Eindwerk Django weerstation

Afbeelding met buitenshuis, hemel, grond, plant

Door AI gegenereerde inhoud is mogelijk onjuist.

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 3](#_Toc199067728)

[Woord vooraf 4](#_Toc199067729)

[1 Inleiding 4](#_Toc199067730)

[1.1. Probleemoriëntering 4](#_Toc199067731)

[1.1.1. De onderzoeksvraag en deelvragen 4](#_Toc199067732)

[1.2. Opzet van dit project 4](#_Toc199067733)

[1.3. Onderzoeksopzet 5](#_Toc199067734)

[2 Onderzoeksfase 6](#_Toc199067735)

[2.1. Literatuurstudie 6](#_Toc199067736)

[2.2. Belangrijke ontdekkingen en technieken 6](#_Toc199067737)

[3 Ontwerpfase 7](#_Toc199067738)

[3.1. Hardware 7](#_Toc199067739)

[3.1.1. Mechanisch 7](#_Toc199067740)

[3.1.2. Elektronisch 8](#_Toc199067741)

[3.2. Software 10](#_Toc199067742)

[3.2.1. Front-end 10](#_Toc199067743)

[3.2.2. Backend 11](#_Toc199067744)

[3.2.3. Databeheer 17](#_Toc199067745)

[3.2.4. Schema volledige plaatje 17](#_Toc199067746)

[4 bijlagen 17](#_Toc199067747)

[Literatuurlijst 21](#_Toc199067748)

[4.1. bronvermeldingen onderzoeksfase 21](#_Toc199067749)

# Figuurlijst

[Figuur 1 geheel project 7](#_Toc199067712)

[Figuur 2 Stevenson screen sensor bescherming 7](#_Toc199067713)

[Figuur 3 geheel van de pluviometer 7](#_Toc199067714)

[Figuur 4 tipping bucket mechanisme 7](#_Toc199067715)

[Figuur 5 Kriwan INT 10 anemometer 8](#_Toc199067716)

[Figuur 6 elektrisch schema 8](#_Toc199067717)

[Figuur 7 Kriwan INT 10 anemometer 8](#_Toc199067718)

[Figuur 8 ADS1115 analog-digital converter 8](#_Toc199067719)

[Figuur 9 Bosch bme680 8](#_Toc199067720)

[Figuur 10 React logo 10](#_Toc199067721)

[Figuur 11 Python logo 11](#_Toc199067722)

[Figuur 12 Django logo 11](#_Toc199067723)

[Figuur 13 SQLite logo 17](#_Toc199067724)

# Woord vooraf

# Inleiding

## 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, kun je zelf kiezen welke sensoren je aansluit en het systeem eenvoudig uitbreiden zodra er nieuwe sensoren beschikbaar komen. Dit biedt een flexibele en schaalbare oplossing die perfect aansluit bij de specifieke behoeften van je woning.

### De onderzoeksvraag en deelvragen

#### onderzoeksvraag

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

#### 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, 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.)

## 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, CO₂, UV en O₃ , om ook luchtkwaliteit te monitoren.

Daarnaast zal de verzamelde data via een webpagina gevisualiseerd kunnen 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.

## 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 verdere uitwerken van dit project.
2. **Kiezen van benodigde sensoren**Na het vaststellen van welke gegevens we zullen gebruiken, 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.

# Onderzoeksfase

## Literatuurstudie

**Regenmeter - Tipping Bucket Mechanisme:** Uit de informatie van de video van ExplainingComputers heb ik het *tipping bucket* mechanisme leren kennen 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 ik de BME680 ga gebruiken, 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 voor mijn project. De nauwkeurigheid van de sensor is essentieel voor betrouwbare metingen en de video bevestigde dat het buitengebruik mogelijk is.

**Windsnelheid - Analoge Windsnelheidsmeter:** Ik kreeg een windsnelheidsmeter van mijn vader, 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 word.

**Temperatuur- en Vochtigheidssensor Bescherming - Stevenson Screen:** Een belangrijk aspect van het project was de bescherming van de temperatuursensor tegen direct zonlicht en regen. Door onderzoek ontdekte ik 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.

## 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.

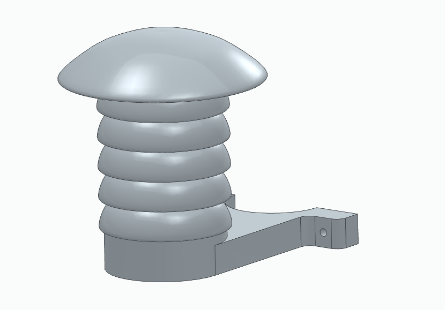
# Ontwerpfase

## Hardware

### 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.

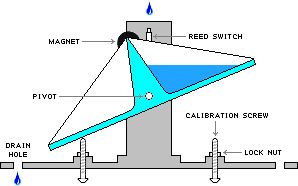
Afbeelding met buitenshuis, hemel, grond, plant

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 1 geheel project

Figuur 2 Stevenson screen sensor bescherming

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.

Afbeelding met keukenaccessoires, koffiemolen, ontwerp

Door AI gegenereerde inhoud is mogelijk onjuist.

Figuur 3 geheel van de pluviometer

Figuur 4 tipping bucket mechanisme

Afbeelding met cilinder

Door AI gegenereerde inhoud is mogelijk onjuist.

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 5 Kriwan INT 10 anemometer

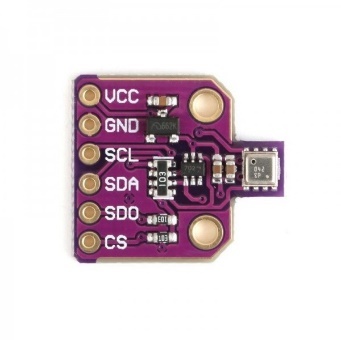
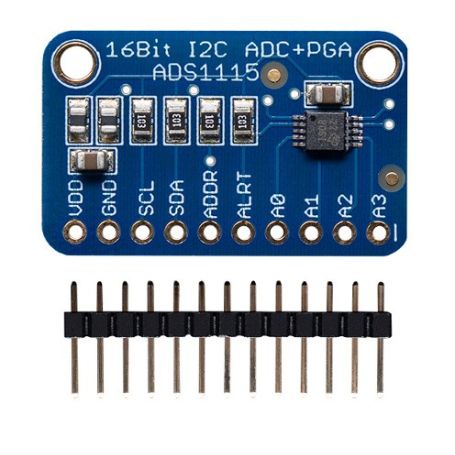
De **elektronica** is centraal samengehouden in een **aftakdoos**, waarin alle verbindingen samenkomen. De kabels worden via de onderzijde van de doos binnengeleid door middel van wartels, die zorgen voor een waterdichte en stevige doorvoer. Dit maakt het geheel geschikt voor gebruik in een buitenomgeving.

### Elektronisch

Afbeelding met tekst, diagram, Plan, schermopname

Door AI gegenereerde inhoud is mogelijk onjuist.Voor de meting van verschillende omgevingswaarden wordt gebruikgemaakt van een aantal sensoren die in verbinding staan met een Raspberry Pi. 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.

Figuur 6 elektrisch schema

In het omhulsel volgens het Stevenson screen-principe bevindt zich 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.

Afbeelding met cilinder

Door AI gegenereerde inhoud is mogelijk onjuist.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 7 Kriwan INT 10 anemometer

Figuur 8 ADS1115 analog-digital converter

Figuur 9 Bosch bme680

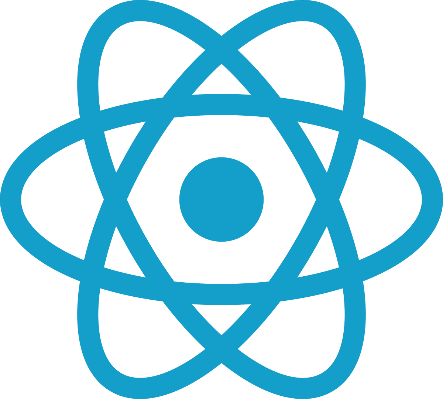
De windsnelheid wordt gemeten met een Kriwan INT10 anemometer, een robuuste industriële sensor die gevoed wordt met 24 volt. De negatieve uitgang van deze sensor is via een shuntweerstand van 235 ohm verbonden met de massa. Een shuntweerstand is een precieze, lage weerstand die wordt gebruikt om de stroom 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.

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.

## Software

### Front-end

#### Inleiding React

Voor de ontwikkeling van de frontend wordt gekozen voor React, een populaire JavaScript-bibliotheek die gebruikt wordt om interactieve gebruikersinterfaces te bouwen. React is ontwikkeld door Meta (voorheen 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.

Figuur 10 React logo

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.

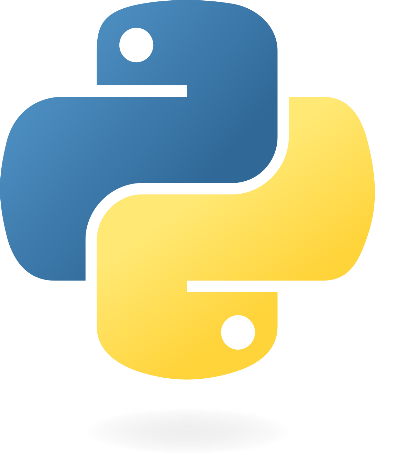
In dit project wordt React gebruikt om de data die wordt doorgestuurd door de Raspberry Pi op een overzichtelijke en interactieve manier te tonen aan de gebruiker. Denk hierbij aan real-time grafieken van temperatuur, luchtvochtigheid en windsnelheid, visuele weergave van de hoeveelheid neerslag, en 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.

#### Enkele belangrijke stukjes code

### Backend

#### 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.

Figuur 11 Python logo

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 12 Django 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 werd 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.

#### Enkele belangrijke stukjes code

##### Het uitlezen van de sensoren

##### Dit zijn de bibliotheken die ik gebruik

import asyncio

import json

import time

import websockets

import RPi.GPIO as GPIO

import threading

import board

import adafruit\_bme680

import Adafruit\_ADS1x15

**Beschrijving van de 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.

##### 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/"

##### Verbinding actief houden

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

async def ping(websocket):

    try:

        while True:

            await websocket.ping()

            await asyncio.sleep(30)

    except Exception as e:

        print(f"Ping error: {e}")

##### 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)

            humidity = round(sensor.humidity)

            pressure = round(sensor.pressure, 2)

            gas = round(sensor.gas, 2)

            # 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,

                "humidity": humidity,

                "pressure": pressure,

                "gas": gas,

                "wind\_speed\_kmh": windsnelheid\_kmph

            }

            # 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

##### 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}")

##### 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()

### Databeheer

#### Inleiding SQLite

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.

Figuur 13 SQLite logo

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.

#### 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 geüpdatete om de paar seconden.

Afbeelding met tekst, schermopname, Lettertype, software

Door AI gegenereerde inhoud is mogelijk onjuist.

Afbeelding met tekst, schermopname, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

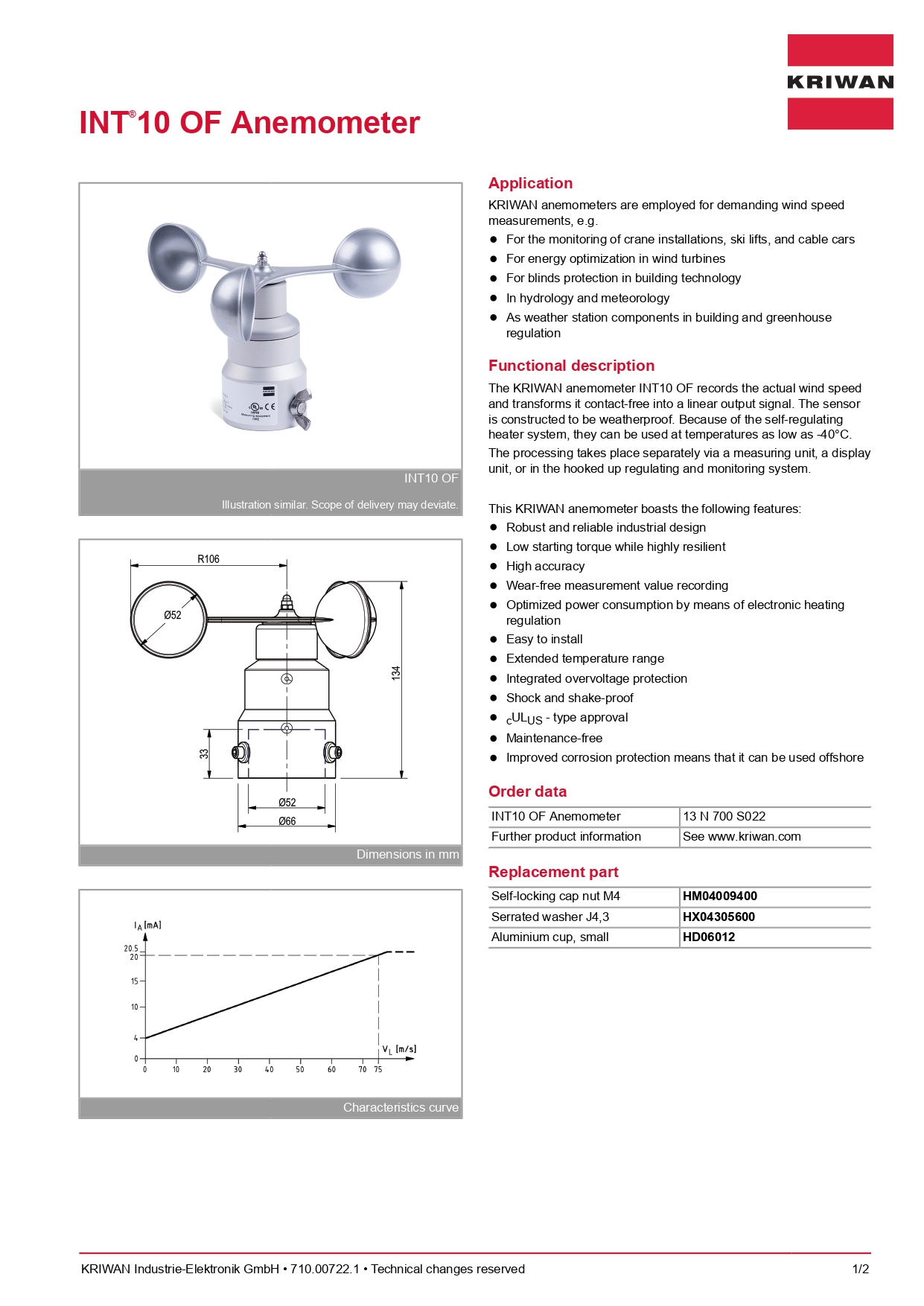
### Schema volledige plaatje

# Afbeelding met diagram, Plan, Technische tekening, schematisch Door AI gegenereerde inhoud is mogelijk onjuist.

Op dit schema kun je goed zien hoe alle componenten met elkaar in verbinding staan. Je ziet dat alle sensoren worden uitgelezen met een Raspberry Pi, via het script alles\_uitlezen.py. Dit script stuurt alle data via twee websockets naar Django. In Django wordt de data vervolgens opgeslagen in de bijbehorende databases. Alleen de websocket /ws/upload/ is direct gekoppeld aan /ws/livesensordata/, zodat de data live doorgestuurd kan worden.

De data in de database wordt opgevraagd via twee API-endpoints (HTTP GET): api/regenmetingen en api/sensordata. Deze gegevens worden vervolgens doorgestuurd naar de bijbehorende React-componenten. De React-frontend is opgebouwd met behulp van routing. Via deze routing kun je navigeren tussen de pagina’s livedata en home.

# bijlagen

Bijlage 1

Afbeelding met tekst, schermopname, document, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

Bijlage 2

Bosch bme680

Bijlage 3

Ads1115 texas instrumetns

Bijlage 4

Raspbarry pi 4

Bijlage 5

Phoenix contacts voeding

# Literatuurlijst

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

* <https://www.thingiverse.com/thing:6636418> (rain gauge bestanden en uitleg)
* <https://randomnerdtutorials.com/raspberry-pi-bme280-python/> (bosch bme sensor aansluiten)
* <https://sensirion.github.io/python-i2c-scd30/installation.html> (code co2 sensor van vulstke)
* <https://www.thingiverse.com/thing:1718334> stevenson screen
* <https://www.youtube.com/watch?v=ChQpD2gsC20&ab_channel=ExplainingComputer> boch bme 280
* <https://www.youtube.com/watch?v=u-1TRpLGH04&t=22s&ab_channel=VolosProjects> ads 1115 sensor voor analoge signalen

## bronvermeldingen onderzoeksfase

* 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>