png" sizes="192x192" href="https://vlot.be/wp-content/themes/vlot/images/favicon/android-icon-192x192.png">

<link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />

<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.js"></script>

<title>Eindproject</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

header {

background-color: #fff;

border-bottom: 1px solid #efefef;

padding: 10px;

text-align: left;

}

header h1 {

margin: 0;

}

header img {

max-width: 7vw;

height: auto;

margin-top: 1vh;

margin-bottom: 1vh;

margin-left: 7vw;

}

header button {

background: none;

border: none;

cursor: pointer;

padding: 5px;

border-radius: 0.5rem;

float: right;

margin-top: 1.4rem;

}

header button:hover {

background: #d5d5d5;

}

.content {

z-index: 1;

margin-left: 1vw;

margin-right: 1vw;

}

.meetGrid {

display:grid;

grid-template-columns: 700px 700px;

grid-row: auto auto;

grid-column-gap: 20px;

grid-row-gap: 20px;

}

.tabs {

display: flex;

flex-wrap: wrap;

}

.tabs > section {

order: 999;

width: 100%;

display: none;

}

.tabs > input {

/\* display: none; Don’t use display:none. Bad for accessibility \*/

/\* Step 6: Hide the radio inputs \*/

opacity: 0;

/\* Step 7: Make sure the radio inputs don’t take up space in layout \*/

position: absolute;

}

/\* Step 8: Select the label right next to the selected input \*/

.tabs > input[type=radio]:checked + label {

/\* Step 9: Highlight the selected label \*/

background: yellow;

}

/\* Step 10: Select the section right next to the label which is next to the selected input \*/

.tabs > input[type=radio]:checked + label + section {

/\* Step 11: Unset the 'display:none' we did in step 5 \*/

display: unset;

}

/\* Done. Make sure to disable the debug code at the beginning of CSS. And now it is time to make the tabs look good \*/

/\* Make the tabs look good \*/

/\* Final Step: Make the tabs pretty with padding and colors \*/

.tabs > label {

padding: .5em 1em;

background: #d5d5d5;

border-radius: 0.8rem;

margin-bottom: 0.5rem;

margin-right: 0.5rem;

border-right: 1px solid #efefef;

cursor: pointer;

}

.tabs > label:last-of-type {

border-right: none;

}

.tabs > input[type=radio]:checked + label {

background: #efefef;

}

.tabs section {

border-radius: 0.8rem;

border: 1px #efefef solid;

border-top: 5px #efefef solid;

padding: 1em;

}

#settingsBar {

width: 100vw;

height: 50px;

left: 0;

background-color: #fff;

border-bottom: 1px solid #efefef;

padding: 10px;

}

#settingsBar div {

display: flex;

margin-right: 5px;

width: fit-content;

height: 100%;

gap: 10px;

}

#settingsBar button {

border: none;

background: #d5d5d5;

color: #000;

cursor: pointer;

padding: 5px 10px;

border-radius: 0.5rem;

font-size: 16px;

}

#settingsBar button:hover {

background: #efefef;

}

#settingsBar button:first-of-type {

margin-left: 20px;

}

#prompt-wrapper

{

display: flex;

z-index: 4;

position: absolute;

top: 0;

left: 0;

min-width: 100vw;

min-height: 100vh;

background: #d5d5d50a;

backdrop-filter: blur(5px);

align-content: center;

justify-content: center;

padding: 10px;

}

#prompt {

display: flex;

margin-top: auto;

margin-bottom: auto;

width: 20vw;

height: 30vh;

background: #d5d5d5;

border-radius: 0.5rem;

align-content: center;

justify-content: center;

flex-direction: column;

padding: 15px;

}

#prompt h1 {

text-align: center;

}

#prompt button {

width: fit-content;

display: flex;

margin-top: auto;

margin-left: auto;

padding: 5px 10px;

}

</style>

</head>

<body>

<header>

<img src="https://vlot.be/wp-content/themes/vlot/images/svg/vlot\_logo.svg" alt="Logo">

<button onclick="toggleSettings()"><span class="material-symbols-outlined" style="font-size: 28px;">settings</span></button>

</header>

<div id="settingsBar" style="display:none;">

<div>

<button id="clearButton">Clear Database</button>

<button id="kalibreer">Kalibreer</button>

</div>

</div>

<div class="content">

<h1>Dashboard</h1>

<div class='tabs'>

<input type="radio" name="tab" id="S1" role="tab" checked>

<label for="S1" id="S1-label">Sensor1</label>

<section aria-labelledby="tab1-label">

<div class="meetGrid">

<canvas id="tempS1" style="width:100%;max-width:700px"></canvas>

<canvas id="drukS1" style="width:100%;max-width:700px"></canvas>

<canvas id="vochtS1" style="width:100%;max-width:700px"></canvas>

<canvas id="gasS1" style="width:100%;max-width:700px"></canvas>

</div>

</section>

<input type="radio" name="tab" id="S2" role="tab">

<label for="S2" id="S2-label">Sensor 2</label>

<section aria-labelledby="tab2-label">

<div class="meetGrid">

<canvas id="tempS2" style="width:100%;max-width:700px"></canvas>

<canvas id="drukS2" style="width:100%;max-width:700px"></canvas>

<canvas id="vochtS2" style="width:100%;max-width:700px"></canvas>

<canvas id="gasS2" style="width:100%;max-width:700px"></canvas>

</div>

</section>

<input type="radio" name="tab" id="S3" role="tab">

<label for="S3" id="S3-label">Sensor 3</label>

<section aria-labelledby="tab2-label">

<div class="meetGrid">

<canvas id="tempS3" style="width:100%;max-width:700px"></canvas>

<canvas id="drukS3" style="width:100%;max-width:700px"></canvas>

<canvas id="vochtS3" style="width:100%;max-width:700px"></canvas>

<canvas id="gasS3" style="width:100%;max-width:700px"></canvas>

</div>

</section>

</div>

</div>

<div id="prompt-wrapper" style="display:none;">

<div id="prompt">

</div>

</div>

</body>

<script>

function convertTimeStringToDate(timestring)

{

// Create a new JavaScript Date object based on the timestamp

// multiplied by 1000 so that the argument is in milliseconds, not seconds

var date = new Date(timestring \* 1000);

// Hours part from the timestamp

var hours = date.getHours();

// Minutes part from the timestamp

var minutes = "0" + date.getMinutes();

// Seconds part from the timestamp

var seconds = "0" + date.getSeconds();

var formattedTime = hours + ':' + minutes.substr(-2) + ':' + seconds.substr(-2);

return formattedTime;

}

async function getWaarde(sensor\_id) {

let response = await fetch('/api/waarde/' + sensor\_id);

let json = await response.json();

return json;

}

async function updateChart(id,charts)

{

let tempWaarden = [];

let drukWaarden = [];

let vochtWaarden = [];

let gasWaarden = [];

let tijdWaarden = [];

const waarde = await getWaarde('A103S' + id);

for (let i = waarde.length - 1; i > 0; i--)

{

tempWaarden.push(waarde[i]['temperatuur'])

drukWaarden.push(waarde[i]['lucht\_druk'])

vochtWaarden.push(waarde[i]['lucht\_vochtigheid'])

gasWaarden.push(waarde[i]['gas'])

tijdWaarden.push(convertTimeStringToDate(waarde[i]['tijd']))

}

for (let i = 0; i < charts.length; i++)

{

charts[i].data.labels = tijdWaarden;

}

charts[0].data.datasets[0].data = tempWaarden;

charts[1].data.datasets[0].data = drukWaarden;

charts[2].data.datasets[0].data = vochtWaarden;

charts[3].data.datasets[0].data = gasWaarden;

for (let i = 0; i < charts.length; i++)

{

charts[i].update();

}

}

function initChart(id)

{

let charts = [];

let temp = new Chart("tempS" + id, {

type: "line",

data: {

labels: [],

datasets: [{

fill: false,

lineTension: 0,

backgroundColor: "rgba(255,51,1,1.0)",

borderColor: "rgba(255,0,0,0)",

data: []

}]

},

options: {

legend: {display: false},

title: {

display: true,

text: "Sensor " + id + " Temperatuur"

},

animation: {

duration: 0

},

scales: {

yAxes: [{

ticks: {

callback: (val) => {

return val + '°C';

},

},

}],

}

}

});

let druk = new Chart("drukS" + id, {

type: "line",

data: {

labels: [],

datasets: [{

fill: false,

lineTension: 0,

backgroundColor: "rgba(43,140,190,1.0)",

borderColor: "rgba(0,0,255,0)",

data: []

}]

},

options: {

legend: {display: false},

title: {

display: true,

text: "Sensor " + id + " Luchtdruk"

},

animation: {

duration: 0

},

scales: {

yAxes: [{

ticks: {

callback: (val) => {

return val + 'Hpa';

},

},

}],

}

}

});

let vocht = new Chart("vochtS" + id, {

type: "line",

data: {

labels: [],

datasets: [{

fill: false,

lineTension: 0,

backgroundColor: "rgba(254,153,20,1.0)",

borderColor: "rgba(0,0,255,0)",

data: []

}]

},

options: {

legend: {display: false},

title: {

display: true,

text: "Sensor " + id + " Luchtvochtigheid"

},

animation: {

duration: 0

},

scales: {

yAxes: [{

ticks: {

callback: (val) => {

return val + '%';

},

},

}],

}

}

});

let gas = new Chart("gasS" + id, {

type: "line",

data: {

labels: [],

datasets: [{

fill: false,

lineTension: 0,

backgroundColor: "rgba(128,196,28,1.0)",

borderColor: "rgba(0,0,255,0)",

data: []

}]

},

options: {

legend: {display: false},

title: {

display: true,

text: "Sensor " + id + " Gas"

},

animation: {

duration: 0

},

scales: {

yAxes: [{

ticks: {

callback: (val) => {

return val + 'Kohm';

},

},

}],

}

}

});

charts.push(temp)

charts.push(druk)

charts.push(vocht)

charts.push(gas)

return charts;

}

function toggleSettings() {

settingsBar = document.getElementById('settingsBar');

if (settingsBar.style.display == "block")

{

settingsBar.style.display = "none";

} else {

settingsBar.style.display = "block";

}

}

document.getElementById("clearButton").onclick=async() => {

await clear();

};

document.getElementById("kalibreer").onclick=async() => {

kalibreer();

};

async function clear() {

try

{

await fetch('/api/clear');

} catch (e){}

prompt = document.getElementById("prompt")

prompt.innerHTML = "<h1>Database is gereset!</h1><button onclick='closePrompt()'>OK</button>"

togglePrompt()

}

async function kalibreer() {

prompt = document.getElementById("prompt")

prompt.innerHTML = '<h1>Kalibreer</h1><div><label for="sensor">Kies een sensor:</label><select name="sensor" id="sensor-select"><option value="sensor1">Sensor 1</option><option value="sensor3">Sensor 3</option></select></div><div><label for="meeting">Kies een meeting:</label><select name="meeting" id="meeting-select"><option value="temperatuur">Temperatuur</option><option value="luchtdruk">Luchtdruk</option><option value="luchtvochtigheid">Luchtvochtigheid</option><option value="gas">Gas</option></select></div><p>Geef een waarde in dat van de sensor moet worden afgetrokken.</p><div><input type="number" value="0" min="-20" max="20" /></div><button onclick="closePrompt()">OK</button>'

togglePrompt()

}

function togglePrompt() {

wrapper = document.getElementById("prompt-wrapper")

if (wrapper.style.display == "flex")

{

wrapper.style.display = "none";

} else {

wrapper.style.display = "flex";

}

}

function closePrompt()

{

wrapper = document.getElementById("prompt-wrapper")

wrapper.style.display = "none";

clearPrompt();

}

function clearPrompt()

{

prompt = document.getElementById("prompt")

prompt.innerHTML = '';

}

chartS1 = initChart(1)

chartS3 = initChart(3)

setInterval(updateChart, 1000,1,chartS1);

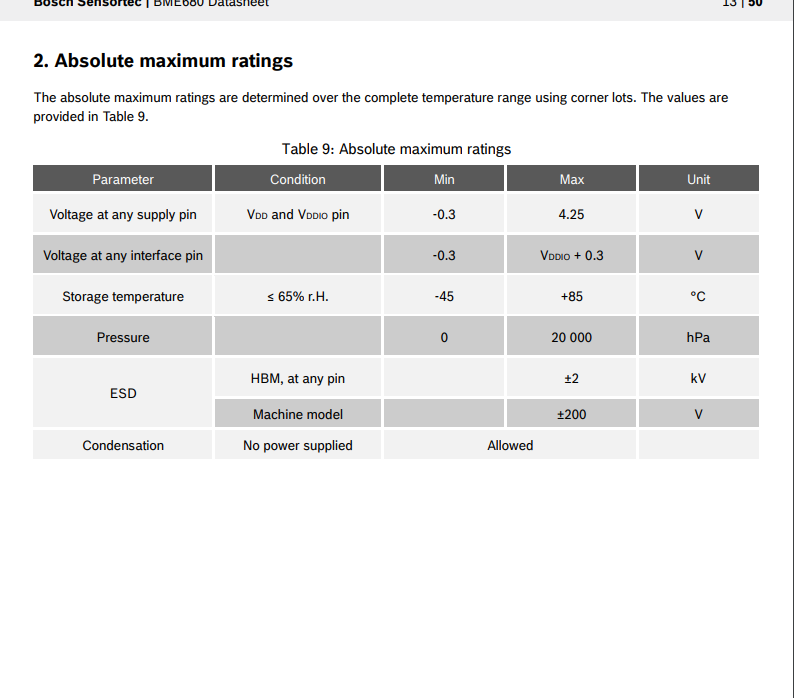
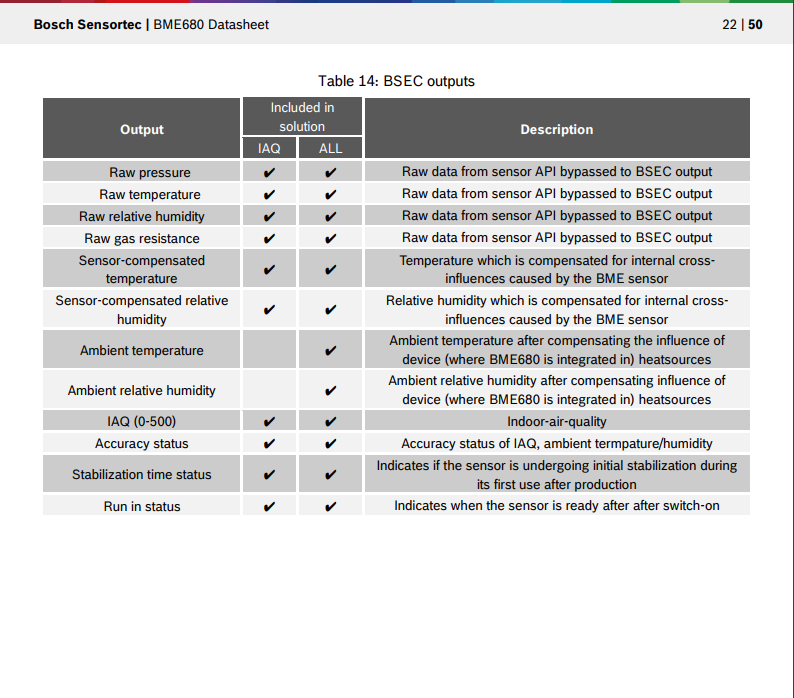
setInterval(updateChart, 1000,3,chartS3);

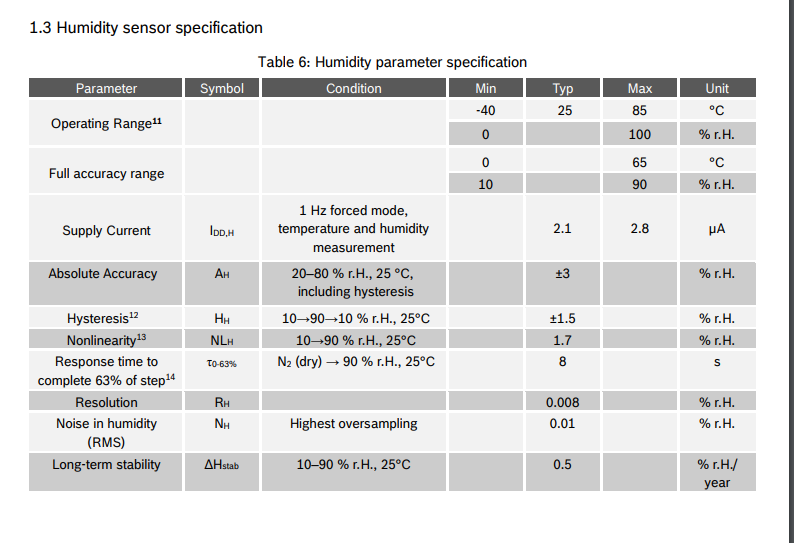
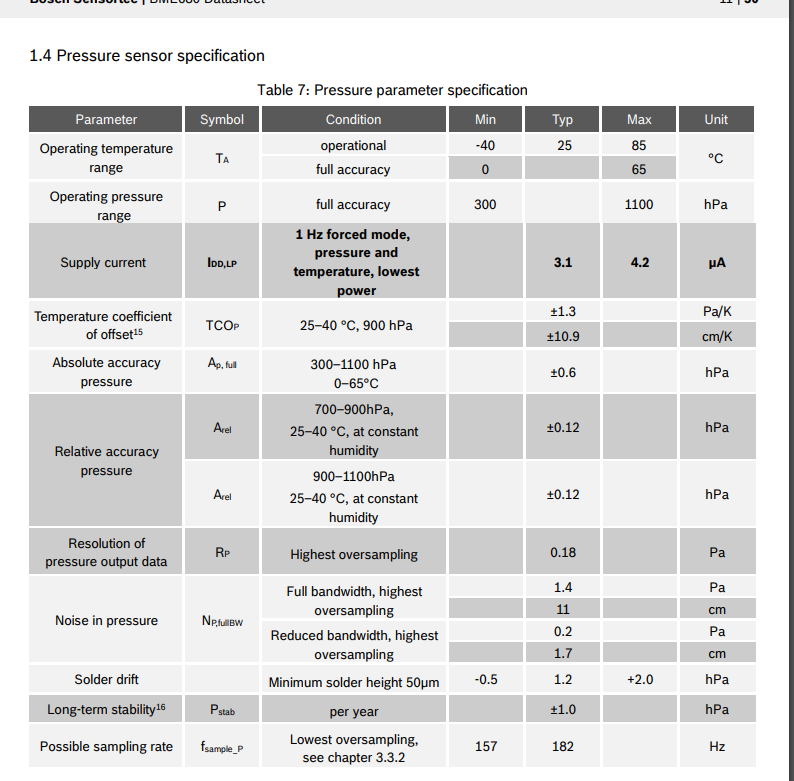
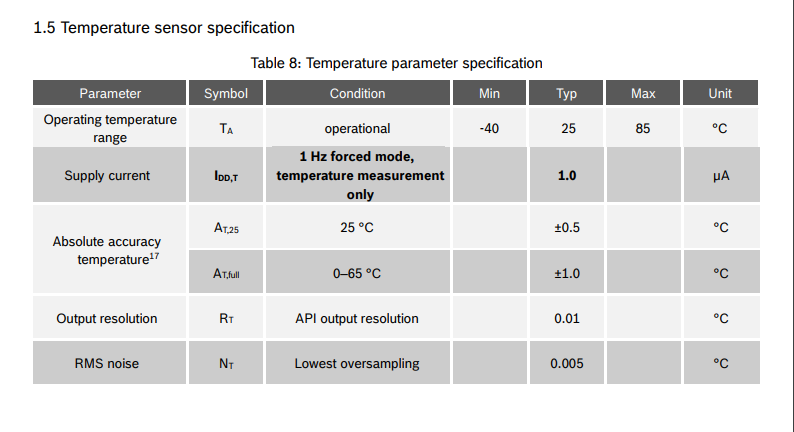
</script>

</html>

## Databladen

[BME680.pdf](https://github.com/quintenVLOT/eindproject/blob/main/datasheets/BME680.pdf)





Enkel de relevante delen uit een datablad. Het volledige datablad van aan ATMega328P [2] heeft geen meerwaarde.