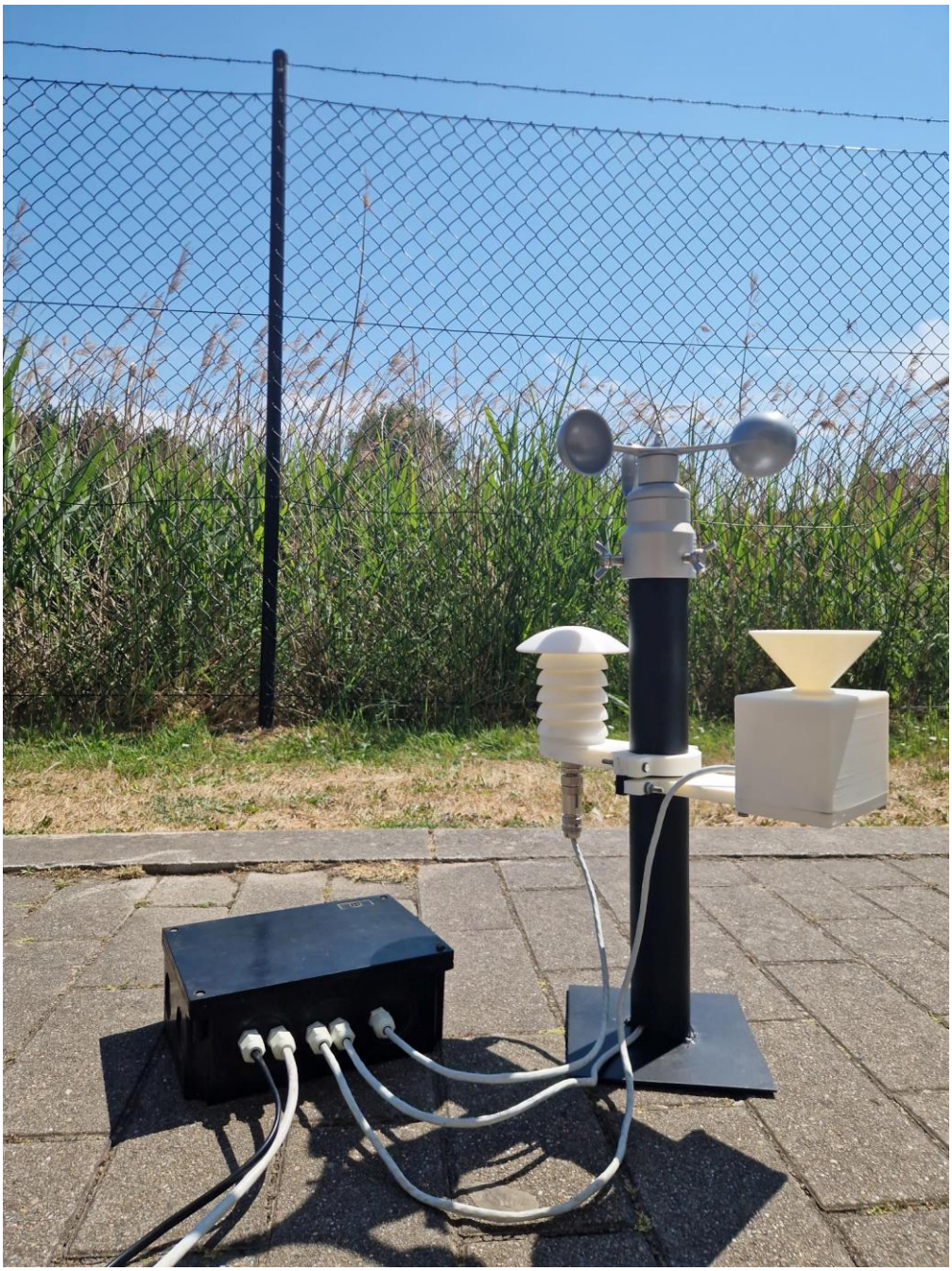
EINDWERK WEERSTATION



Door Jasper Lanoote
‘t Saam Campus Cardijn
Studiedomein: informatica- en communicatiewetenschappen
Onder begeleiding van Dhr. Daan Blomme, Dhr. Hannes De Kezel & Dhr. Johannes Debusschere
Schooljaar 2024-2025

Inhoudsopgave

Figuurlijst

Woord vooraf

Ik heb gekozen voor dit project omdat het goed van toepassing kan zijn thuis aangezien er een home assistant aanwezig is bij ons thuis. De waarden die hieruit komen kan ik dan ook goed gebruiken. Daarnaast was ik ook erg benieuwd naar hoe je digitaal de hoeveelheid regen kan meten want hier bestaat namelijk niet echt een kant en klare sensor voor.

Ik bedank in de eerste plaats graag mijn papa, die me veel geholpen heeft in het bedenken van heel wat kleine technische dingetjes. Hij heeft er mee voor gezorgd dat ik een goede windsnelheidsmeter kon verkrijgen, welke een grote meerwaarde is voor dit project.

Naast mijn papa bedank ik graag ook de leerkrachten die me de verschillende nodige vaardigheden geleerd hebben in de voorbije jaren om dit project te kunnen realiseren. Specifiek bedank ik graag Dhr. Daan Blomme, Dhr. Hannes De Kezel en Dhr. Johannes Debusschere.

1 Inleiding

1.1. Probleemoriëntering

Betrouwbare weersomstandigheden zijn cruciaal voor toepassingen zoals home assistants, die onder andere ventilatie en verwarming regelen. Zo kan in een home assistant ingesteld worden dat de ventilatie niet actief is wanneer het buiten heel koud of vochtig is, om te voorkomen dat er te koude of vochtige lucht binnentrekt. Commerciële weerstations bieden vaak slechts een beperkt aantal meetparameters voor een hoge prijs. Door zelf een weerstation te ontwerpen, kan er gekozen worden welke sensoren worden aangesloten op het systeem en kan er eenvoudig uitgebreid worden zodra er nieuwe sensoren beschikbaar zijn. Dit biedt een flexibele en schaalbare oplossing die perfect aansluit bij de specifieke behoeften van de woning.

1.1.1. De onderzoeksvraag en deelvragen

1.1.1.1. Onderzoeksvraag

Hoe kan een weerstation gebaseerd op een Raspberry Pi worden ontwikkeld om nauwkeurige metingen van weersomstandigheden te integreren met Home Assistant?

1.1.1.2. Deelvragen:

1. Welke sensoren zijn belangrijk om te integreren in het systeem?
2. Welke stappen zijn nodig om de sensordata via een Raspberry Pi en Django op te halen, te verwerken en beschikbaar te maken voor Home Assistant?
3. Hoe kan de verzamelde data goed en overzichtelijk worden gevisualiseerd via een webinterface en in Home Assistant?
4. Hoe betrouwbaar en nauwkeurig moeten de meters zijn om bruikbaar te zijn? (Als hij 50% van de tijd geen data geeft ben je er weinig mee. Als hij 5 graden naast de effectieve temperatuur is ben je er eveneens vrij weinig mee.)

1.2. Opzet van dit project

Het doel van dit project is de ontwikkeling van een weerstation dat in staat is om verschillende weersomstandigheden te meten, zoals temperatuur, luchtvochtigheid, luchtdruk, windrichting en windsnelheid. De sensoren zullen worden aangesloten op een Raspberry Pi, die via een Django-server de data verwerkt en opslaat in een database.

Het eindresultaat moet een compleet weerstation zijn dat nauwkeurig verschillende weersomstandigheden meet, waaronder regenmetingen met een digitale pluviometer, windsnelheid en -richting, en digitale metingen van

temperatuur, luchtvochtigheid en atmosferische druk. Daarnaast zal het systeem mogelijks worden uitgebreid met extra sensoren voor gasdetectie, zoals CO, CO_2, UV en O_3 , om ook luchtkwaliteit te monitoren.

Daarnaast zal de verzamelde data via een webpagina gevisualiseerd worden, zodat de gebruiker zowel de live data als de historische gegevens kan bekijken. Dit systeem zal in communicatie staan met homeassistent voor het regelen van de ventilatie en verwarming, afhankelijk van de weersomstandigheden zoals luchtvochtigheid en temperatuur.

1.3. Onderzoeksopzet

1. **Bepalen van te verzamelen gegevens**
   De eerste stap is het bepalen van welke soorten gegevens er verzameld moeten worden. Het is essentieel om duidelijk te krijgen welke gegevens belangrijk zijn voor het verder uitwerken van dit project.
2. **Kiezen van benodigde sensoren**
   Na het vaststellen van welke gegevens er gebruikt gaan worden, moeten geschikte sensoren gekozen worden. Hierbij wordt gekeken naar de verschillende opties en moet bepaald worden welke sensoren het beste aansluiten bij de te verzamelen gegevens en de nauwkeurigheid die nodig is voor het project.
3. **Uitlezen van sensoren met de hardware**
   Zodra de sensoren zijn gekozen, wordt de hardware voorbereid. De sensoren worden aangesloten op een Raspberry Pi en een Python-programma wordt geschreven om de sensoren uit te lezen.
4. **Uitlezen van de sensoren rechtstreeks vanuit Django**
   Nadat het eenvoudige Python-programma voor iedere sensor is geschreven, worden deze programma's omgezet naar iets wat rechtstreeks vanuit Django kan worden uitgevoerd. Dit zal gebeuren aan de hand van een apart Python script die de data naar een websocket stuurt.
5. **Kiezen van opslagmethode voor gegevens**
   De gegevens moeten op een efficiënte manier worden opgeslagen, zodat ze gemakkelijk toegankelijk blijven voor verdere verwerking. Er zal gebruik worden gemaakt van een SQLite-database in het Django-framework.
6. **Integreren van gegevensverzameling in een webapplicatie**
   De verzamelde gegevens worden vervolgens geïntegreerd in een webapplicatie. Er zal een website rechtstreeks op de Django-server worden gehost die het mogelijk maakt om de gegevens op een gebruiksvriendelijke manier te presenteren, bijvoorbeeld via een dashboard of andere visualisaties.
7. **Integratie met Home Assistant**
   Tot slot wordt er onderzocht hoe de verzamelde gegevens toegankelijk kunnen worden gemaakt voor Home Assistant. Dit kan bijvoorbeeld door middel van een API waarmee de gegevens vanuit de databank opgevraagd kunnen worden en geïntegreerd zullen worden in de verschillende functionaliteiten van het smart home-systeem.

2 Onderzoeksfase

2.1. Literatuurstudie

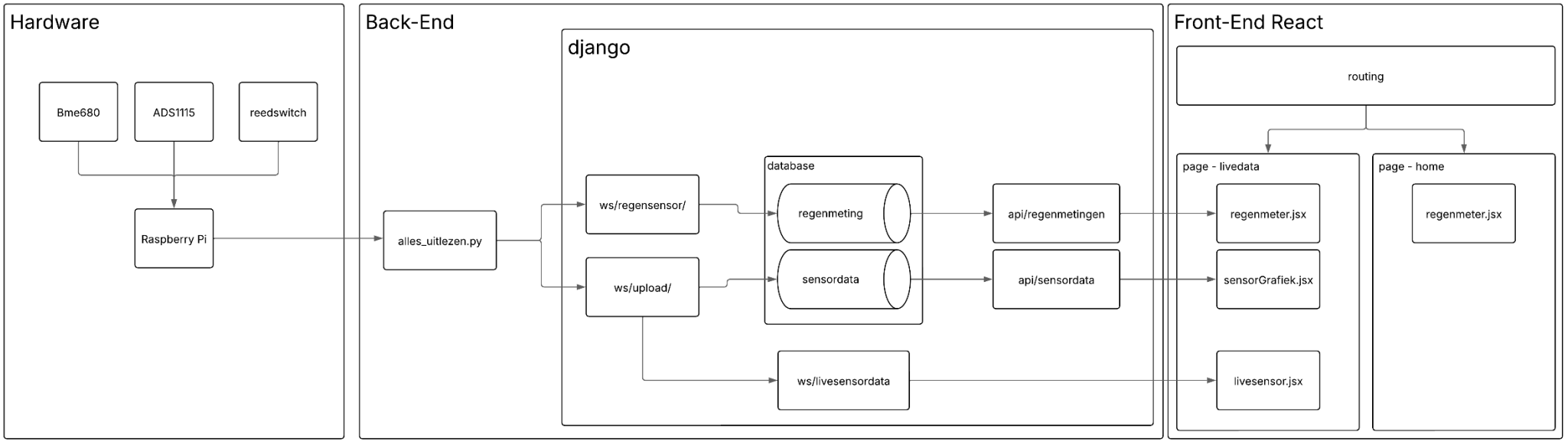
Regenmeter - Tipping Bucket Mechanisme: Uit de informatie van de video van ExplainingComputers is het *tipping bucket* mechanisme naar boven gekomen als een effectieve manier om regen te meten (ExplainingComputers, 2024). Dit systeem is eenvoudig te implementeren met een microcontroller en kan worden aangepast voor mijn project. Het principe werkt door het meten van het aantal keren dat een emmer met een bepaalde hoeveelheid regen wordt omgekeerd, wat precies meet hoeveel regen er valt.
Temperatuur- en Luchtvochtigheidssensor - Bosch BME680: De Bosch BME280 sensor die in een andere studie wordt genoemd heeft me geholpen om de geschikte sensor voor temperatuur- en luchtvochtigheidmetingen te vinden (ExplainingComputers, 2021). Hoewel de BME680 gebruikt wordt, biedt deze sensor dezelfde basisfunctionaliteit, met een extra mogelijkheid om luchtkwaliteit te meten. Het is belangrijk dat de sensor buiten gebruikt kan worden, wat goed aansluit bij de eisen van dit project. De nauwkeurigheid van de sensor is essentieel voor betrouwbare metingen en de video bevestigde dat het buitengebruik mogelijk is.
Windsnelheid - Analoge Windsnelheidsmeter: dit is een 4-20ma windsnelheid sensor. Het was nodig om een extra sensor toe te voegen om dit analoge signaal te kunnen uitlezen op een Raspberry Pi. De ADS1115 analoge-naar-digitaal omzetter was de oplossing, zoals beschreven in de video van VolosProjects (Volos Projects, 2022). Deze module helpt me om de analoge spanningswaarden om te zetten naar een digitale waarde die vervolgens kan gebruikt worden om de windsnelheid in km/h te berekenen. Aangezien de sensor moet gevoed worden met een 24v voeding moest er een manier zijn om de uit te lezen waarde te verlagen naar max 5 volt. Dit kan gedaan worden aan de hand van een shunt resistor zodat er een spanning over de weerstand kan gemeten worden die max 5 volt wordt.
Temperatuur- en Vochtigheidssensor Bescherming - Stevenson Screen: Een belangrijk aspect van het project is de bescherming van de temperatuursensor tegen direct zonlicht en regen. Door onderzoek is duidelijk dat de Stevenson Screen (James Clough, 2016), een ideale behuizing voor het beschermen van de sensoren. Dit biedt een schaduwrijke, geventileerde ruimte, zodat de sensor niet direct in contact komt met de elementen.

2.2. Belangrijke ontdekkingen en technieken

- **Tipping Bucket techniek:** Dit is de basis voor het meten van regenhoeveelheden, waarbij het aantal keren dat een emmer omgekeerd wordt, aangeeft hoeveel regen is gevallen.
- **I2C protocol:** Dit protocol maakt communicatie tussen verschillende sensoren en microcontrollers eenvoudiger, zoals tussen de Bosch BME680 en de Raspberry Pi. Dit protocol wordt ook gebruikt voor de ADS1115.
- **Shunt resistor:** dit zorgt ervoor dat het mogelijk wordt om een 4-20ma sensor uit te lezen met een Raspbarry Pi en de ADS1115

3 Ontwerpfase

3.1. Schema



Figuur 1 overzichtsschema

In dit schema kan er al goed bekeken worden hoe het project eruitziet en in elkaar zit. Zo kan de rest van het verslag goed gevolgd worden. Helemaal links kan de hardware bekeken worden. Er zijn 3 componenten verbonden met een Raspberry Pi. Als er naar rechts opgeschoven wordt kan de Back-End bekeken worden. Hierin worden alle 3 de sensoren uitgelezen met eenzelfde bestand alles_uitlezen.py. Daarna wordt de data verzonden naar een Django server via twee websockets, namelijk regensensor en upload. Regensensor stuurt de data naar een tabel genaamd regenmetingen in de database. Upload zet ook de data in de database in tabel sensordata maar stelt de data ook direct nog eens live beschikbaar via een websocket. De data die in de database staat is beschikbaar via 2 api's namelijk regenmetingen en sensordata. Deze data wordt opgehaald door enkele componenten in de React front-end. Livesensor.jsx leest de websocket livesensordata, dit zorgt ervoor dat de data live kan bekeken worden op de website. Hiernaast wordt er bij regenmeter.jsx de api regenmetingen uitgelezen. In deze component wordt de data weergegeven in een duidelijke grafiek. Ook sensorGrafiek.jsx leest een api uit, namelijk sensordata. In deze component wordt de data ook goed voorgesteld in een grafiek. De website heeft 2 pagina's, home en livedata. Op de homepagina staat er gewoon wat info over het project. Op de pagina livedata kun je de livedata en de grafieken zien.

3.2. Hardware

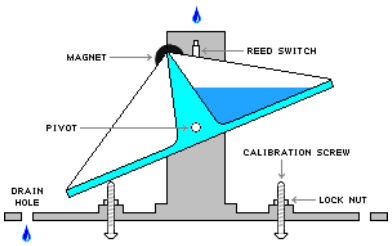
3.2.1. Mechanisch

Voor het ontwerp is ervoor gekozen om alle componenten te monteren op een buis met een diameter van 42 mm. Om de temperatuursensor te beschermen, is er een omhulsel geprint volgens het principe van een **Stevenson screen**. Dit ontwerp zorgt ervoor dat er voldoende luchtcirculatie is rond de sensor, terwijl directe blootstelling aan neerslag wordt voorkomen. Het omhulsel is vervaardigd uit ABS, het meest geschikte materiaal dat op school beschikbaar was vanwege de weerstand tegen Uv-straling en water.

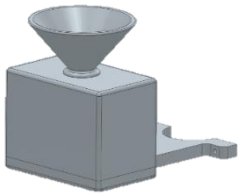
Figuur 3 Stevenson screen sensor bescherming

Figuur 2 geheel project

Het **tipping bucket-mechanisme** is eveneens geprint in ABS. Het systeem bestaat uit een kantelbakje met twee compartimenten, geplaatst onder een trechter die het regenwater naar het midden van het bakje leidt. Zodra één compartiment vol is, kantelt het bakje automatisch naar de andere zijde. Aan het bakje is een magneet bevestigd die bij elke kantelbeweging een reedswitch activeert. Door het aantal kantelingen te registreren, kan nauwkeurig worden berekend hoeveel neerslag er is gevallen, aangezien elke kanteling overeenkomt met een vooraf bekende hoeveelheid water. Ook dit mechanisme is bevestigd aan de buis, op vergelijkbare wijze als de temperatuursensor, en wordt beschermd door een omhulsel waarop de trechter is gemonteerd.



Figuur 5 tipping bucket mechanisme



Figuur 4 geheel van de pluviometer

Bovenaan het geheel is de **windsnelheidssensor** geplaatst. Deze sensor is ontworpen om direct op een buis met een diameter van 42 mm te worden gemonteerd. De bijbehorende bekabeling is netjes weggewerkt binnenin de buis, wat bijdraagt aan een nette en functionele afwerking van het geheel.

Figuur 6 Kriwan INT 10 anemometer

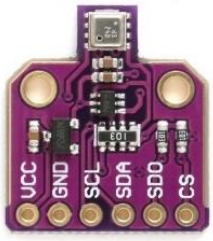
De **elektronica** is centraal samengehouden in een **aftakdoos**, waarin alle verbindingen samenkomen. De kabels worden via de onderzijde van de doos binnengeleid door middel van wartels, die zorgen voor een waterdichte en stevige doorvoer. Dit maakt het geheel geschikt voor gebruik in een buitenomgeving.

3.2.2. Elektrisch

Voor de meting van verschillende omgevingswaarden wordt gebruikgemaakt van een aantal sensoren die in verbinding staan met een Raspberry Pi. Een Raspberry Pi is eigenlijk een mini computer waar er een aantal gpio pinnen op zitten. De Raspberry Pi die gebruikt is, is een 4 model B 2gb.

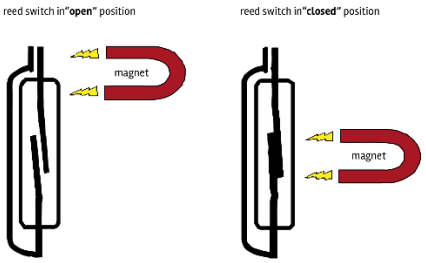
Figuur 7 Raspbarry Pi 4 model B 2gb

De communicatie tussen de sensoren en de Raspberry Pi verloopt hoofdzakelijk via het I²C-protocol. Dit is een seriële interface die werkt met twee lijnen: een datalijn (SDA) en een kloklijn (SCL). Dankzij het gebruik van unieke adressen kunnen meerdere I²C-apparaten via dezelfde pinnen worden aangesloten, wat het systeem overzichtelijk en uitbreidbaar maakt. In het omhulsel volgens het Stevenson screen-principe zit er een Bosch BME680-sensor. Deze compacte sensor meet temperatuur, luchtvochtigheid, luchtdruk en gasconcentratie en biedt dus een brede set aan omgevingsdata binnen een klein formaat. De BME680 communiceert via I²C met de Raspberry Pi, wordt gevoed met 5 volt, en deelt een gemeenschappelijke massa met de Pi. Omdat de sensor afgeschermd is tegen directe neerslag maar toch voldoende luchtcirculatie krijgt, kunnen nauwkeurige en stabiele metingen worden uitgevoerd.



Figuur 8 Bosch bme680

Voor de meting van neerslag wordt gebruikgemaakt van een tipping bucket-mechanisme waarbij een bakje met twee compartimenten kantelt zodra één zijde gevuld is met regenwater. Elke kantelbeweging activeert een reedswitch, die in het systeem is verbonden met een 3.3V voedingspin en GPIO22 van de Raspberry Pi. Telkens wanneer de magneet aan het bakje langs de reedswitch beweegt, sluit het contact kort en wordt een puls geregistreerd. Aangezien het volume per kanteling bekend is, kan op basis van het aantal geregistreerde pulsen nauwkeurig worden berekend hoeveel neerslag er is gevallen.

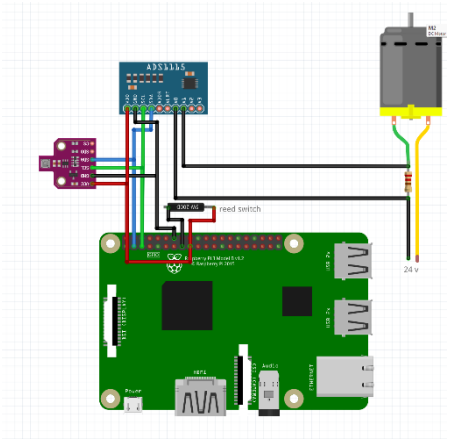


Figuur 9 werking reedswitch

De windsnelheid wordt gemeten met een Kriwan INT10 anemometer, een robuuste industriële sensor die gevoed wordt met 24 volt. De negatieve uitgang van deze sensor is via een shuntresistor van 235 ohm verbonden met de massa. Een shuntweerstand is een precieze, lage weerstand die wordt gebruikt om de stroom door de sensor te meten aan de hand van de spanningsval die over de weerstand ontstaat. Deze spanning wordt gemeten met een ADS1115, een 16-bits analoog-naar-digitaalomzetter die eveneens via I²C met de Raspberry Pi is verbonden. Zo kan de rotatiesnelheid van de sensor worden omgezet naar een spanningswaarde en van daaruit worden omgerekend naar een windsnelheid.

Figuur 10 ADS1115
analog-digital converter

Alle elektronische componenten zijn centraal ondergebracht in een aftakdoos, waarin de bedrading netjes samenkomt. De kabels worden langs de onderzijde van de doos ingevoerd via wartels, wat zorgt voor een waterdichte en mechanisch stevige afwerking. De shuntweerstand, ADS1115 en overige verbindingen bevinden zich eveneens in deze doos. Dankzij de modulaire opbouw en het gebruik van standaardprotocollen zoals I²C blijft het systeem overzichtelijk, onderhoudsvriendelijk en eenvoudig uitbreidbaar.

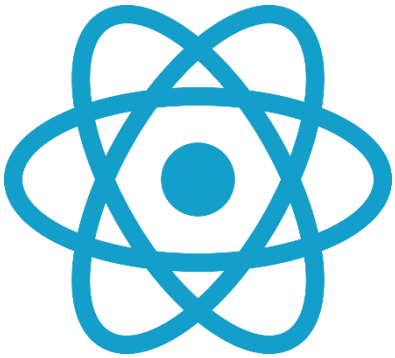


Figuur 11 elektrisch schema

3.3. Software

3.3.1. Front-end

3.3.1.1. Inleiding React



Figuur 12 React logo

Voor de ontwikkeling van de frontend wordt gekozen voor React, een JavaScript-bibliotheek die gebruikt wordt om interactieve gebruikersinterfaces te bouwen. React is gemaakt door Meta (Facebook) en is vooral sterk in het bouwen van zogenaamde *single-page applications* waarbij de inhoud van een pagina dynamisch kan worden aangepast zonder dat de volledige pagina opnieuw geladen hoeft te worden. Dit zorgt voor een vlotte en moderne gebruikerservaring.
Het basisprincipe van React is dat de gebruikersinterface wordt opgebouwd uit kleine, herbruikbare componenten. Elk component is een stuk code dat een bepaald deel van de interface beschrijft, bijvoorbeeld een grafiek, een knop of een datakaart. Deze componenten kunnen onderling gegevens uitwisselen via *props* (eigenschappen) en kunnen hun eigen interne toestand bijhouden via *state*. Wanneer de gegevens of de toestand veranderen, zorgt React automatisch voor het bijwerken van enkel die onderdelen van de interface waar nodig, zonder onnodige herlaadacties.
React maakt gebruik van JSX, een uitbreiding op JavaScript die het mogelijk maakt om HTML-achtige structuur direct in JavaScript-code te schrijven. Dit zorgt voor een intuïtieve manier van werken waarbij logica en opmaak dicht bij elkaar staan.
In dit project wordt React gebruikt om de data die wordt doorgestuurd door de Raspberry Pi op een overzichtelijke en interactieve manier te tonen aan de gebruiker. Denk hierbij aan real-time grafieken van temperatuur, luchtvochtigheid en windsnelheid, visuele weergave van de hoeveelheid neerslag, en tabellen met meetwaarden. De frontend maakt gebruik van componenten die automatisch vernieuwen wanneer nieuwe data beschikbaar is, bijvoorbeeld via een API of WebSocket-verbinding.
Door de keuze voor React is het mogelijk om snel te ontwikkelen, code gestructureerd te houden en eenvoudig nieuwe functionaliteiten toe te voegen. Bovendien wordt React breed ondersteund, goed gedocumenteerd, en veel gebruikt in de industrie, wat het een duurzame keuze maakt voor dit project.

3.3.1.2. Code

3.3.1.2.1. Livesensor.jsx

Het importeren van React is nodig om een React component te maken. useState en useEffect worden gebruikt om de status van de component bij te houden en de websocket verbinding te maken. Het css bestand wordt ook geïmporteerd voor de opmaak van de componenten.

```
import React, { useEffect, useState } from 'react';
import './Livesensor.css'
```

Hier wordt het component gedefinieerd Livesensor. Daarnaast wordt met useState een tijdelijke plaats aangemaakt om sensorgegevens op te slaan.

```
export default function Livesensor() {
  const [sensorData, setSensorData] = useState(null);
```

Zodra het component is ingeladen wordt er via useEffect een verbinding opgezet met een lokale websocket die realtime de data van de sensoren verstuurt.

```
  useEffect(() => {
    const ws = new WebSocket('ws://localhost:8000/ws/liveSensorData/');
```

Wanneer de verbinding is geopend gaat er in de console een bericht verschijnen die dit bevestigt. Als er dan data binnenkomt wordt deze opgeslagen in sensorData met setSensorData.

```
    ws.onopen = () => {
      console.log('Verbonden met WebSocket-server');
    };

    ws.onmessage = (event) => {
      const data = JSON.parse(event.data);
      setSensorData(data);
    };
```

Als er een fout ontstaat met de websocket wordt de fout doorgegeven in de console. Als het component niet meer is ingeladen zal de websocket verbinding verbreken.

```
    ws.onerror = (event) => {
      console.error('WebSocket-fout:', event);
    };

    ws.onclose = () => {
      console.log('WebSocket-verbinding gesloten');
    };

    return () => {
      if (ws) {
        ws.close();
      }
    };
  }, []);
```

Hier wordt er een tabel aangemaakt om de gegevens gestructureerd te tonen.

```
  return (
    <div className='flexbox'>
      <h2>Live sensor data</h2>
      {/* Controleer of we data ontvangen hebben */}
      <div>
        {sensorData ? (
          <table>
            <thead>
              <tr>
                <th> Temperatuur °C </th>
                <th> Vochtigheid % </th>
                <th> Druk hPa </th>
                <th> Gas PPM </th>
                <th> Wind Snelheid km/h </th>
              </tr>
            </thead>
            <tbody>
              <tr>
                <td>{sensorData.temperature}</td>
                <td>{sensorData.humidity}</td>
                <td>{sensorData.pressure}</td>
                <td>{sensorData.gas}</td>
                <td>{sensorData.wind_speed_kmh}</td>
              </tr>
            </tbody>
          </table>
        ) : (
          <p>Wachten op gegevens...</p>
        )}
      </div>
    </div>
  );
};
```

3.3.2. Backend

3.3.2.1. Inleiding Python en Django

De backend van het project is gebouwd met Python en het Django-framework. Django is een krachtig webframework dat het mogelijk maakt om op een gestructureerde en efficiënte manier webapplicaties te ontwikkelen. Binnen dit project fungeert Django als centrale schakel tussen de hardware (zoals sensoren op de Raspberry Pi) en de frontend die de data aan de gebruiker toont.
Django biedt van nature ondersteuning voor het opzetten van een duidelijke projectstructuur met onder andere views, modellen en routes. Hierdoor kunnen inkomende en uitgaande gegevens overzichtelijk worden beheerd. Om live data naar de browser te kunnen sturen, werd Django uitgebreid met Django Channels. Deze extensie maakt het mogelijk om naast gewone HTTP-verzoeken ook WebSocket-verbindingen te gebruiken. WebSockets zijn ideaal voor real-time toepassingen omdat ze een constante verbinding tussen server en client behouden. Zodra er dus nieuwe sensorwaarden beschikbaar zijn, kunnen deze onmiddellijk doorgestuurd worden naar de gebruiker zonder dat de pagina opnieuw geladen hoeft te worden.

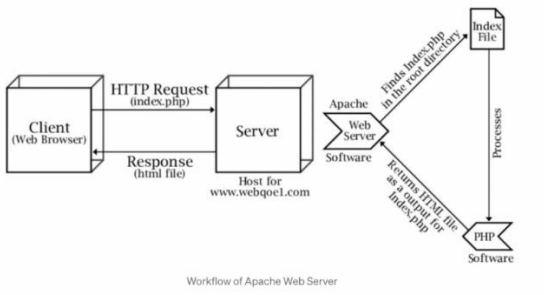
Figuur 14 Django logo

Figuur 13 Python logo

Op de Raspberry Pi worden de verschillende sensoren aangestuurd via Python-scripts. Deze scripts lezen de actuele gegevens uit, zoals temperatuur, luchtdruk of windsnelheid, en sturen deze vervolgens door naar de backend. Vanuit Django worden deze data in realtime via de WebSocket-verbinding naar de frontend gestuurd. Hierdoor verschijnt de nieuwe informatie direct op het scherm van de gebruiker, zonder vertraging.
Django werd gekozen als backendframework vanwege zijn betrouwbaarheid, uitgebreide documentatie en goede uitbreidbaarheid. Bovendien past het goed bij het gebruik van Python op de Raspberry Pi en werkt het uitstekend samen met moderne frontendtechnologieën zoals React. Dankzij Django en Channels is het mogelijk om een backend te bouwen die niet alleen stabiel en overzichtelijk is, maar ook voldoet aan de eisen van een moderne, realtime gegevensstroom.

3.3.2.2. Apache HTTP Server

Apache is een gratis en open source web server software. Deze webserver software is de meest gebruikte in de wereld. Een webserver ontvangt een aanvraag (request) van een client, meestal is dit een webbrowser, en stuurt dan de juiste webpagina of bestanden terug.
Apache wordt geconfigureerd via het configuratiebestand httpd.conf. In dit bestand worden belangrijke instellingen vastgelegd, zoals de locatie van webbestanden en beveiligingsinstellingen.



Figuur 15 schema werking Apache

3.3.2.3. Code

3.3.2.3.1. Het uitlezen van de sensoren

3.3.2.3.2. Gebruikte bibliotheken

- **asyncio**: Wordt gebruikt om twee while-lussen gelijktijdig te laten draaien binnen hetzelfde programma. Dit maakt het mogelijk om verschillende sensortaken parallel en efficiënt uit te voeren.
- **adafruit_bme680** en **Adafruit_ADS1x15**: Deze bibliotheken zorgen voor het uitlezen van de **BME680** (temperatuur, vochtigheid, luchtdruk en gas) en de **ADS1115** (analoog-naar-digitaal converter, gebruikt voor de windsnelheid).
- **json**: Wordt gebruikt om de sensordata in **JSON-formaat** te structureren zodat deze eenvoudig verstuurd kan worden via een netwerkverbinding.
- **time**: Wordt gebruikt om tijdsmetingen uit te voeren, bijvoorbeeld om 60 seconden lang pulsen van de reed-switch te tellen.
- **websockets**: Maakt het mogelijk om een **WebSocket-verbinding** op te zetten met de backend (Django-server), zodat sensordata real-time verzonden kan worden.
- **RPi.GPIO**: Zorgt voor toegang tot de **GPIO-pinnen van de Raspberry Pi**, nodig om de reed-switch (regenmeter) uit te lezen.
- **threading**: Wordt gebruikt om twee asynchrone functies (async functies) in **parallelle threads** uit te voeren. Elke thread draait een aparte asyncio event loop, wat nodig is omdat asyncio.run() slechts in één event loop tegelijk kan draaien per thread.
- **board**: Een Adafruit-bibliotheek die de I²C-pinnen van de Raspberry Pi toegankelijk maakt op een abstracte en platformonafhankelijke manier. Deze wordt gebruikt om de **BME680-sensor via I²C** aan te sturen.

```
import asyncio
import json
import time
import websockets
import RPi.GPIO as GPIO
import threading
import board
import adafruit_bme680
import Adafruit_ADS1x15
```

3.3.2.3.3. Initialisatie van GPIO, sensoren en WebSocket-URL's

In de onderstaande code worden de GPIO-pinnen ingesteld, de adressen van de sensoren opgegeven en de WebSocket-URL's gedefinieerd:

```
# GPIO-instellingen voor regen sensor
GPIO.setmode(GPIO.BCM)
button_pin = 22  # GPIO-pin voor de drukknop
GPIO.setup(button_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

# WebSocket URL voor het versturen van regenval-data
REGEN_WS_URL = "ws://localhost:8000/ws/regensensor/"

# Variabelen voor de regen sensor
calibratie_factor = 132  # Aantal keer overgeschakeld per 100 mm regen
running = True

# Setup ADS1115 voor windsnelheid met de ADS1115 module
adc = Adafruit_ADS1x15.ADS1115(address=0x48, busnum=1)
GAIN = 1

# Initialiseer de I2C bus voor de BME680 sensor
i2c = board.I2C()
sensor = adafruit_bme680.Adafruit_BME680_I2C(i2c)

# WebSocket URL van de server voor sensor data
SENSOR_WS_URL = "ws://localhost:8000/ws/upload/"
```

3.3.2.3.4. Verbinding actief houden

De onderstaande functie stuurt elke 30 seconden een ping naar de WebSocket-server. Dit voorkomt dat de verbinding automatisch wordt verbroken bij inactiviteit:

```
async def ping(websocket):
    try:
        while True:
            await websocket.ping()
            await asyncio.sleep(30)
    except Exception as e:
        print(f"Ping error: {e}")
```

3.3.2.3.5. Sensorwaarden uitlezen en verzenden

De onderstaande functie leest continu de gegevens uit van de BME680- en ADS1115-sensoren. De waarden worden afgerond en gestructureerd in JSON-formaat, waarna ze via een WebSocket-verbinding naar de backend worden verzonden.
Daarnaast wordt in deze functie ook de windsnelheid berekend. De ADS1115-sensor meet het spanningsverschil over de shunt resistor van de windsensor. Deze spanning wordt omgerekend naar een windsnelheid met behulp van een lineaire schaal tussen de minimale en maximale spanningswaarden (0,97 V tot 4,7 V) en de bijbehorende windsnelheid (0 tot 35 m/s). De gemeten snelheid wordt vervolgens omgerekend naar kilometer per uur (km/h). Het gebruik van round( ) zorgt ervoor dat de waarden netjes worden afgerond op een geschikt aantal decimalen, zodat ze eenvoudiger te interpreteren zijn en minder datavolume innemen bij verzending.

```
async def send_sensor_data():
    while True:
        try:
            # Lees sensorwaarden uit
            temperature = round(sensor.temperature, 1)
            …
            # Lees de waarde van de ADS1115 en bereken de windsnelheid
            value = adc.read_adc_difference(0, gain=GAIN)
            voltage = value * (4.096 / 32768.0)
            min_volt = 0.97
            max_volt = 4.7
            min_wind = 0
            max_wind = 35

            windsnelheid = ((voltage - min_volt) * (max_wind - min_wind) /
(max_volt - min_volt)) + min_wind
            windsnelheid = max(windsnelheid, 0)
            windsnelheid_kmph = round(windsnelheid * 3.6, 2)  # km/h

            # Maak een dictionary met de data
            data = {
                "temperature": temperature,
                …
            }
           # Open de WebSocket-verbinding
            async with websockets.connect(SENSOR_WS_URL) as websocket:
                # Start een aparte taak voor pings
                ping_task = asyncio.create_task(ping(websocket))

                # Stuur de sensor data naar de WebSocket
                await websocket.send(json.dumps(data))
                print(f"Sent data: {data}")

                # Wacht een paar seconden voor de volgende meting
                await asyncio.sleep(3)

                # Stop de ping-taak na het verzenden van data
```

```
                ping_task.cancel()
                try:
                    await ping_task
                except asyncio.CancelledError:
                    pass

        except Exception as e:
            print(f"Error: {e}")
            await asyncio.sleep(5)  # Even wachten voordat je het opnieuw
probeert
```

3.3.2.3.6. Uitlezen en verzenden van regenval (reed-switch)

In deze functie wordt gedurende 60 seconden bijgehouden hoe vaak de reed-switch is geactiveerd. Dit wordt omgerekend naar de hoeveelheid neerslag in millimeter en via WebSocket verzonden:

```
async def send_regen_data():
    global running
    try:
        while running:
            # Start een minuut lang tellen
            counter = 0
            previous_button_state = GPIO.LOW
            start_time = time.time()

            print("Start met tellen gedurende een minuut...")

            while time.time() - start_time < 60:
                button_state = GPIO.input(button_pin)

                if button_state == GPIO.HIGH and previous_button_state ==
GPIO.LOW:
                    counter += 1

                previous_button_state = button_state
                await asyncio.sleep(0.1)  # Gebruik asyncio.sleep in plaats van
time.sleep

            # Bereken regenval
            regenval = counter * (100 / calibratie_factor)
            tijdstip = time.strftime("%Y-%m-%d %H:%M:%S")

            # Verstuur data via WebSocket
            data = {
                "regenval": round(regenval, 2),
                "tijdstip": tijdstip
            }

            print(f"Tijdstip: {tijdstip}, Totaal aantal drukken: {counter},
Regenval: {regenval:.2f} mm")
```

```
            await send_to_websocket(data)

    except KeyboardInterrupt:
        stop_loop()
    except Exception as e:
        print(f"Er trad een fout op: {e}")
        stop_loop()
```

De data wordt verstuurd via deze aparte functie:

```
async def send_to_websocket(data):
    try:
        async with websockets.connect(REGEN_WS_URL) as websocket:
            await websocket.send(json.dumps(data))
            print(f"Verstuurd naar RegenSensorUpload WebSocket: {data}")
    except Exception as e:
        print(f"Fout bij het verbinden met de WebSocket: {e}")
```

3.3.2.3.7. Starten en stoppen van het programma

Tot slot zorgen deze functies ervoor dat het programma correct gestart en gestopt wordt:

```
# Stop de regensensor lus
def stop_loop():
    global running
    running = False
    GPIO.cleanup()
    print("De regensensor lus is gestopt en GPIO is opgeruimd.")

# Start beide taken in afzonderlijke threads
def start_threads():
    sensor_thread = threading.Thread(target=lambda:
asyncio.run(send_sensor_data()))
    regen_thread = threading.Thread(target=lambda:
asyncio.run(send_regen_data()))
    sensor_thread.start()
    regen_thread.start()
    sensor_thread.join()
    regen_thread.join()

if __name__ == "__main__":
    try:
        start_threads()  # Start beide taken in verschillende threads
    except KeyboardInterrupt:
        stop_loop()
```

3.3.3. Databeheer

3.3.3.1. Inleiding SQLite

Figuur 16 SQLite logo

Voor het opslaan en beheren van gegevens wordt in dit project gebruikgemaakt van SQLite, een lichte, ingebouwde relationele database die standaard ondersteund wordt door Django. SQLite is bijzonder geschikt voor kleinere toepassingen zoals embedded systemen of prototypes, omdat het geen aparte database-server vereist. De volledige database is namelijk opgeslagen in één enkel bestand op het systeem, wat het beheer eenvoudig maakt en installatieproblemen voorkomt. Django werkt zeer vlot samen met SQLite dankzij het ingebouwde ORM (Object-Relational Mapping)-systeem. In plaats van rechtstreeks met complexe SQL-queries te werken, laat het ORM toe om gegevens via Python-objecten aan te maken, op te vragen, bij te werken en te verwijderen. Dit betekent dat modellen in de Django-code automatisch vertaald worden naar tabellen in de database, en dat data op een overzichtelijke en veilige manier beheerd kan worden.
Een bijkomend voordeel van SQLite is dat het perfect werkt zonder extra configuratie. Tijdens de ontwikkeling op de Raspberry Pi was het daarom mogelijk om snel en zonder externe afhankelijkheden een volledig werkende databaseomgeving op te zetten. Mocht de toepassing in de toekomst uitgebreid worden of in productie gaan, dan kan de SQLite-database eenvoudig vervangen worden door een krachtigere oplossing zoals PostgreSQL of MySQL zonder dat de logica in de Django-code aangepast hoeft te worden.
Door de combinatie van Django en SQLite was het mogelijk om het databeheer efficiënt, eenvoudig en betrouwbaar te realiseren, zonder onnodige complexiteit toe te voegen aan het systeem.

3.3.3.2. Opbouw database

Er zijn 2 tabellen gemaakt in het project. 1 hiervan is voor het opslaan van weerdata, de andere voor alle andere data. De 2 tabellen zijn gescheiden omdat ze op een ander tijdsinterval worden aangevuld. De regenmetingen worden om de minuut doorgegeven waarbij de temperatuur, humidity,… worden geüpdate om de paar seconden.

SELECT * FROM weerstation_regenmeting — Schema — Query Editor

	id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT	regenval REAL NOT NULL	tijdstip datetime NOT NULL
1	1	1.5151515151515151	2025-02-02 10:52:32…
2	2	13.636363636363637	2025-02-02 10:53:32…
3	3	6.060606060606606	2025-02-02 10:54:32…
4	4	0.0	2025-02-02 10:55:32…
5	5	0.0	2025-02-02 10:56:32…

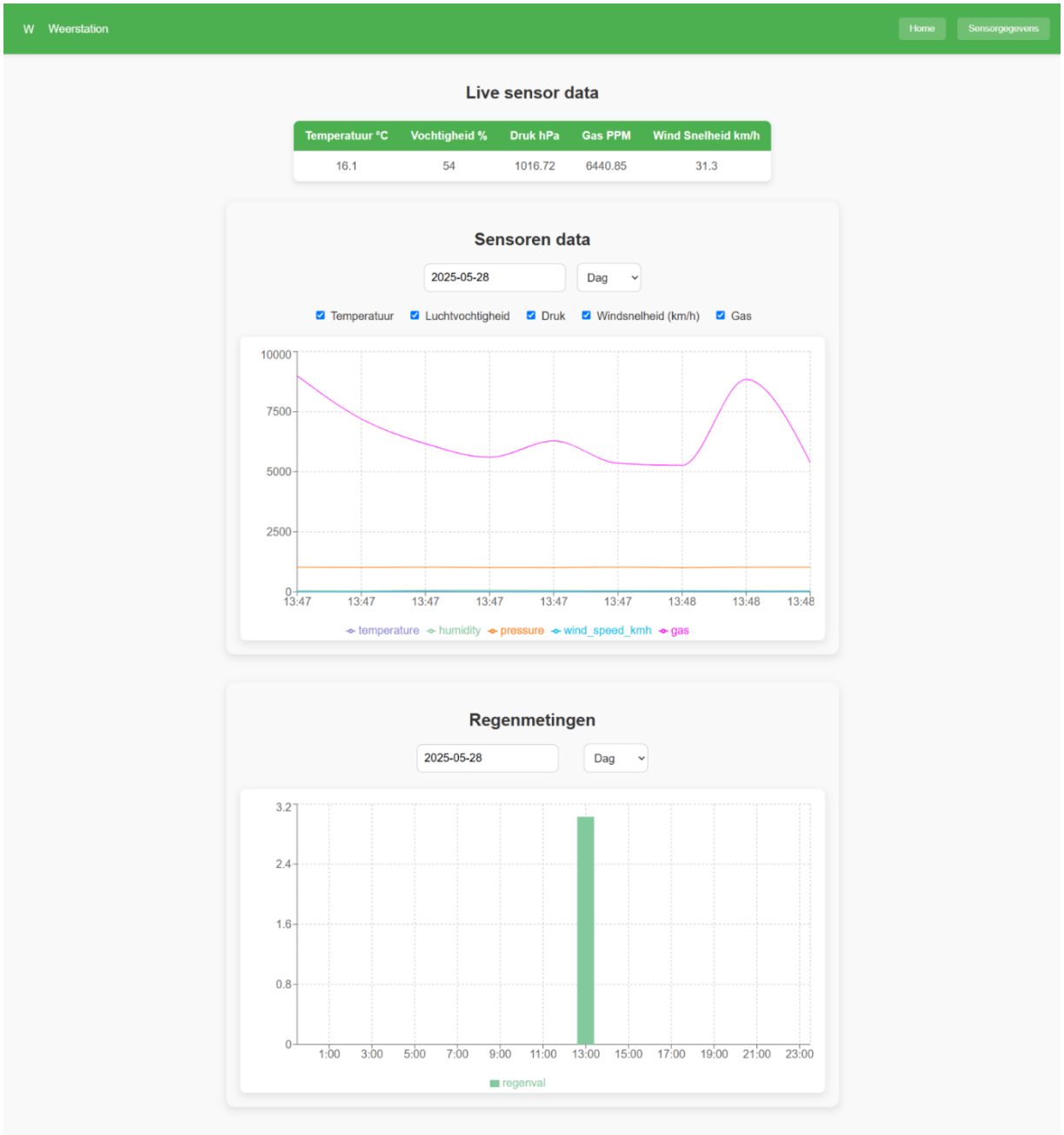
Figuur 17 tabel regenmeting

SELECT * FROM weerstation_sensordata — Schema — Query Editor — Auto Reload — Find — Other Tools…

	id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT	temperature REAL NOT NULL	humidity REAL NOT NULL	pressure REAL NOT NULL	gas REAL NOT NULL	wind_speed_kmh REAL NOT NULL	timestamp datetime
1	1	20.4659375	48.08299686360526	1018.2117573111083	242614.0	0.4838671590319529	2025-02-08 10:33:55…
2	2	20.4778515625	47.97134526766588	1018.2057232655496	62350.0	0.47475438093203604	2025-02-08 10:33:57…
3	3	20.50578125	47.837889314425794	1018.2057958509253	85495.0	0.4838671590319529	2025-02-08 10:33:59…
4	4	20.5411328125	47.67570784131473	1018.2027031335457	102018.0	0.4474160466322816	2025-02-08 10:34:01…
5	5	20.57765625	47.5316079206316	1018.2033228606606	116108.0	0.4838671590319529	2025-02-08 10:34:04…

Figuur 18 tabel sensordata

4 Eindresultaat website



Dit is het eindresultaat van de website. Bovenaan wordt de live data weergegeven, gevolgd door een grafiek met de geschiedenis van temperatuur, luchtvochtigheid, atmosferische druk, windsnelheid en gaswaarden. Onder deze grafiek staat een aparte weergave van de gemeten regenval.

Beide grafieken zijn interactief, je kunt doorheen verschillende datums navigeren om historische gegevens te bekijken. In de bovenste grafiek is het bovendien mogelijk om te selecteren welke sensordata zichtbaar moet zijn.

5 Besluit

Dit project is goed verlopen, al was er wel verwacht toch nog een windrichtingsensor te kunnen toevoegen aan het geheel. Ook het integreren van de data in homeassistent is nog niet gelukt. Dit zijn dingen die er later nog zouden kunnen aan toegevoegd worden om dit project volledig af te werken.

Moeilijke woorden

Home Assistant – Een open-source platform om slimme apparaten in huis te automatiseren en centraal aan te sturen.
Raspberry Pi – Een kleine, goedkope computer die vaak wordt gebruikt voor hobbyprojecten, zoals domotica of programmeren.
Webinterface – Een gebruikersinterface die via een webbrowser toegankelijk is, vaak gebruikt om instellingen te beheren of gegevens te bekijken.
Python programma – Een stukje software geschreven in de programmeertaal Python, vaak gebruikt vanwege de eenvoud en leesbaarheid.
Web socket – Een technologie waarmee een permanente verbinding tussen een server en een client mogelijk is voor realtime communicatie.
SQLite-database – Een lichte, ingebouwde database die vaak lokaal wordt gebruikt om gegevens op te slaan zonder dat er een aparte server nodig is.
Django-framework – Een populair webframework in Python waarmee je snel veilige en schaalbare webapplicaties kunt bouwen.
API (Application Programming Interface) – Een set regels waarmee verschillende softwaretoepassingen met elkaar kunnen communiceren.
Shunt resistor – Een weerstand die wordt gebruikt om een kleine spanning te meten die overeenkomt met een elektrische stroom.
ABS (Acrylonitrile Butadiene Styrene) – Een sterke, harde kunststof die vaak wordt gebruikt voor behuizingen of onderdelen in elektronica.
Reedswitch – Een schakelaar die reageert op een magneet; vaak gebruikt om beweging of positie te detecteren.
GPIO (General Purpose Input/Output) – Pinnen op een Raspberry Pi die gebruikt kunnen worden om elektronische signalen te ontvangen of te verzenden.
Wartel – Een schroefbaar doorvoerstuk dat kabels stevig vastzet en beschermt bij het binnenkomen in een kast of behuizing.

Bijlagen

Bijlage 1: Kriwan int 10 anemometer
Bijlage 2: Bosch bme680
Bijlage 3: Ads1115 texas instrumetns
Bijlage 4: Raspbarry pi 4
Bijlage 5: Phoenix contacts voeding
Bijlage 6: Broncode https://github.com/jasper-lanoote/eindwerk

Bijlages beschikbaar in github repository: https://github.com/jasper-lanoote/eindwerk-verslagen-en-bijlages/tree/main/bijlages

Literatuurlijst

(apa7 -> https://www.scribbr.nl/bronvermelding/generator/apa/)

- Thingiverse.com, & Explaining Computers. (2024, 25 mei). *Raspberry Pi Pico Rain Gauge by ExplainingComputers*. Thingiverse. https://www.thingiverse.com/thing:6636418
- ExplainingComputers. (2021, 3 januari). *Raspberry Pi Weather Station* [Video]. YouTube. https://www.youtube.com/watch?v=ChQpD2gsC20
- James Clough. (2016, 14 augustus). *Solar Radiation Shield (Stevenson Screen) for Outdoor Thermometer by clough42*. Thingiverse. https://www.thingiverse.com/thing:1718334
- Volos Projects. (2022, 12 april). *ESP32 and ADS1115 ADC- Voltmeter project* [Video]. YouTube. https://www.youtube.com/watch?v=u-1TRpLGH04
- *Raspberry Pi: BME280 Temperature, Humidity and Pressure Sensor (Python)*. (2023). Random Nerd Tutorial. https://randomnerdtutorials.com/raspberry-pi-bme280-python/
- *W3Schools.com*. (z.d.). https://www.w3schools.com/
- *Stack overflow*. (z.d.). https://stackoverflow.com/questions
- *GitHub · Build and ship software on a single, collaborative platform*. (2025). GitHub. https://github.com/