Het overbrengen en visualiseren van real-time sensordata tijdens een brandweerinzet

Afstudeeropdracht HBO-ICT

Versie 2.3

Naam:	R. Holtman
Student nummer:	465101
Opleiding:	HBO-ICT bij hogeschool Saxion te Enschede
Module:	Afstuderen
Docent:	R.M.P.L.C. van den Nieuwenhoff
Opdrachtgever:	Veiligheidsregio Noord- en Oost-Gelderland

Voorwoord

Voor u ligt de scriptie ‘Het overbrengen en visualiseren van real-time sensordata tijdens een brandweerinzet’. Het onderzoek voor deze scriptie is uitgevoerd bij de afdeling brandweer van de veiligheidsregio Noord- en Oost-Gelderland (VNOG). Deze scriptie is geschreven in het kader van mijn afstuderen van de bachelor HBO-ICT bij hogeschool Saxion te Enschede. Van april 2022 tot en met oktober 2022 ben ik bezig geweest met het onderzoek, het realiseren van het proof of concept en het schrijven van deze scriptie.

Tijdens dit onderzoek stonden mijn begeleiders Adriaan Westland (Senior Vakspecialist Operationele Voorbereiding), Carlo Balistreri (Beheerder Operationele Informatiesystemen), Eric Klein Goldewijk (Specialist BI en Datawarehouse) en René van den Nieuwenhoff, de begeleider vanuit de opleiding, altijd voor mij klaar. Zij hebben tijdens het onderzoek op een professionele en prettige wijze bijgestaan om dit mooie resultaat te kunnen realiseren.

Bij deze wil ik graag mijn begeleiders bedanken voor de fijne begeleiding en hun ondersteuning tijdens dit traject. Tevens wil ik de medewerkers van de Veiligheidsregio graag bedanken voor de fijne samenwerking en goede adviezen. Daarnaast wil ik Kiara Veenendaal en Richard Holtman bedanken voor alle tijd die zij hebben gestoken in het meelezen van mijn scriptie.

Tot slot wil ik Olav Strotmann (Afdelingshoofd brandweerzorg) in het bijzonder bedanken voor de mogelijkheid om dit project bij de veiligheidsregio Noord- en Oost-Gelderland uit te mogen voeren.

Ik wens u veel leesplezier toe.

Rick Holtman

Apeldoorn, 30 Oktober 2022

Versiebeheer

Datum	Versie	Opmerkingen
14-09-2022	V0.1	Initiatie document
07-10-2022	V0.2	Concept voor spellingscheck
09-10-2022	V1.0	Afronding voor inlevering conceptversie
26-10-2022	V1.1	Concept voor feedback
30-10-2022	V2.0	Afronding voor inlevering definitieve versie
30-11-2022	V2.1	Concept voor feedback
12-12-2022	V2.2	Concept voor feedback
16-12-2022	V2.3	Afronding voor inlevering nieuwe definitieve versie

Begrippenlijst

Begrip	Uitleg
VNOG	Veiligheidsregio Noord- en Oost-Gelderland
POC	Proof of concept
BiO	Baseline informatiebeveiliging Overheid
Tankautospuit (TS)	Een type voertuig dat door de brandweer ingezet wordt als basisvoertuig. Het voertuig is zo ingericht dat de eerste slag geslagen kan worden bij brandbestrijding of ongevallen
Node	Een apparaat dat als onafhankelijke eenheid kan worden beschouwd
API	Een geformaliseerde manier om gegevens uit te wisselen tussen applicaties
Client	Een applicatie of een computersysteem met toegang tot een ander systeem, de server, via een netwerk
Scrapen	Computertechniek waarbij software wordt gebruikt om informatie van webpagina's te extraheren en te analyseren
Framework	Een platform waarmee softwareapplicaties gebouwd kunnen worden gebaseerd op vaste regels en elementen
Record	Een gestructureerde gegevenseenheid in een tabel
Dashboard	Applicatie bedoelt voor het weergeven van informatie via widgets of gadgets

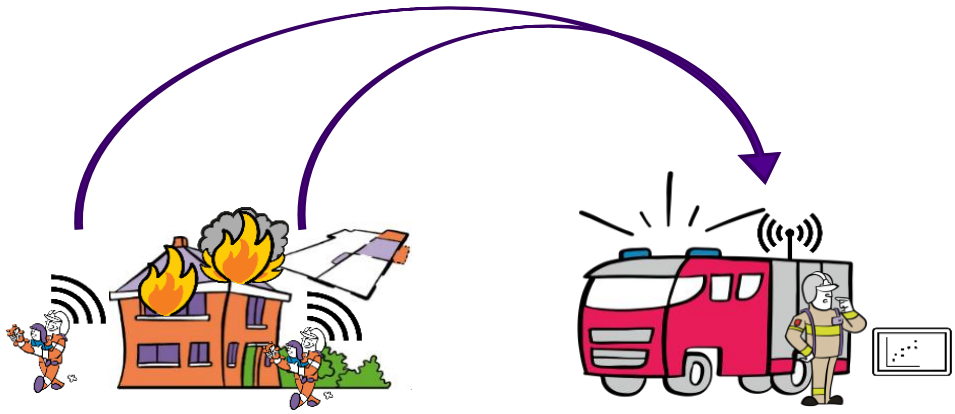
Inhoudsopgave

1 Inleiding

De brandweer is onderdeel van de veiligheidsregio en heeft als wettelijke taak om tijdig en adequaat te kunnen optreden als gevolg van een incident. Eén van deze incidenten is brandbestrijding. De inzet en het grillige en vaak onvoorspelbare verloop van een brandbestrijding is voor het brandweerpersoneel niet zonder risico's. Om deze risico's te kunnen verkleinen is de brandweer continu opzoek naar oplossingen die hieraan bijdragen. In dit kader wil de brandweer graag onderzoeken of het beoordelen van real-time sensor data kan bijdragen aan een veilig werkomgeving tijdens een brandweerinzet.

Tijdens dit afstudeerproject is er een proof of concept ontwikkeld, welke het voor de brandweer mogelijk maakt om hiermee te experimenteren. Dit kan met verschillende soorten en type sensoren welke op een eenvoudig manier real-time kunnen worden uitgelezen en op een voor de brandweer werkbare manier kunnen worden gevisualiseerd.

Het systeem waar onderzoek na is gedaan, geeft de VNOG de mogelijkheid om te kunnen experimenteren met verschillende sensoren en de impact van de sensordata te kunnen beoordelen.



Figuur 1. Globaal overzicht van het ontwikkelde systeem

2 Leeswijzer

In dit document is het hele proces beschreven van hoe dit project verlopen is en de uitkomsten hiervan De onderdelen die beschreven worden in dit document zijn als volgt:

- De context van de opdracht en de daarbij horende uitdaging;
- Welke proces methodiek is gebruikt en de toelichting hiervan;
- Het beoogde onderzoek en de hierbij horende hoofd- en deelvragen;
- De fase waarin er een zo goed mogelijk beeld wordt gevormd van de huidige situatie en zo ook de behoefte van de gebruikers en het systeem;
- De fase waarin de beoogde oplossing en dus het functionele ontwerp wordt gedefinieerd;
- De fase waar er door onderzoek te doen een oplossingsrichting wordt gekozen;
- Het technisch ontwerp voor het proof of concept op basis van de oplossingsrichting;
- Hoe het proof of concept is gerealiseerd en de uiteenzetting van de daarbij gemaakte keuzes;
- Hoe het proof of concept op verschillende manieren is getest en de resultaten hiervan;
- De conclusie op de hoofdvraag, aanbevelingen voor een volgende iteratie en aandachtspunten van de oplossing.

3 De organisatie

De veiligheidsregio Noord- en Oost-Gelderland is één van de 25 veiligheidsregio’s in Nederland. Iedere veiligheidsregio zet zich in voor de veiligheid van de inwoners en bezoekers van dat gebied. Zo zorgt de veiligheidsregio ervoor dat er een brandweer is. Ook maakt de veiligheidsregio afspraken over de aanpak van rampen en crises. Een goede samenwerking tussen hulpverleningsdiensten, overheden, bedrijven en burgers is daarbij belangrijk. (Veiligheidsregio's en crisisbeheersing, 2022)

Figuur 2. Kaart veiligheidsregio's in Nederland.

De VNOG bestaat uit 22 gemeenten. Het bestuur van de VNOG, het Algemeen Bestuur, bestaat uit de 22 burgemeesters van de gemeenten. In het gebied wonen op een oppervlakte van 3.000km² ruim 870.000 mensen. Binnen deze regio zijn 56 brandweerposten actief. Zij voeren de dagelijkse brandweerzorg uit. De brandweer op regioniveau zorgt onder andere voor de voorbereiding op rampen en calamiteiten, samen met de GHOR (geneeskundige hulpverlening in de regio). Hiervoor werken ze samen met politie en de gemeenten. Er werken zo'n 1.500 brandweervrijwilligers en 300 beroepskrachten in de regio. (Veiligheidsregio Noord- en Oost-Gelderland, 2022)

Het afstudeerproject zal worden gedaan binnen de afdeling brandweerzorg, deze afdeling is verantwoordelijk voor het snel en adequaat leveren van brandweerzorg en de voorbereiding hierop. Het team operationele voorbereiding opereert op het grensvlak tussen voorbereiding en incidentbestrijding, met als hoofddoel een maximale en optimale voorbereiding van de incidentbestrijding afgestemd op de risico’s in de regio.

4 De uitdaging

Binnen de brandweer is er de behoefte om informatie gestuurd te werken. Om de veiligheid van het brandweerpersoneel zo goed mogelijk te kunnen waarborgen, wordt er tijdens een inzet ter plekke een risicoanalyse gemaakt van de huidige situatie, waarop vervolgens de inzet wordt bepaald. Hierbij is het van belang dat er zoveel mogelijk actuele informatie beschikbaar is om hier een betrouwbaar beeld van te vormen. Op dit moment gebeurt dit vooral op basis van observaties en bestaande algemene informatie.

Om in deze informatiebehoefte te voorzien wil de brandweer een platform waarbinnen zij snel kunnen experimenteren met verschillende soorten sensoren. Het snel tussen verschillende soorten sensoren kunnen wisselen stelt de brandweer in staat om te onderzoeken welke sensoren een daadwerkelijk significante impact hebben op de risicoanalyse.

4.1 Opdracht definitie

Om de brandweer in staat te stellen om met real-time sensordata te experimenteren is de volgende opdracht gedefinieerd:

“Het ontwikkelen van een proof of concept welke met een zo simpel mogelijke configuratie alle soorten sensordata real-time kan samenbrengen naar een centraal punt. Dit centrale punt zal deze sensordata vervolgens opslaan en visualiseren, zodat de data gebruikt kan worden voor een risicoanalyse.”

4.2 Afbakening

De opdracht heeft het risico in zich om behoorlijk breed en uitgebreid te benaderen. Daarom is het wenselijk om deze vooraf af te bakenen om tot aan de doelstelling van de opdracht te voldoen.

4.2.1 Hardware

Om het proof of concept af te kaderen is er qua hardware twee significante aspecten om rekening mee te houden, namelijk de sensoren en de omgevingsvariabelen.

Binnen deze opdracht zal er enkel rekening worden gehouden met sensoren welke numerieke waardes genereren. Denk hierbij aan temperatuurmetingen, druk van een ademluchtcilinder, etc. Dit houdt in dat er bijvoorbeeld geen camera's aan het systeem gekoppeld kunnen gaan worden. Welke sensoren nuttige informatie zouden kunnen opleveren voor de brandweer valt buiten het kader van deze opdracht.

Tijdens de selectie van de hardware zal geen rekening worden gehouden met omgevingsvariabelen zoals temperatuur of waterdichtheid. Dit zijn beide aspecten welke afgevangen kunnen worden met een behuizing of iets soortgelijks.

4.2.2 Software

Voor het ontwikkelen van de software dient er rekening te worden gehouden met wetgeving, regulaties en wensen.

Deze opdracht zal omdat de brandweer een overheidsorgaan is, te maken krijgen met verschillende wet- en regelgeving. Hierbij kan worden gedacht aan NEN-normen en de AVG. Omdat het hier gaat om een proof of concept zal dit aspect binnen deze opdracht buiten beschouwing worden gelaten.

Tevens is een wens van de organisatie dat zoveel mogelijk wordt ontwikkeld met openstandaarden. Daarnaast dient er rekening te worden gehouden met de Baseline Informatiebeveiliging Overheid. Hierin staan richtlijnen vanuit de overheid en is verplicht om te gebruiken binnen de organisatie.

5 Het proces

Om de opdracht te kunnen realiseren is het belangrijk om een duidelijk proces te hebben. Voor de realisatie van de opdracht is gekozen om gebruik te maken van verschillende fases welke uiteindelijk tot een proof of concept leidt. Hieronder zal deze methodiek worden toegelicht. Vervolgens zal worden weergegeven welke onderzoeksvragen er worden gesteld.

In het plan van aanpak is gedefinieerd dat er gebruik gemaakt zal worden van verschillende fases. Elk van deze fases heeft een specifiek doel om zo een duidelijke basis te hebben voor de volgende fase. Deze fases zijn als volgt:

Beeldvorming: Tijdens deze fase zal er gekeken worden naar de gebruikers van het systeem en wat zij nodig hebben. Deze fase draagt op deze manier bij aan het vaststellen van de huidige situatie en het opstellen van requirements voor het project.

Mogelijke oplossing: Bij de tweede fase wordt bedacht hoe de uitdaging opgelost gaat worden. Tijdens deze fase worden aan de hand van de huidige en de vorige fase dan ook de requirements vastgesteld.

Oplossingsrichting: Tijdens deze derde fase worden er verschillende mogelijkheden uitgewerkt voor de in de vorige fase bedachte oplossing. Op basis hiervan worden tijdens de volgende fase het technisch ontwerp uitgewerkt en gerealiseerd.

Realisatie: Bij de laatste fase zal er een Technisch ontwerp worden opgesteld en het proof of concept van de oplossing worden gerealiseerd.

5.1 Onderzoek

Om de opdracht succesvol uit te kunnen voeren, zal er een onderzoek moeten worden uitgevoerd om uiteindelijk tot een oplossing voor het probleem van de brandweer te komen. Het doel van het onderzoek zal het bepalen van een oplossingsrichting zijn. Deze oplossingsrichting is de basis voor het ontwerp en realisatie van het proof of concept.

5.1.1 Onderzoeksvragen

Op basis van de opdracht en het beoogde resultaat is de volgende hoofdvraag opgesteld:

> ***Hoe kan de brandweer tijdens een inzet real-time sensordata overbrengen en vervolgens visualiseren zonder afhankelijk te zijn van het soort sensor?***

Om antwoord te krijgen op de hoofdvraag zijn er verschillende aspecten die onderzocht dienen te worden. Deze worden onderzocht middels de volgende vier deel vragen:

1. **Hoe kan de binnenkomende data efficiënt worden verzameld en opgeslagen?**

De focus bij deze deelvraag ligt op de efficiëntie. Het uitgangspunt is dat de sensordata real-time gevisualiseerd wordt. Daarbij is het van belang dat de data op een efficiënte manier kan worden verzameld en opgeslagen, zodat dit niet tot onnodige vertraging binnen het systeem zal leiden.

2. **Hoe kan de binnenkomende data real-time gevisualiseerd worden?**

De relevantie van deze deelvraag zit hem in het feit dat de brandweer zonder visualisatie “niks” aan de sensordata heeft. Omdat de brandweer real time sensordata wil kunnen gebruiken, zal hierop ook de focus liggen. Onder real time visualiseren wordt verstaan het tonen van het verloop van de sensordata. Hierdoor kan de gebruiker eenvoudig zien als hier omstandigheden veranderen.

3. **Wat zijn de requirements voor de draadloze communicatie en welke technieken voldoen hieraan?**

Om de data te kunnen versturen binnen het netwerk is er een draadloze communicatietechniek nodig. De focus zal bij het onderzoek op deze deelvraag liggen, welke technieken er beschikbaar zijn, welke technieken aan de requirements voldoen en wat deze (niet functionele) requirements zijn.

4. **Welke hardware voldoet voor het proof of concept?**

Tenslotte dient er hardware geselecteerd te worden om het proof of concept te realiseren en te testen. Om deze reden zal er ook onderzoek moeten worden gedaan naar welke hardware hier geschikt is.

Wanneer alle deelvragen zijn beantwoord, kan er een technisch ontwerp worden opgesteld. Deze wordt gebruikt om het proof of concept te realiseren. Het proof of concept dient als bewijs dat de hoofdvraag is beantwoord.

5.1.2 Onderzoeksmethode

In onderstaande tabel per deelvraag de onderzoeksmethode weergegeven.

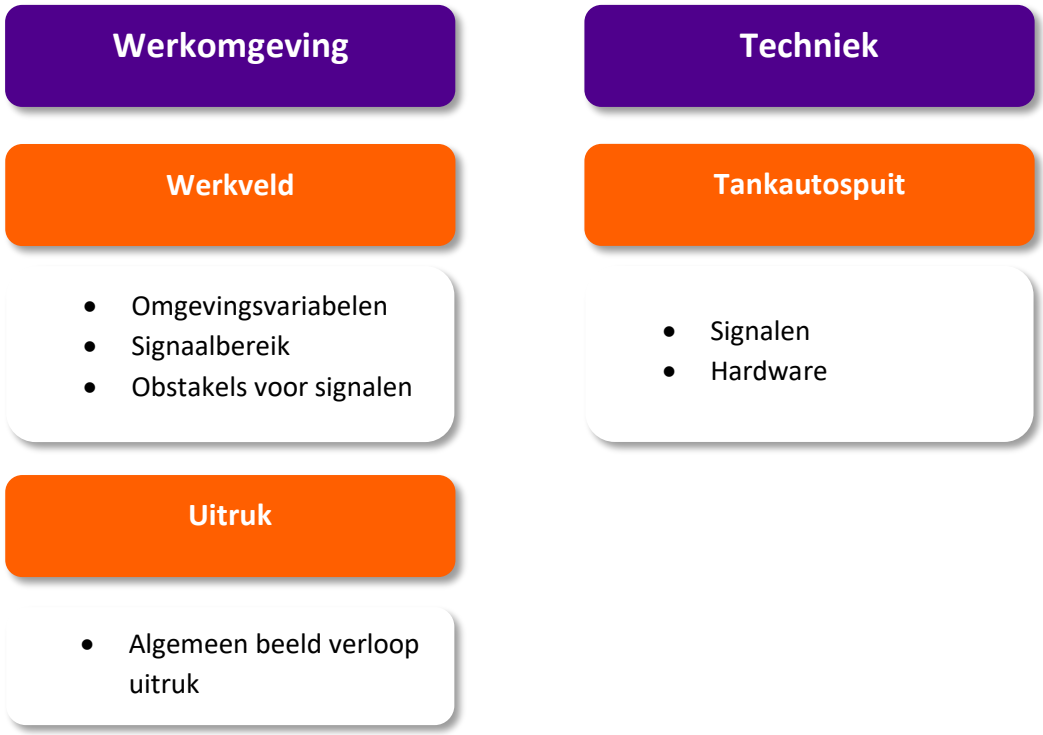
Vraag	Onderzoeksmethode	Beroepsproduct
1	Door literatuuronderzoek te doen naar de eigenschappen van sensordata zal uitgezocht worden welke technieken toegepast kunnen worden voor het proof of concept.	Technisch ontwerp Prototype
2	Door literatuuronderzoek naar de verschillende beschikbare mogelijkheden voor het visualiseren van data zal er worden bepaald welke techniek hiervoor gebruikt kan worden.	Technisch ontwerp Prototype
3	Er zullen interviews met betrokkenen worden afgenomen om een beeld te schetsen van in welke situatie het systeem gebruikt zal gaan worden. Ook zal er literatuuronderzoek gedaan worden naar welke technieken hier geschikt voor zijn.	Requirements Prototype
4	Door literatuuronderzoek te doen naar geschikte hardware wordt het mogelijk om een proof of concept te ontwikkelen welke gebruikt kan worden om het technisch ontwerp en dus de oplossingsrichting te bewijzen.	Technisch ontwerp Prototype
Hoofdvraag	Door middel van de beantwoorde deelvragen zal de hoofdvraag worden uitgewerkt.	

6 Beeldvorming

In het vorige hoofdstuk is besproken welke methodiek zal worden gebruikt tijdens deze opdracht. De eerste fase van deze methodiek is de beeldvormingsfase, in deze fase wordt er getracht een zo goed mogelijk beeld te vormen van de behoefte van de gebruikers en het systeem. Het doel van deze fase is het duidelijk definiëren van de huidige situatie.

6.1 Onderzoeksopzet

De opzet is het onderzoeken van verschillende aspecten binnen de brandweer. Met name de situatie in de werkomgeving en de technische situatie op de tankautospuit. Om een duidelijk beeld te schetsen van de te onderzoeken aspecten is onderstaand figuur opgesteld.



Bij het onderzoek naar de werkomgeving zal de focus liggen op een overzicht en in welke omgeving het beoogde systeem gebruikt zal gaan worden. Daarnaast zal er ook nog een gedeeltelijke beeldvorming van het uitrukproces beschreven worden. Hiermee wordt verwacht inzicht te creëren van het uitrukproces zodat hier, waar mogelijk, rekening mee kan worden gehouden.

Bij de techniek zal de focus liggen op de beeldvorming van de bestaande signalen en hardware op de tankautospuit. Door hier onderzoek na te doen kan dit tijdens het definiëren van de oplossing mee worden genomen.

6.2 Werkomgeving

Om een duidelijk beeld te vormen van waar de oplossing gebruikt moet gaan worden, is hier onderzoek naar gedaan. Dit onderzoek is gedaan door interviews af te nemen met verschillende mensen, die werkzaam zijn in het werkveld. De resultaten van dit onderzoek staan hieronder uitgewerkt.

6.2.1 Werkveld

Voor het onderzoek met betrekking tot het werkveld heb ik met twee ervaringsdeskundige, Adriaan Westland en Richard Holtman interviews afgenomen. In deze interviews is de reden van dit interview uitgelegd en vervolgens gevraagd hoe een brandweer inzet in zijn werk gaat en hierop vervolgvragen gesteld om zo verschillende aspecten verder uit te lichten. Uit deze interviews zijn de volgende resultaten naar voren gekomen.

6.2.1.1 Omgevingsvariabelen

De werkomstandigheden waarin de apparatuur zal moeten functioneren is in een extreme omgeving. Zo zou de omgevingstemperatuur boven de 600°C kunnen uitkomen, zit de lucht vol met roet en moet de apparatuur robuust zijn opgebouwd om tegen val- en slagschade bestand te zijn. Daarnaast wordt de uitrusting na een inzet telkens zorgvuldig schoongemaakt (Holtman, 2022).

6.2.1.2 Signaalbereik

De brandweer voert haar werkzaamheden uit in alle soorten en type gebouwen. Hiervoor is het van belang dat het werkbereik afdoende is voor het functioneren van het systeem.

De volgende twee aspecten hebben invloed op het werkbereik;

1. Afstand;
2. Bebouwing en objecten.

Afstand

Uit de interviews is gebleken dat de afstand die overbrugd zal moeten worden de volgende afstanden zijn

1. Grotendeels binnen 55 meter, de lengte van de hogedruk slang;
2. Vrijwel nooit meer als 200 meter.

Uit deze interviews kan dus geconcludeerd worden dat een maximaal bereik van 200 meter afdoende is, het gemiddelde bereik zal ergens onder de 55 meter liggen, dit is lengte van de hogedruk slang welke voornamelijk wordt gebruikt tijdens een inzet.

Bebouwing en objecten

Echter dient er hierbij ook rekening te worden gehouden met de bebouwing. Bebouwing kan het bereik van het systeem sterk beïnvloeden, zeker bij bijvoorbeeld gebouwen zoals een ziekenhuis welke vaak voornamelijk uit beton bestaan. Zo heeft bijvoorbeeld het Gelre ziekenhuis in Apeldoorn versterkers voor het communicatiesysteem van de hulpdiensten, het zogenaamde C2000 systeem. (Holtman, 2022)

6.2.1.3 Uitruk

Om beeld te krijgen wat het werkproces is van de brandweer vanaf het moment van alarmering tot aan het aankomen op het incidentadres, is er een interview met een deskundige afgenomen. De hoofdonderdelen zijn in het onderstaande schema weergegeven. Voor dit werkproces is in de praktijk een 5 tot 8 minuten gemiddeld beschikbaar om gereed te zijn met alle uitrusting. Dit is met name afhankelijk van de rijtijd van de kazerne naar het incidentadres (Holtman, 2022).

Zoals hierboven is te zien moet er een aantal handelingen worden verricht binnen een kort tijdsbestek. Een opmerking die hierbij werd gemaakt is dat het zelf toevoegen, aan moeten zetten of mee moeten nemen van een dergelijk systeem geen reële optie is, omdat dit arbeidsintensief kan zijn. Hieruit kan geconcludeerd worden dat het systeem ergens geïntegreerd zal moeten worden in iets wat de gebruikers altijd al standaard bij zich dragen. Hierbij kan bijvoorbeeld worden gedacht aan het ademluchttoestel (Holtman, 2022). Een ademluchttoestel wordt tijdens het aanrijden omgehangen op de rug en heeft ruimte om hierop extra onderdelen bij op te hangen. Zie onderstaande afbeelding:

Figuur 3. Ademluchttoestellen in gebruik door de brandweer.

6.3 Techniek

Voor het onderzoek naar het technische aspect zijn er tevens interviews afgenomen met verschillende professionals met uitgebreide kennis over de technische details binnen de brandweer(voertuigen). Deze interviews zijn afgenomen bij Carlo Balistreri, Eric Klein Goldewijk en Martijn Veurink. Ook hier heb ik eerst het doel van de interviews uitgelegd en daarna gevraagd naar de verschillende systemen. Hierop zijn vervolgvragen gesteld om meer informatie te krijgen over verschillende aspecten.

6.3.1 Hardware en signalen

Om een duidelijk beeld te krijgen van de bestaande hardware die aanwezig is bij een inzet is onderzoek gedaan naar deze systemen. Door dit onderzoek kan rekening gehouden worden met voornamelijk de signalen die zich om het voertuig bevinden, zodat hierin zo min mogelijk interferentie zal ontstaan.

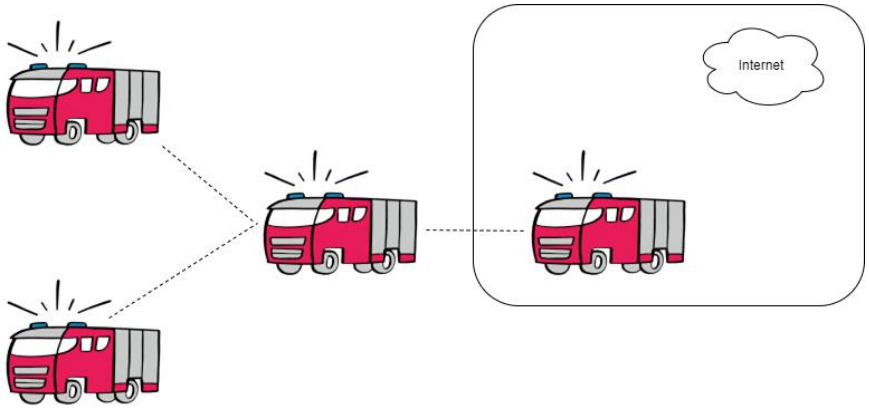
6.3.1.1 C2000

C2000 is het landelijke communicatiesysteem voor de hulpverleningsdiensten in Nederland. Het wordt 24/7 uur gebruikt door vooral politie, brandweer, ambulancediensten en bepaalde onderdelen van het ministerie van Defensie zoals de Koninklijke Marechaussee. Hulpverleners kunnen met C2000 communiceren met de meldkamer en met elkaar. Het systeem wordt ook gebruikt bij grootschalige incidenten en rampen. (C2000, 2022) Het C2000-systeem bestaat uit 3 verschillende onderdelen, zoals in onderstaand figuur kan worden gezien.

Voor T2000 zijn de frequentiebanden 380,000 MHz - 385,000 MHz (mobiel) en 390,000 MHz - 395,000 MHz (vast) in gebruik. P2000 maakt gebruik van de frequentie 169,650 MHz. (C2000, 2022)

6.3.1.2 Wifi en 4G

Naast de C2000-verbindingen bevindt zich op de brandweervoertuigen ook verschillende apparatuur die gebruikt maakt van 2.4 GHz Wifi. Deze verbindingen worden gebruikt voor de communicatie na buiten toe. Een concreet voorbeeld hiervan is het mesh-systeem van defensie die door de brandweer wordt gebruikt in gebieden, denk bijvoorbeeld aan natuurbranden, waar beperkte mobiele verbinding beschikbaar is. Dit systeem bouwt een mesh-netwerk op tussen alle brandweervoertuigen zodat ook als alleen nog een enkel brandweervoertuig verbinding naar buiten heeft, deze alle data kan versturen. (Balistreri, 2022)



Figuur 4. Vereenvoudigde illustratie mesh-systeem tijdens natuurbranden

Ook zit er op de voertuigen een combi-antenne welke de volgende verbindingen heeft:

- GPS;
- 2x 4G verbinding;
- 2x 2.4 GHz Wifi verbinding.

2.4GHz wifi gebruikt de frequentiebanden 2,400 MHz – 2,4835 MHz (Wired, 2022), 4G maakt gebruik van 1800 MHz en 2600 MHz. (KPN, 2022)

6.3.1.3 Bluetooth

Op dit moment wordt er nóg geen gebruik gemaakt van bluetooth. Het is echter niet uitgesloten dat hier in de toekomst gebruik van zal worden gemaakt. (Balistreri, 2022)

6.3.2 Benodigde snelheid

Naast interviews is ook onderzoek gedaan naar de benodigde snelheid. Om deze te bepalen is in dit onderzoek gekeken naar de grootte van verschillende requests. De gebruikte requests zijn representatief voor de verwachte requests die ook in het systeem zouden worden uitgevoerd. Na analyse van de gegevens is gebleken dat de gemiddelde grootte van dergelijke requests tussen de 25kb en 75kb ligt. Dit betekent dat, in het algemeen, requests die in het systeem uitgevoerd worden relatief klein zijn.

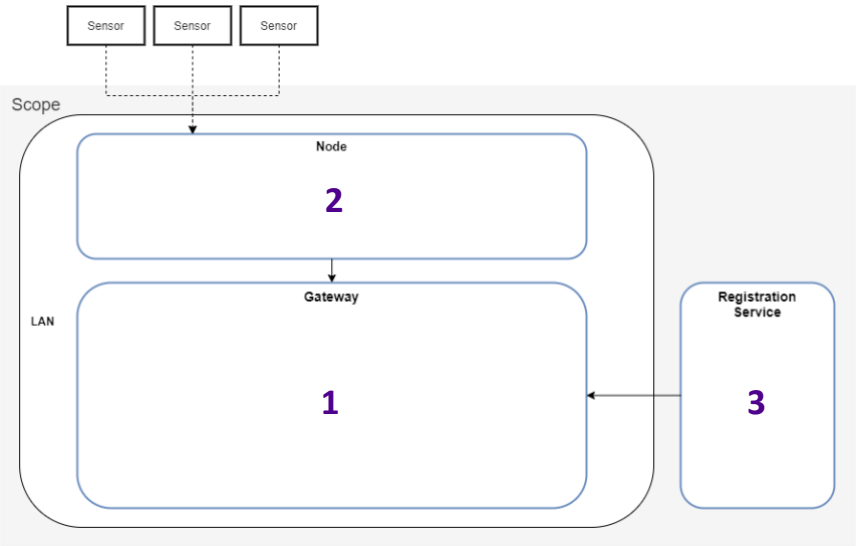
7 Mogelijke oplossing

Met de verkregen informatie uit de vorige fase kan nu een mogelijke oplossing worden geformuleerd. In dit hoofdstuk zal dan ook een concrete oplossing worden beschreven. Aan de hand van deze oplossing, welke tevens zal dienen als functioneel ontwerp en de opgedane kennis uit de vorige fase zullen vervolgens de (niet) functionele requirements worden opgesteld.

7.1 Functioneel ontwerp

Om tot het functionele ontwerp dan wel de beoogde oplossing te komen is er gebrainstormd om zo verschillende mogelijkheden te bekijken. De uiteindelijke beoogde oplossing zal bestaan uit verschillende componenten die samen de gewenste resultaten moeten hebben. Deze worden hieronder beschreven:

1. Een software gestuurde hardware architectuur, de gateway. Dit component zal zich op de tankautospuit bevinden en verantwoordelijk zijn voor:
    a. Het ophalen van de sensordata van de node (*component 2)*
    b. Het opslaan en visualiseren van deze sensordata
2. Eén of meerdere software gestuurde hardware architecturen, de nodes. Dit component zal zich op het ademluchttoestel bevinden en is verantwoordelijk voor:
    a. Het ophalen van de sensordata bij de sensoren.
    b. Het beschikbaar stellen van deze sensordata voor de gateway (*component 1)*.
3. Een software architectuur. Dit component zal zich off-site bevinden en is verantwoordelijk voor:
    a. Het bijhouden van gegevens over alle nodes (*component 2*) en gateways (*component 1*) binnen de brandweer.



Figuur 5. Beoogde architectuur

7.2 Requirements

Aan de hand van kennis uit voorgaande projectfasen en de beoogde oplossing zijn er verschillende requirements opgesteld, deze zijn tevens de basis voor het technisch ontwerp. De functionele requirements zijn volgens de 'MoSCoW'-methodiek geprioriteerd. Hieronder zal deze methodiek en de keuze hiervoor worden uitgelegd.

7.2.1 MoSCoW

Deze methodiek is gekozen, omdat deze instaat is om de requirements zo op te verdelen dat er een *minimum viable product* gegarandeerd kan worden, met daarnaast *nice to haves*. Dit sluit aan omdat de brandweer eerst wil experimenteren en het mogelijk vanuit die versie wil door ontwikkelen.

MoSCoW is een prioriteringstechniek die is ontwikkeld door Dai Clegg van Oracle. De techniek is afkomstig uit de softwareontwikkeling en is bekend geworden door het frequent gebruik als onderdeel van de Agile ontwikkelmethode Dynamics Systems Development Method (DSDM).

Hoewel afkomstig uit de softwareontwikkeling, wordt de techniek inmiddels toegepast in allerlei sectoren, waaronder de industrie, consultancy en andere zakelijke dienstverlening. (MoSCoW, 2022)

MoSCoW staat voor:

Must have:

Een must have is een harde eis waaraan niet te tornen valt. Hier moeten we gewoonweg aan voldoen. Als het beoogde resultaat niet wordt behaald, voldoet het projectresultaat niet aan de eisen en is de oplevering of het project mislukt.

Should have:

Een should have is ook een eis, maar geen keiharde. Hieraan moet eigenlijk ook aan worden voldaan, maar er is wel enige speling: misschien mag het resultaat enigszins afwijken, is er een alternatieve oplossing mogelijk, of kunnen de eisen bij een volgende oplevering meegenomen worden.

Could have:

Een could have is geen eis, maar een wens. Onze inspanning heeft meerwaarde en het is zeker nuttig de wens in te vullen, maar dit is niet noodzakelijk. Could haves zijn zaken wanneer er tijd is, tijd aan kunt besteden als aan de must en should haves is voldaan en er nog wat tijd en middelen over zijn.

Won't have:

Een won't have is een eis of wens, waarvan is afgesproken dat deze – in ieder geval in de huidige sprint/oplevering – niet binnen de scope valt. Het is dus niet zo, dat het leuk zou zijn als deze eis of wens wordt ingevuld. Integendeel: er mogen aan een won't have geen tijd en resources besteed worden, want dit heeft namelijk geen toegevoegde waarde of gaat ten koste van belangrijkere requirements.

7.2.2 Functionele requirements

Op basis van het reeds uitgevoerde onderzoek in de vorige fase en de beoogde oplossing zijn de functionele requirements naar voren gekomen, deze zijn gevalideerd met de organisatie.

Node

ID	Beschrijving	Prioriteit
RN1	De node moet een mesh-netwerk met andere node(s) en/of de gateway opbouwen	*Must*
RN2	De node moet sensordata kunnen accepteren.	*Must*
RN3	De node moet binnenkomende sensor data kunnen formatteren	*Must*
RN4	De node moet een GET-request kunnen accepteren	*Must*
RN5	Niet verstuurde data moet opnieuw verstuurd kunnen worden	*Should*
RN6	De node moet met zo min mogelijk configuratiesensoren herkennen	*Could*

Gateway

ID	Beschrijving	Prioriteit
RG1	De gateway moet een mesh-netwerk met de andere node(s) opbouwen	*Must*
RG2	De gateway moet de nodes kunnen scrapen en de sensor data correct opslaan	*Must*
RG3	De visualisatietool moet sensordata uit de database kunnen ophalen en real-time visualiseren	*Must*
RG4	De gateway moet eigen configuratiebestanden kunnen updaten	*Should*
RG5	De gateway moet actieve nodes kunnen ophalen middels een API	*Should*

Registratie service

ID	Beschrijving	Prioriteit
RA1	Moet een registratie van nodes en gateways bijhouden	*Should*
RA2	Moet een GET-request kunnen ontvangen	*Should*
RA3	Moet middels een API-key een gateway kunnen authentiseren en identificeren	*Should*
RA4	Moet de correcte configuratie voor de gateway kunnen genereren	*Should*

7.2.3 Niet functionele requirements

Uit de kennis opgedaan in de vorige fases zijn ook drie niet functionele requirements naar voren gekomen, ook deze zijn gevalideerd met de organisatie:

ID	Beschrijving
RO1	Een maximaal bereik van ongeveer 200 meter vanaf het voertuig
RO2	Een minimale snelheid van 100 kb/s
RO3	Werking op accu

7.2.3.1 Toelichting requirements

Het maximale bereik van 200 meter is gebaseerd op de interviews die zijn afgenomen. Hier werd echter ook aangeven dat meestal een bereik van nabij de 55 meter zou voldoen. Dit is namelijk de lengte van de hogedruk slang welke in de brandweervoertuigen aanwezig is en voornamelijk wordt gebruikt tijdens een inzet.

De minimale snelheid van 100 kb/s is gebaseerd op eigen onderzoek. In dit onderzoek is gekeken naar de grote van verschillende requests. De gebruikte requests zijn representatief voor de requests welke ook in het systeem uit gevoerd zouden worden. Hieruit is gebleken dat deze gemiddelde grote van een dergelijke request tussen de 25kb en 75kb zouden zitten. Daarom is ervoor gekozen een snelheid van 100kb/s aan te houden om ruim aan deze request te kunnen voldoen.

Tenslotte de werking op accu, door het analyseren van het proces van de brandweer en het onderzoek naar de omgeving waarin het systeem zou moeten werken. Ondanks dat het niet specifiek genoemd wordt, kan geconcludeerd worden dat het gewenst is dat de nodes op basis van accu moet kunnen werken.

8 Oplossingsrichting

Nadat aan de hand van de vorige projectfases een beoogde oplossing en de requirements bepaald zijn, zal er tijdens deze fase gekeken worden naar verschillende oplossingsrichtingen om uiteindelijk een proof of concept te kunnen realiseren.

Tijdens deze fase zal er onderzoek gedaan worden naar verschillende aspecten van de beoogde oplossing. De volgende aspecten zullen worden onderzocht in dit hoofdstuk:

1. Verzameling en opslag sensordata;
2. Real-time visualisatie;
3. Draadloze communicatie;
4. Geschikte Hardware;
5. API-framework.

Deze aspecten zijn direct te koppelen aan de eerder geformuleerde onderzoeksvragen. In dit hoofdstuk zal dan ook de focus liggen op het beantwoorden van deze vragen door middel van het onderzoek naar deze aspecten. Naast het beantwoorden van de onderzoeksvragen is onderzoek naar deze aspecten tevens belangrijk om verschillende delen van de requirements verder te kunnen uitwerken om uiteindelijk tot een proof of concept te kunnen komen.

8.1 Verzameling en opslag van sensordata

Binnen de organisatie wordt er voornamelijk gebruikt gemaakt van PostgreSQL. Op het eerste gezicht lijkt dit een uitstekend argument om PostgreSQL dan ook voor de realisatie van het proof of concept te gebruiken. Echter wordt er binnen de opdracht een specifiek soort data gebruikt, namelijk sensordata.

Sensordata, van sensoren zoals in de afbakening beschreven, heeft enkele specifieke eigenschappen, namelijk:

1. Het uitlezen van sensordata gebeurt altijd op een uniek moment in de tijd. Waardoor deze als unieke identifier gebruikt kunnen worden.
2. Sensordata is altijd een numerieke waarde, omdat het een meting betreft. Waardoor een specifiek datatype kan worden gebruikt.

Met deze kennis kan gekeken worden naar hoe dit soort data het beste verwerkt kan worden in een database. Zoals eerder benoemd wordt binnen de organisatie voornamelijk gebruik gemaakt van PostgreSQL. Dit zou een valide optie zijn, maar niet de meest efficiënte. Het belang van deze efficiëntie is al tijdens het opstellen van de deelvragen naar voren gekomen.

Een database type die juist uitermate geschikt is voor data met deze eigenschappen is een time-series database. Onderzoek naar deze database soort heeft uitgewezen dat deze database soort miljarden records zeer efficiënt kan opslaan door gebruikt te maken van de aanwezige timestamp. Door deze timestamp kan de data volgens een ander format opgeslagen worden welke de aanwezige algoritmes in staat stellen hier zeer efficiënt gebruik van te maken (Time series database, 2022).

Om te bepalen welke time-series database het meest geschikt zou zijn ia er gekeken naar verschillende alternatieven. Door gebruik te maken van drie verschillende bronnen is er uiteindelijk een drietal krachtige en veel gebruikte time-series databases naar voren gekomen (Time series database, 2022) (Best Time Series Databases, 2022) (7 Powerful Time-Series Databases, 2022).

Dit zijn de volgende drie time-series databases in volgorde van populariteit:

1. InfluxDB;
2. Kdb+;
3. Prometheus.

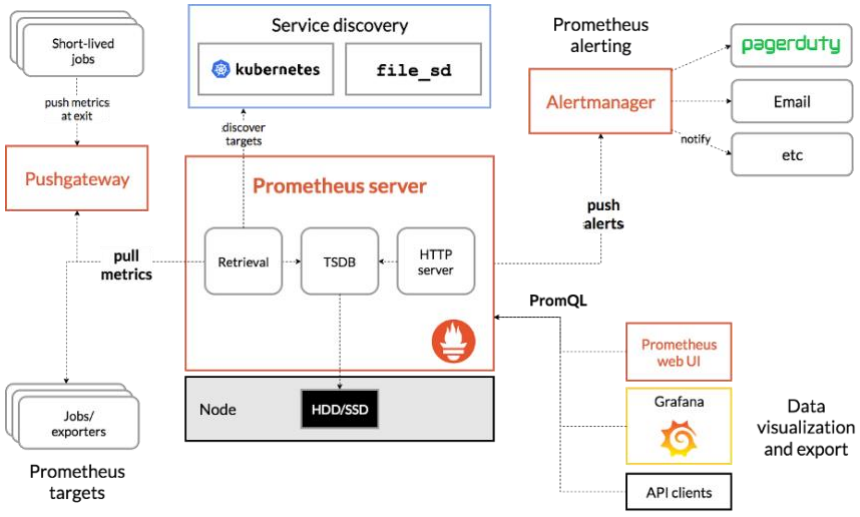
8.1.1 Conclusie

Theoretisch gezien zou de meest populaire optie de voorkeur hebben als er geen andere factoren meespelen. Echter heeft de organisatie de wens uitgesproken dat zoveel mogelijk volgens open standaarden wordt ontwikkeld. Enkel Prometheus is als open standaard ontwikkeld en is dus ook als oplossingsrichting gekozen.

8.1.2 Prometheus

Prometheus is een open source oplossing, welke zogenaamde *metrics* verzamelt via de scraping van HTTP end-points. Dit maakt een significant verschil ten opzichte van een conventionele database, waar de data normaliter naar de database wordt gestuurd. In dit systeem is dit echter een voordeel. Doordat hier de database juist bepaald waar de data vandaan wordt opgehaald kan de configuratie centraal op de gateway worden aangepast, onafhankelijk van de fysieke nodes. Hierdoor zou de registratie service ook enkel contact hebben met de gateway zelf en niks met de nodes hoeven te communiceren.

In de afbeelding is een Prometheus architectuur afgebeeld met verschillende gekoppelde componenten. De configuratie wordt bepaald in het blauwe vakje en de data van verschillende endpoints wordt gescraped en weergegeven links in de afbeelding. De visualisatie van de data is te zien rechtsonder. Er kunnen ook alerts worden toegevoegd, wat een goede vervolgstap zou kunnen zijn voor de VNOG.



Figuur 6. Typische Prometheus architectuur

8.2 Real-time visualisatie

Eén van de meest bekende manieren om data te visualiseren is het gebruik van een dashboard. Een dashboard is een instrument dat wordt gebruikt voor het verzamelen en weergeven van informatie en biedt dus een snel overzicht van data (Dashboard, 2022).

Een dashboard kan met veel verschillende technieken en talen worden ontwikkeld. Het is het meest optimaal om hierbij gebruik te maken van een framework. Het framework dient zo naadloos mogelijk aan te sluiten bij de datasoort en de opslag daarvan. Eerder is Prometheus als oplossingsrichting gekozen, op basis hiervan is vervolgonderzoek gedaan naar een geschikt framework. Uit onderzoek (How to visualize time series data, 2022) zijn een aantal mogelijkheden naar voren gekomen welke geschikt zouden kunnen zijn. Dit zijn de volgende opties:

- InlfuxDB
- Grafana
- Seeq

8.2.1 Conclusie

Alle opties zouden goede opties kunnen zijn om het gewenste resultaat te bereiken. Echter is uiteindelijk Grafana gekozen als oplossing. Garfana is gemaakt door de makers van Prometheus en sluit hierdoor daar dan ook naadloos op aan. Daarnaast ondersteunt dit framework ook een automatische realtime (1s) refresh rate, iets wat een requirement (RG3) is. Tenslotte zijn er geen significante nadelen voor Grafana naar voren gekomen. Daarom is er voor gekozen om Grafana als oplossingsrichting te kiezen.

Figuur 7. Voorbeeld van een visualisatie middels grafana.

8.3 Draadloze communicatie

Voor het bepalen van een oplossingsrichting voor draadloze communicatietechniek zijn de volgende twee niet functionele requirements relevant:

ID	Beschrijving
RO1	Een maximaal bereik van ongeveer 200 meter vanaf het voertuig
RO2	Een minimale snelheid van 100kb/s

Op basis van de bovenstaande requirements zullen er verschillende technieken worden onderzocht zodat deze vervolgens met de niet functionele requirements kunnen worden vergeleken.

8.3.1 Vergelijking verbindingstechnieken

Als eerst is er onderzoek (Rowley, 2022) gedaan naar verschillende, veel gebruikte, technieken. Hieruit zijn een zestal technieken naar voren gekomen waarvan na een initiële vergelijking er vier over bleven. Deze technieken zijn verder onderzocht door te kijken naar de aspecten: verbindingssnelheid, bereik, latency en stroomverbruik. Door juist deze aspecten ik kaart te brengen kan er worden bepaald op de techniek aan de requirements voldoet en of deze dan ook daadwerkelijk geschikt is voor de beoogde oplossing. Op basis van het onderzoek en de initiële vergelijking zijn de volgende vier technieken naar voren gekomen:

1. WiFi 6 (802.11ax);
2. 4G;
3. LoRaWAN;
4. Bluetooth 5.

Uit het vervolgonderzoek naar de bovenstaande technieken zijn de volgende resultaten naar voren gekomen:

	Wifi 6	4G	LoRaWAN	Bluetooth 5
Gem. Upload snelheid	200 Mbps	10 Mbps	0.3 kbps - 5 kbps	40 Mbps
Gem. Download snelheid	200 Mbps	20 Mbps	0.3 kbps - 5 kbps	40 Mbps
Max Bereik buiten	240 meter	n.v.t.	15000 meter	200 meter
Max Bereik binnen	70 meter	n.v.t.	3000 meter	40 meter
Gem. Latency	<4ms	35ms	NB	20 ms
Stroomverbruik	Relatief hoog	Relatief hoog	Erg weinig	Laag
Opmerkingen	Mesh netwerk mogelijk	Bereik afhankelijk van dekking	Duty cycle van <1% per dag 128 bit end to end encryption	Mesh netwerk mogelijk

(business.com, 2022) (Quora, 2022) (cnet.com, 2022) (KPN, 2022) (Tweakers, 2022) (Limitations, 2022) (BLE 5 vs BLE 4, 2022)

8.3.2 Afwegingen

Om te bepalen welke technieken zouden voldoen als oplossingsrichting, is er per techniek gekeken of deze aan de eerder gestelde requirements voldoen en of er nog verdere significante voor- en/of nadelen zijn. Hieronder staat schematisch het resultaat.

	Voldoet	Opmerkingen
WiFi 6	Ja	Erg hoge snelheden en erg lage latency[1], mesh-netwerk[2] optie om bereik uit te breiden.
4G	Ja	Voldoet, bereik is afhankelijk van de dekking van de provider welke niet overal voorziet. Dit is een groot minpunt voor deze techniek.
LoRaWAN	Nee	Maximale duty cycle[3] van <1% per dag, omgerekend 864 seconden per dag met een te lage snelheid.
Bluetooth 5	Ja	Voldoet aan alle eisen. Mesh-netwerk optie om bereik uit te breiden.

LoraWAN is de enige die niet aan één van de requirements voldoet, namelijk de snelheid. Naast het feit dat deze techniek een te lage duty cycle heeft. De overige opties voldoen wel aan de gestelde requirements. Ondanks dat 4G ook voldoet zal deze ook buiten beschouwing worden gelaten vanwege de afhankelijkheid van een derde partij en het bereik wat zij dicteren. De overige twee opties, Wifi 6 en Bluetooth 5 zijn beide valide opties.

8.3.3 Conclusie

Zowel WiFi 6 als Bluetooth 5 zijn beide geschikte technieken voor de beoogde oplossing, met als voordeel dat het bereik van beide kan worden uitgebreid door het gebruik van een mesh-netwerk. Uiteindelijk is gekozen om Wifi 6 als oplossingsrichting, omdat deze een beter bereik én lagere latency heeft dan Bluetooth 5.

Wifi 6 heeft de voorkeur echter is deze beperkt beschikbaar en wordt nog beperkt ondersteund. Daarom is gekozen om te werken met Wifi 5 (802.11ac). Deze is door de betere beschikbaarheid op verschillende platforms en de relatief beperkte verschil in functionaliteit t.o.v. Wifi 6 (Zie onderstaande tabel) als oplossingsrichting gekozen.

	Upload snelheid	Download snelheid	Bereik buiten	Bereik binnen
Wifi 5	140 Mbps	140 Mbps	220 meter	65 meter
Wifi 6	200 Mbps	200 Mbps	240 meter	70 meter

(Wifishop.nl, 2022)

[1] Vertraging in de dataoverdracht over een datacommunicatienetwerk. (Latentie, 2022)
[2] Een groep verbindingsapparaten, zoals wifi-routers die als één netwerk werken. (Wat is een mesh-netwerk?, 2022)
[3] De verhouding tussen de duur die een belasting of circuit aan is ten opzichte van de duur gedurende welk de belasting of het circuit uit is. (Wat is de duty cycle?, 2022)

8.4 Geschikte hardware

Voor het realiseren van de beoogde oplossing is ook hardware nodig om de betreffende software op te draaien. Om deze hardware te selecteren zijn er bepaalde eisen waar deze aan moeten voldoen, wat deze eisen moeten zijn, is naar voren gekomen in het vorige hoofdstuk. Daaruit is geconcludeerd dat Wifi 5 (801.11ac) als oplossingsrichting gekozen is.

8.4.1 Afwegingen

Als eerst is gekeken naar welke platformen er beschikbaar zijn en welke van deze er mee worden genomen in dit onderzoek. Uit verschillende bronnen (Rowley, 2022) (Kamal, 2020) (Murphy, 2022) is gebleken dat er vier veelgebruikte opties zijn voor dit soort projecten, namelijk:

- Raspberry Pi 4B;
- Raspberry Pi Zero W2;
- Arduino Wifi;
- ESP 32 (Espressif).

Om een goede afweging te kunnen maken is het van uitermate van belang de juiste parameters te bekijken. Daarom zijn deze parameters in het onderzoek meegenomen:

	Waarom	Afweging
RAM	Werkgeheugen is van belang om bijvoorbeeld sensor waarden tijdelijk te bewaren voordat deze opgeslagen zijn	Wordt op basis van inzicht meegenomen
Geheugen	Geheugen is belangrijk voor het opslaan van data, maar ook voor de maximale grote van de applicatie	Wordt op basis van inzicht meegenomen
GPIO pins	GPIO pins zijn van belang om verschillende soorten sensoren hardware matig te kunnen aansluiten.	Wordt op basis van inzicht meegenomen
Verbindingen	Voor de verbinding is Wifi 5 als oplossingsrichting gekozen en is de verbinding als parameter dus van belang	Moet beschikken over Wifi 5, andere verbindingen kunnen ‘nice to have’ zijn
OS	Belangrijk om te bepalen hoe software ontwikkeld dient te worden	Geen beslissende, voor volledigheid onderzoek.

Voor alle vier de platformen zijn de verschillende aspecten onderzocht. Hieronder worden de uitkomsten van het onderzoek weergegeven

	RB Pi 4B	RB Pi Zero W2	Arduino Wifi	ESP32
RAM	1, 2, 4 or 8 GB	512 MB	2 KB	16 KB
Geheugen	SD-card	SD-card	32 KB	320 KB
GPIO-pins	40 pins[4]	40 pins[4]	14 pins	43 pins[4]
Verbindingen	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE Gigabit Ethernet	2.4GHz 802.11 b/g/n wireless LAN Bluetooth 4.2, Bluetooth Low Energy (BLE), onboard antenna	Serial 802.11 b/g/n 2.4 GHz	802.11 b/g/n Bluetooth: v4.2 BR/EDR and BLE
OS	Linux based	Linux based	Geen	Geen
Voldoet voor Node	Ja, te krachtig	Ja	Ja	Ja
Voldoet voor Gateway	Ja	Nee	Nee	Nee

(RPi 4B specifications, 2022) (RPi Zero W2 specifications, 2022) (Arduino Uno Wifi, 2022) (ESP 32 datasheet, 2022)

8.4.2 Conclusie

Voor het platform van de nodes is de Raspberry Pi Zero W2 als oplossingsrichting gekozen. Dit platform ondersteunt de gekozen verbindingstechniek, bevat geruime mogelijkheden voor het aansluiten van sensoren, voldoende ruimte beschikbaar om een significant deel van de data te bufferen en voldoende overige specificaties. Daarnaast heeft de Raspberry Pi een besturingssysteem op basis van Linux, waardoor er geruime mogelijkheden zijn voor softwareontwikkeling in verschillende talen.

De specificaties van de Raspberry Pi Zero W2 zijn niet voldoende voor de gateway. Daarom is voor het platform van de gateway de Raspberry Pi 4B als oplossingsrichting gekozen. Dit platform heeft meer RAM waardoor deze geschikter zal zijn voor de gateway en bestaat de mogelijkheid zonder vast stroompunt online te zijn door middel van een accu. (Pi duration tests, 2022) (portable rpi projects, 2022)

8.4.3 Toekomst en alternatieven

De hierboven gekozen oplossingsrichting is in geen geval de enige oplossing waar dit proof of concept mee zou kunnen werken en het is dan ook niet uitgesloten dat in de toekomst, zeker met de huidige snelheid waarmee de techniek zich ontwikkelt, gebruik gemaakt kan worden van andere oplossingen.

[4] Niet alle pins kunnen gebruikt worden voor het aansluiten van sensoren. Zo zijn er verschillende pins die als ‘ground’ dienen en pins die gereserveerd zijn voor seriële connecties. Echter zijn er ruim voldoende pins beschikbaar voor het aansluiten van verschillende sensoren.

8.5 API framework

Voor deze opdracht zal er gebruik gemaakt gaan worden van een API. Daarom is er onderzoek gedaan naar de verschillende oplossingsrichtingen.

Bij het ontwikkelen van software is support van de community erg belangrijk. De community kan namelijk enorm helpen tijdens het ontwikkelen door het stellen van vragen op platforms als Stackoverflow. Daarom is er gekeken welke talen er op dit moment het meest gebruikt worden binnen de community. Er zijn twee talen die er met kop en schouders bovenuit springen, namelijk Python en Java (Insights 2022, 2022).

Omdat deze twee talen het meest gebruikt worden, wordt gekeken naar de verschillen tussen de frameworks van beide talen. Voor java is er voor Springboot gekozen. Hiervoor is gekozen, omdat dit de meest gebruikte en bekendste framework binnen Java is. Voor python is er gekozen voor Flask. Hiervoor is gekozen omdat Flask binnen de VNOG de standaard is. Om deze twee opties te vergelijken is er gekeken naar voor- en nadelen van beide opties. Dit is hieronder schematisch weergegeven.

Framework	Voordelen	Nadelen
Springboot	• Snel en makkelijk om te ontwikkelen • Grote community • Beperkte source code • Standalone	• Veel ongebruikte afhankelijkheden • Niet geschikt voor erg grote complexe applicaties
Flask	• Makkelijk te leren • Flexibel • Geïntegreerde testen	• 1 request per keer per instantie • Makkelijk lage kwaliteit code te schrijven

(pros and cons of springboot, 2022) (Python vs Flask, 2022)

8.5.1 Conclusie

Beide opties hebben goede voor- en nadelen ten opzichte van elkaar, geen van beide opties heeft écht een echt groot voordeel ten opzichte van de ander. Maar omdat Flask reeds bekend is binnen de organisatie is het framework Flask en dus Python als oplossingsrichting gekozen.

9 Realisatie

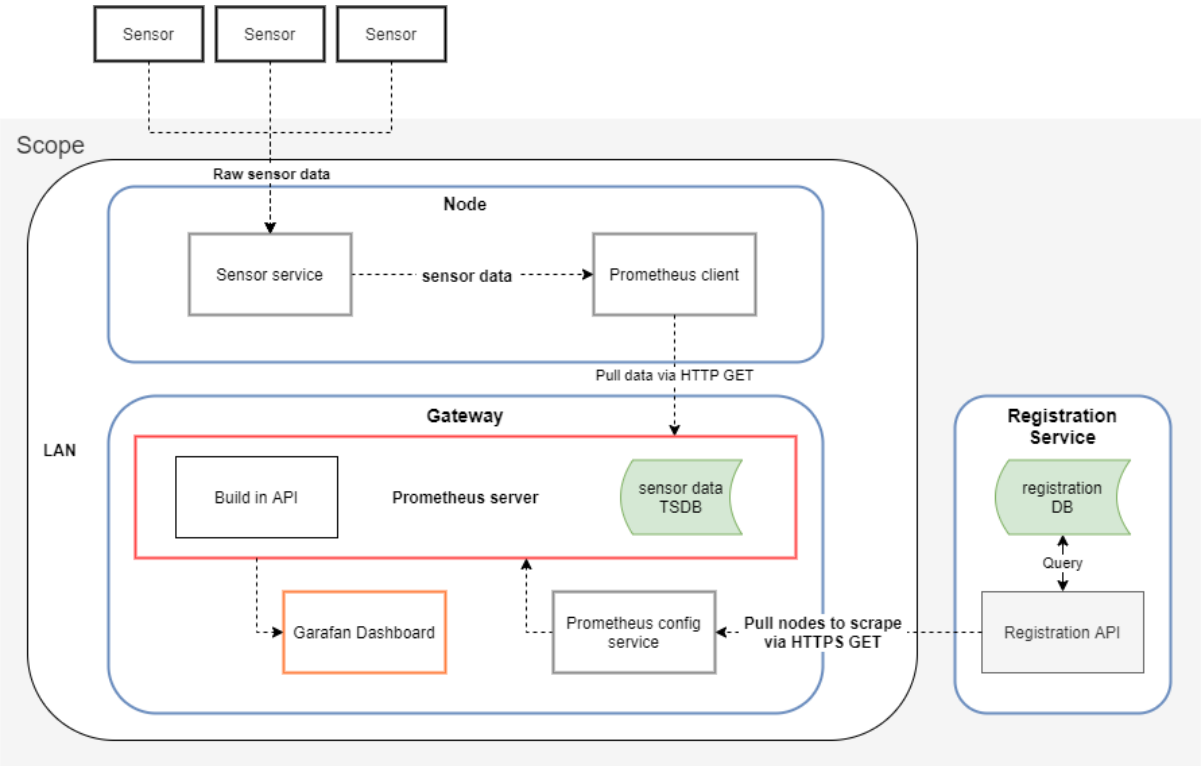
In het vorige hoofdstuk is beschreven welke oplossingsrichtingen er gekozen zijn en waarom deze gekozen zijn. De volgende stap in het proces is het ontwerpen van een technisch ontwerp en de realisatie hiervan. Het technisch ontwerp geeft een weergave van de werking van de oplossing en licht de gemaakte keuzes toe.

9.1 Technisch ontwerp

Het primaire doel van de software zal de communicatie tussen de nodes en de gateway zijn. Daarnaast is het van belang dat de software de gegenereerde data op kan slaan en dat de mogelijkheid bestaat tot visualisatie van de data. Ook zal er een registratieservice moeten zijn, die bijhoudt welke nodes data mogen sturen.

9.1.1 Architectuur

De architectuur voor de oplossing zal bestaan uit drie verschillende componenten, namelijk de Node, Gateway en de Registration service, die met elkaar communiceren via het HTTP dan wel het HTTPS-protocol. De nodes en gateways zullen zich in een ad-hoc LAN-netwerk bevinden (RN1 & RG1).



Figuur 8. Software architectuur proof of concept.

Als eerst het *Node*-component, dit component is verantwoordelijk voor het binnen halen en registreren van de binnenkomende sensordata en deze vervolgens beschikbaar te stellen voor het gateway-component. De data van verscheidene sensoren zal worden uitgelezen middels de sensor service (RN2) en dit vervolgens worden vertaald naar een *metric* (RN3), welke kan worden begrepen door de Prometheus-client. De Prometheus-client is een client die gemaakt is om te communiceren

met de Prometheus-server, welke wordt gebruikt om de data op te slaan. De Prometheus-client zorgt ervoor dat de data die door de sensorservice is geformatteerd nu klaar staat om opgehaald te worden (RN4).

Het tweede component is de *Gateway*. De Gateway draagt zorg voor het binnenhalen van de data en het visualiseren van deze data. De kern van de Gateway is de Prometheus-server. Deze server zorgt er voor dat deze de data van de verschillende nodes kan scrapen en kan opslaan in zijn eigen time-series database (RG2). Om de data te visualiseren wordt er een dashboard gebruikt. Hiervoor wordt gebruik gemaakt van Grafana, een framework speciaal gemaakt om data uit een Prometheus server te visualiseren (RG3). Tenslotte bevat de gateway een configuratieservice, deze service is verantwoordelijk voor het ophalen van de actieve nodes via de registratie API (RG5) en vervolgens het updaten van de juiste configuratie bestanden (RG4).

Het laatste component van de architectuur is de *Registation service*. Dit component draagt zorg voor het bijhouden van alle nodes en gateways. Zo kan er hier worden bijgehouden of een node of gateway actief is en wat zijn locatie is. Deze gegevens worden opgeslagen in een database (RA1), deze kan vervolgens benaderd worden via een API (RA2).

9.1.2 Ad-hoc netwerk

Voor het opzetten van het ad-hoc (mesh) netwerk wordt gebruik gemaakt van BATMAN advanced. Hiervoor is gekozen, omdat de focus bij deze techniek ligt op de decentralisatie van kennis over de beste route door het netwerk - geen enkel knooppunt heeft alle gegevens. Deze techniek elimineert de noodzaak om informatie over netwerkwijzigingen naar elk knooppunt in het netwerk te verspreiden. Het individuele knooppunt slaat alleen informatie op over de "richting" waarvan het gegevens heeft ontvangen. De gegevens worden van knooppunt naar knooppunt doorgegeven en pakketten krijgen individuele, dynamisch gecreëerde routes. Er ontstaat zo dus een netwerk van collectieve intelligentie. (B.A.T.M.A.N., 2022) Om deze reden is deze techniek een uitermate goede oplossing voor het te ontwikkelen systeem.

BATMAN maakt tevens geen gebruik meer van IP-adressen, in tegenstelling tot conventionele manieren die gebruik maken van routing tables. BATMAN werkt namelijk op laag 2 van het OSI-model en beïnvloedt op deze manier niet bovenliggende protocollen. BATMAN werkt als een linux kernel driver en beperkt zo overhead tot een minimum en is niet afhankelijk van andere drivers. BATMAN kan gebruikt worden samen met onder andere wifi, ethernet, lan, vpn etc. (Batman-adv, 2022).

9.1.3 Gateway

Het hoofddoel van de gateway is het opslaan en visualiseren van de data middels Prometheus en Grafana. Echter heeft prometheus een configuratie nodig, welke beschrijft van waar de data gescraped kan worden. Hiervoor is de configuratieservice in het leven geroepen.

9.1.3.1 Configuratieservice

De configuratieservice is het onderdeel van de gateway welke verantwoordelijk is voor het aanspreken van de registratie API en het updaten van de Prometheusconfiguratie. Deze service haalt eerst de benodigde JSON op middels een HTTPS GET request, welke vervolgens wordt weggeschreven naar het configuratiebestand. Ten slotte stuurt de service een HTTP POST request naar de Prometheusserver om deze te laten weten dat er een nieuwe configuratie geladen wordt.

De configuratieservice kan als los script worden uitgevoerd, waardoor er op ieder gewenst moment de huidige configuratie geladen kan worden. Daarnaast staat cruciale informatie zoals de url van de registratie API, maar ook de API key van de gateway in een JSON-configuratiebestand. Hierdoor zijn deze makkelijk aan te passen en te bewerken, indien nodig.

9.1.3.2 Docker

Zowel de Prometheusserver als Garfana worden middels Docker opgezet. Docker is een platform welke Docker-containers draait, een Docker-container is een verpakkingsindeling waarmee alle code en afhankelijkheden van een applicatie worden verpakt in een standaardindeling, waardoor deze snel en betrouwbaar in computeromgevingen kan worden uitgevoerd. Een Docker-container is een populaire lichtgewicht, zelfstandige, uitvoerbare container die alles bevat wat nodig is om een applicatie uit te voeren, inclusief bibliotheken, systeemtools, code en runtime. Docker is ook een softwareplatform waarmee ontwikkelaars snel applicaties in containers kunnen bouwen, testen en implementeren. (Docker, 2022) Er is voor Docker gekozen, omdat op deze manier snel instanties van dit systeem kunnen worden opgezet op verschillende gateways.

9.1.4 Node

Omdat er voor de gateway gekozen is voor de Prometheusserver is het meer dan logisch dat er voor de node gebruik is gemaakt van de Prometheus-client. De Prometheus-client is feitelijk een server welke alle beschikbare metrics binnen de applicatie exposed op het *'/metrics'* end point. Via dit end point kan de Prometheus-server vervolgens de beschikbare data scrapen.

9.1.4.1 Sensorservice

De sensorservice is de service welke verantwoordelijk is voor het binnenhalen van de sensordata en deze te formatteren in metrics. Middels een scheduler wordt elke vijf seconden de data van de sensoren weggeschreven naar een metric, welke vervolgens door de Prometheus-client wordt verwerkt.

9.1.5 Registration service

Binnen de Registration service wordt de administratie van nodes en gateways bijgehouden. Op deze manier kunnen er gemakkelijk nodes en gateways worden toegevoegd of buiten werking worden gesteld. De Registration service kan op deze manier ook zorgen dat er configuraties beschikbaar gesteld kunnen worden voor de gateways.

9.1.5.1 Database

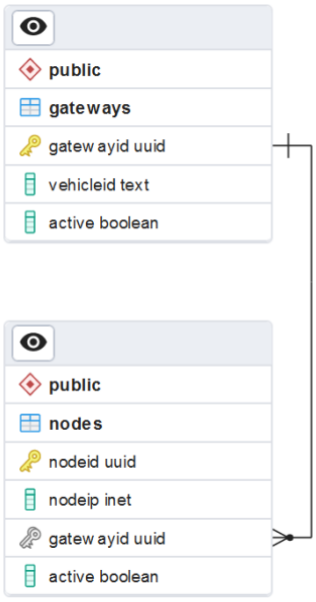
Voor de keuze van de database binnen de service is gekozen voor PostgreSQL-database. Deze database is binnen de organisatie de standaard en past naadloos binnen deze service omdat het hier geen speciaal soort data betreft. De database is opgezet middels Docker samen met de administratie tool PgAdmin4. Voor het ontwerp van de database is uitgegaan van het volgende model.

Gateway

gateway_id	UUID van de gateway. Tevens de primary key
vehicle_id	Het id van het brandweervoertuig waarop de gateway zich bevind, elk brandweervoertuig heeft een uniek voertuig nummer
active	Een boolean-waarde die aangeeft of de desbetreffende gateway actief is

Node

node_id	UUID van de node. Tevens de primary key
node_ip	Het lokale statische IP-adres van de node. Via dit IP kan de node gescraped worden door de gateway
gateway_id	UUID van de gateway waaraan de node is gekoppeld. Deze waarde is tevens een foreign key refererend naar `gateway_id` in de Gateway tabel
active	Een boolean waarde die aangeeft of de desbetreffende node actief is

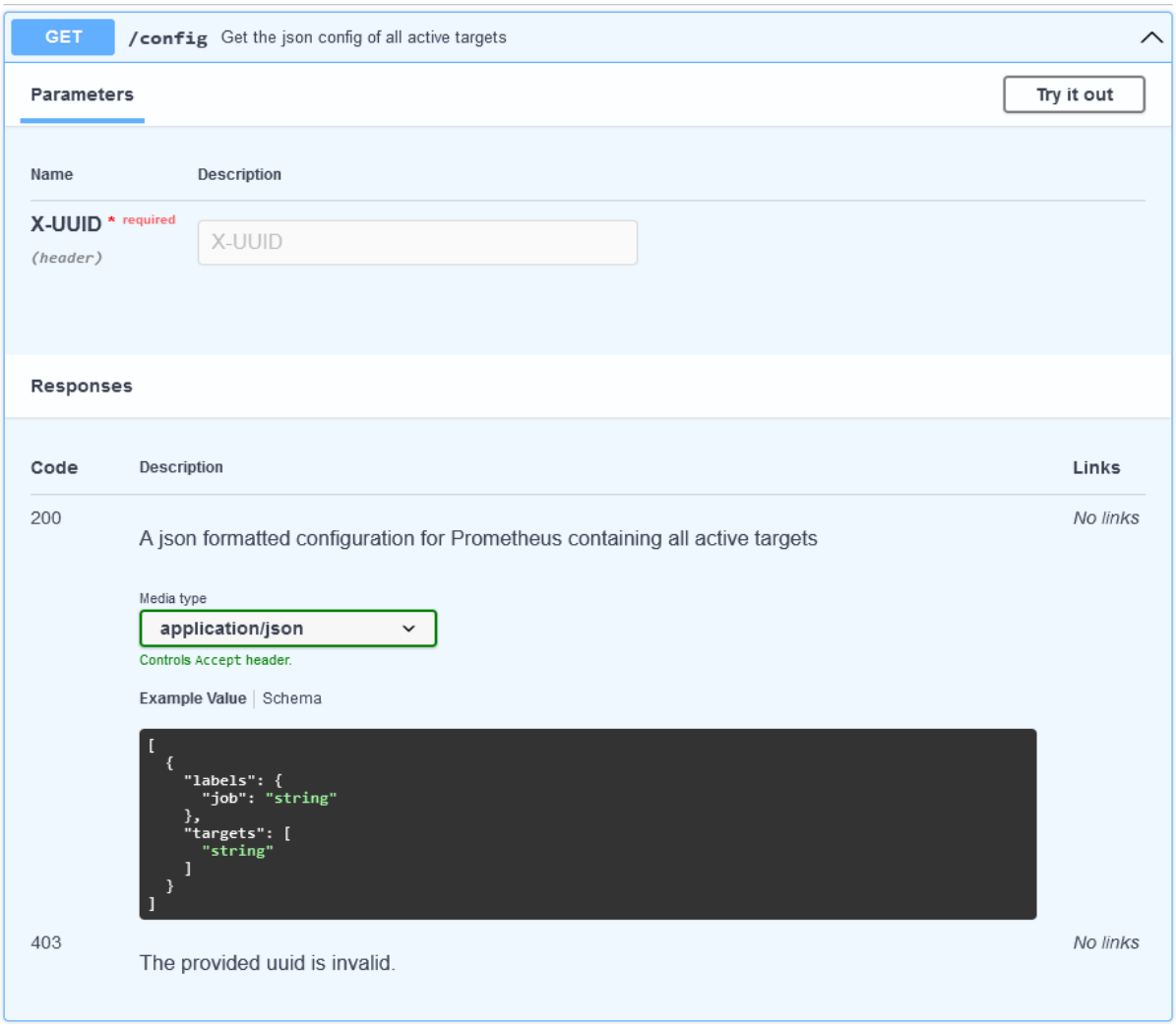


Figuur 9. Entity relationship diagram

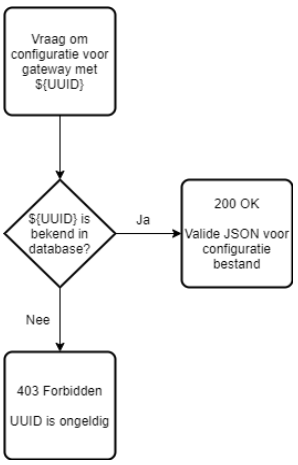
9.1.5.2 API

Voor de API zal gebruik worden gemaakt van een REST-API zoals beschreven in de API design rules (REST API design rules, 2022), welke door de overheid is verplicht (API, 2022). Hierbij is uitgegaan van het volgende ontwerp. Het volledige ontwerp volgens de Open-API 3 specificatie kan worden gevonden in bijlage 3.

GET /config



Dit path is gemaakt om de configuratie service binnen de gateway de configuratie van de Prometheus-server te laten aanpassen. Bij het doen van deze get request dient de UUID van de gateway mee te worden gegeven. De volgende flowchart zal worden afgelopen binnen de API:



Figuur 10. Flowchart /config.

PATCH /gateways

PATCH /gateways update the state of a node

Parameters — Try it out

Name	Description
X-UUID * required (header)	X-UUID
X-New-State * required (header)	X-New-State

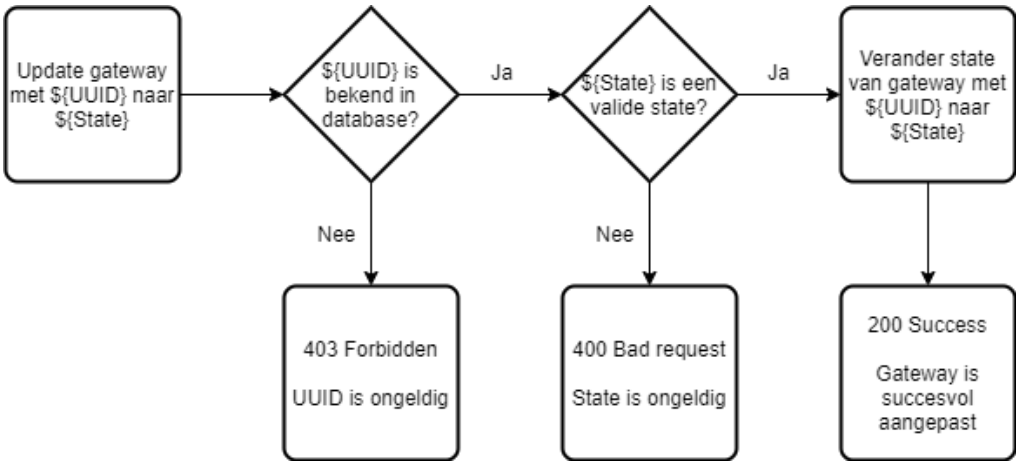
Responses

Code	Description	Links
200	The was gateway updated succesfully	*No links*
400	The provided state is invalid.	*No links*
403	The provided uuid is invalid.	*No links*

Dit path is gemaakt voor het op actief of non-actief zetten van de gateways binnen de database. Bij het doen van deze request dienen de volgende twee headers mee worden gegeven:

- **UUID:** De UUID van de gateway die aangepast dient te worden;
- **New-State:** De nieuwe staat van de gateway, actief (true) of non-actief (false)

De volgende flowchart zal worden afgelopen binnen de API:



Figuur 11. Flowchart /configs.

Patch /nodes

PATCH /nodes update the state of a node

Parameters | Try it out

Name	Description
X-UUID * required (header)	X-UUID
X-New-State * required (header)	X-New-State

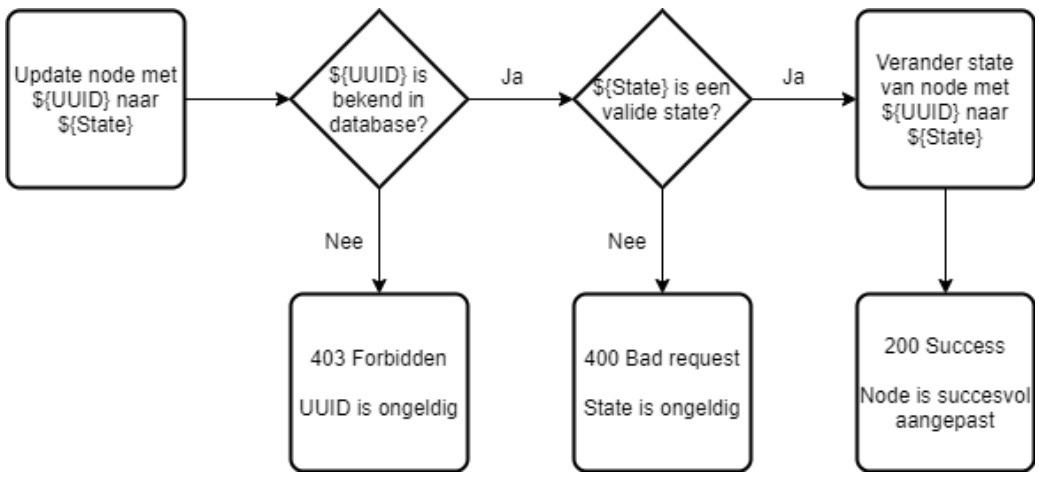
Responses

Code	Description	Links
200	The node was updated succesfully	*No links*
400	The provided state is invalid.	*No links*
403	The provided uuid is invalid.	*No links*

Dit path is gemaakt voor het op actief of non-actief zetten van de nodes binnen de database. Bij het doen van deze request dienen de volgende twee headers mee worden gegeven

- **UUID:** De UUID van de node die aangepast dient te worden
- **New-State:** De nieuwe staat van de gateway, actief (true) of non-actief (false)

De volgende flowchart zal worden afgelopen binnen de API:



Figuur 12. Flowchart /nodes.

9.1.5.3 Authenticatie

Voor het gebruik van de API is gekozen voor het gebruik van de UUID v4 als een soort API key. De UUID is de UUID van de betreffende node of gateway. Deze keuze is gemaakt op basis van het volgende:

1. Een UUID v4 is altijd random en vrijwel onmogelijk brute-forcebaar.
2. De UUID van de gateways is enkel bekend bij de gateways zelf en in de database.
3. Het bemachtigen van een UUID van de gateway:
    a. levert enkel een lijst op met lokale IP-adressen, die alleen nuttig zijn als men toegang heeft tot de LAN waarin de gateway zich bevindt.
        i. Toegang tot de LAN van een voertuig van buitenaf is vrijwel onmogelijk door de wisselende IP-adressen van de voertuigen.
4. De UUID van de nodes:
    a. is enkel bekend in de database
    b. geeft geen enkel de mogelijkheid om de betreffende node op actief te zetten.

Doordat de UUID's zo uniek zijn en bemachtiging van een UUID geen bruikbare informatie prijsgeeft zonder toegang te hebben tot de LAN waarin de gateway zich bevindt, is het gebruik op deze manier voldoende veilig.

9.2 Realisatie

In het vorige hoofdstuk is besproken hoe de technische structuur van het proof of concept eruitziet. In dit hoofdstuk zal besproken worden hoe het implementeren van verschillende componenten van het prototype is verlopen en zullen de tijdens de realisatie gemaakte keuzes worden toegelicht.

9.2.1 Gateway configuratie

Omdat de architectuur van de gateway in een dynamische omgeving werkt, is er zoals eerder toegelicht de keuze gemaakt voor het ontwerpen van een configuratieservice. Deze configuratieservice is op de volgende manier gerealiseerd.

Als eerste haalt de gateway uit een JSON-configuratiebestand de benodigde parameters. Dit bestand is als volgt opgebouwd:

1. **gatewayUuid** UUID van de betreffende gateway
2. **apiUrl** Url van de API waar de registratie wordt bijgehouden
3. **prometheusReloadUrl** Url welke de prometheus-server laat weten dat er een nieuwe configuratie beschikbaar is
4. **configFile** Path naar het configuratie bestand van de Prometheus server

Daarna wordt via de API de nieuwe configuratie met alle beschikbare nodes opgehaald en wordt deze weggeschreven naar de locatie, zoals gedefinieerd in het configuratiebestand.

Als laatste stap geeft de configuratieservice Prometheus een seintje dat deze de nieuwe configuratie kan ophalen.

Door de configuratie op deze manier in te zetten kunnen makkelijk aanpassingen worden gemaakt. Door deze snelle aanpassingen kan snel getest worden met verschillende instanties van de services.

```python
def get_targets(uuid):
    payload = {'X-UUID': uuid}
    resp = requests.get(api_url, headers=payload)

    while resp.status_code != 200:
        resp = requests.get(api_url)

    return resp.json()


Rick Holtman
def update_targets(new_targets_json):
    with open(config_file, 'w', encoding='utf-8') as f:
        json.dump(new_targets_json, f, ensure_ascii=False, indent=4)


Rick Holtman
def notify():
    requests.post(prometheus_reload_url)


if __name__ == '__main__':
    update_targets(get_targets(gateway_uuid))
    notify()
```

Figuur 13. Configuratie service

9.2.2 Registratie API

Bij het ontwerpen met de API is er voornamelijk gefocust op het afvangen van invalide input. Om dit zo overzichtelijk mogelijk te houden, is er gekozen om binnen de applicatie gebruik te maken van custom exceptions. Door deze custom exceptions blijft de applicatie overzichtelijk en kunnen er tevens specifieke statuscodes worden teruggegeven op basis van de request.

```
@app.route('/gateways', methods=['UPDATE'])
def gateways():
    uuid = request.headers.get('X-UUID')
    state = request.headers.get('X-New-State')

    try:
        validate_gateway_uuid(uuid)
        validate_state(state)

        update_gateway_by_uuid(uuid, state)

        return "Success", 200

    except exceptions.stateInvalid:
        return "Bad request, the provided X-New-State should be either true or false", 400

    except exceptions.uuidInvalid:
        return "Forbidden, the provided X-UUID is not valid. Please update the UUID or register it in the database", 403
```

Figuur 14. Voorbeeld gebruik van exceptions

9.2.3 Ad-hoc netwerk

Voor het opzetten van het ad-hoc netwerk op de verschillende Raspberry Pi's zijn veel stappen vereist, waarvan sommige stappen elke keer dat het systeem herstart opnieuw uitgevoerd moeten worden. Om dit proces te versimpelen is ervoor gekozen om een shell-script te schrijven en deze bij het opstarten van het systeem automatisch te laten uitvoeren. Dit script bevat de volgende stappen:

1. Het toevoegen van *batman-adv* aan de Linux kernel.
2. Het configureren van de wifi netwerk adapter (wlan0) naar ad-hoc modus, deze een statisch IPv4 adres geven en het configureren van overige aspecten zoals onder andere de *essid*.
3. Het koppelen van *bat0* netwerk adapter en de *wlan0* netwerk adapter via *batman-adv*.
4. Het online zetten van de *bat0* netwerk adapter.

Dit script wordt automatisch uitgevoerd doordat de volgende regel is toegevoegd aan */etc/rc.local*.

```
sudo sh /path/to/batctl/batsetup.sh
```

9.2.4 Status requirements

Na de realisatie zijn alle requirements op twee na afgerond, dit zijn RN5 en RN6. Deze requirements zijn niet afgerond wegens een gebrek aan ontwikkeltijd.

Node

ID	Beschrijving	Prioriteit	Voldaan
RN1	De node moet een mesh netwerk met andere node(s) en/of de gateway opbouwen.	*Must*	✓
RN2	De sensor service moet inkomende sensordata kunnen formatteren naar een *metric*	*Must*	✓
RN3	De sensor service moet de *metrics* kunnen exporteren	*Must*	✓
RN4	De Prometheus client moet een GET request kunnen accepteren	*Must*	✓
RN5	De dataservice moet missende data opnieuw kunnen sturen	*Should*	✗
RN6	De sensor service moet met zo min mogelijk configuratie de soort sensordata herkennen.	*Could*	✗

Gateway

ID	Beschrijving	Prioriteit	Voldaan
RG1	De gateway moet een mesh netwerk met de andere node(s) opbouwen.	*Must*	✓
RG2	De Prometheus server moet de nodes kunnen scrapen en deze correct opslaan.	*Must*	✓
RG3	Het dashboard moet data uit de Prometheus server kunnen ophalen en real time visualiseren.	*Must*	✓
RG4	De configuratieservice moet configuratiebestanden van de Prometheus server kunnen updaten.	*Should*	✓
RG5	De configuratieservice moet actieve nodes kunnen ophalen middels een API.	*Should*	✓

Registratie service

ID	Beschrijving	Prioriteit	Voldaan
RA1	Moet een registratie van nodes en gateways bijhouden.	*Should*	✓
RA2	Moet een GET request kunnen ontvangen.	*Should*	✓
RA3	Moet middels een API key een gateway kunnen authentiseren en identificeren.	*Should*	✓
RA4	Moet de correcte configuratie voor de gateway kunnen genereren.	*Should*	✓

10 Testen en resultaten

Tot slot het testen van het gerealiseerde prototype. Door het prototype aan testen te onderwerpen kan worden bevestigd of deze aan de gestelde requirements voldoet. Op deze wijze kan worden vastgesteld of het ontwerp de hoofdvraag beantwoord. Daarnaast is testen tevens een belangrijk onderdeel van het schrijven van software. Testen zijn belangrijk om te kunnen waarborgen dat de software ook functioneert zoals ontworpen. Daarom zal er in dit hoofdstuk besproken worden hoe de software, de hardware en de combinatie hiervan is getest.

10.1 Software testen

Er zijn verschillende manieren om software te testen. Software wordt normaliter op drie verschillende manieren getest. Namelijk unit testen, integratie testen en handmatige testen. Binnen dit project is er gebruik gemaakt van handmatige en unit testen. Er was helaas geen tijd beschikbaar voor het implementeren van integratie testen.

10.1.1 Handmatige en unit testen

Bij handmatige testen wordt de nieuwe functionaliteit getest zonder een geautomatiseerde tool. De handmatige testen worden gebruikt als er (nog) geen integratie testen geschreven zijn. De handmatige testen worden uitgevoerd door de ontwikkelaar zelf. Bij het testen zijn zowel de happy en bad flow getest.

Unit tests zijn kleine, individuele tests die specifieke delen van je code testen om te controleren of die delen werken zoals verwacht. Ze zijn belangrijk omdat ze helpen om fouten en bugs te identificeren en op te lossen voordat ze impact hebben op de gebruiker of de rest van de applicatie. Bovendien maken unit tests het gemakkelijker om je code te onderhouden en te verbeteren, omdat ze ervoor zorgen dat je code betrouwbaar en gemakkelijk te begrijpen is voor andere ontwikkelaars.

Hoewel unit tests en handmatige tests verschillende doelen hebben, zijn ze beide belangrijk voor het ontwikkelen en onderhouden van software-applicaties. Unit tests helpen ontwikkelaars om te controleren of specifieke stukken code werken zoals verwacht, terwijl handmatige tests worden gebruikt om te controleren of de volledige applicatie werkt zoals bedoeld. Bovendien kunnen handmatige tests worden gebruikt om te testen of de software aan de behoeften van de gebruiker voldoet. Zowel unit tests als handmatige tests zijn belangrijk omdat ze helpen om fouten en bugs te identificeren en op te lossen, wat kan leiden tot betere en betrouwbaardere software.

10.2 Unit testen

Bij deze testen is gebruik gemaakt van setup en teardown functies. Het doel van deze functies is om ervoor te zorgen dat elke test in een testcase afzonderlijk wordt uitgevoerd, zodat de resultaten van een test niet beïnvloed worden door de resultaten van een andere test. Dit maakt het makkelijker om te begrijpen waarom een test mislukt en hoe die opgelost kan worden. Bovendien zorgen setup en teardown ervoor dat je code schoon en efficiënt blijft, omdat ze ervoor zorgen dat de tests niet overlappen en dat de testomgeving altijd in dezelfde staat is voordat elke test wordt uitgevoerd.

De test_validate_gateway_uuid test controleert bijvoorbeeld of de validate_gateway_uuid functie een exceptie gooit als de opgegeven gateway UUID niet wordt gevonden in de database, en of de functie geen exceptie gooit als de UUID wel wordt gevonden.

De test_validate_node_uuid test controleert of de validate_node_uuid functie een exceptie gooit als de opgegeven node UUID niet wordt gevonden in de database, en of de functie geen exceptie gooit als de UUID wel wordt gevonden.

De test_get_nodes_by_gateway test controleert of de get_nodes_by_gateway functie een lijst van nodes teruggeeft die zijn gekoppeld aan de opgegeven gateway.

De test_update_node_by_uuid test controleert of de update_node_by_uuid functie de actieve staat van de node met de opgegeven UUID bijwerkt.

De test_update_gateway_by_uuid test controleert of de update_gateway_by_uuid functie de actieve staat van de gateway met de opgegeven UUID bijwerkt.

De test_targets_route test controleert of de /targets route een lijst van nodes teruggeeft die zijn gekoppeld aan de opgegeven gateway, en of de route een 403 error teruggeeft als de opgegeven gateway UUID niet geldig is.

De test_nodes_route test controleert of de /nodes route de actieve staat van de node met de opgegeven UUID bijwerkt, en of de route een 400 error teruggeeft als de opgegeven staat niet geldig is, en of de route een 403 error teruggeeft als de opgegeven node UUID niet geldig is.

De test_gateways_route test controleert of de /gateways route de actieve staat van de gateway met de opgegeven UUID bijwerkt, en of de route een 400 error teruggeeft als de opgegeven staat niet geldig is, en of de route een 403 error teruggeeft als de opgegeven gateway UUID niet geldig is.

Alle unit testen zijn in tabel vorm opgenomen in de bijlage.

```python
def test_targets_route(self):
    # Test that the targets route returns a list of nodes associated with the specified gateway
    with app.test_client() as c:
        response = c.get('/targets', headers={'X-UUID': self.gateway.gatewayid})
        self.assertEqual(response.status_code, 200)
        self.assertIsInstance(response.json, list)
        self.assertTrue(all(isinstance(node, dict) for node in response.json))

    # Test that the targets route returns a 403 error if the provided gateway UUID is not valid
    with app.test_client() as c:
        response = c.get('/targets', headers={'X-UUID': 'invalid-uuid'})
        self.assertEqual(response.status_code, 403)
```

Figuur 15. Deel van een geschreven unit test

10.3 Hardware en omgevingstesten

Om het systeem te testen op bruikbaarheid zijn verschillende testen uitgevoerd. Echter zijn deze tests niet volledig met de beoogde hardware uitgevoerd. Door de huidige lange levertijden van de hardware zijn de tests voor de nodes met verschillende oudere modellen van de Raspberry Pi, een Raspberry Pi 3 model B en model B+, uitgevoerd. De hardware voor de gateway zal wel met de gekozen hardware worden uitgevoerd.

Er zijn verschillende testen uitgevoerd om te beoordelen of de software dan wel hardware daadwerkelijk voldoen aan de requirements. Tijdens deze testen zijn er verschillende scenario's uitgevoerd. Deze scenario's zijn, representatief voor de realiteit. Tijdens deze scenario's zijn de volgende karakteristieken genoteerd:

1. De latency
    a. Laagste (min)
    b. Gemiddelde (avg)
    c. Hoogste (max)
    d. Standaardafwijking (mdev)
2. De afstand tussen de nodes dan wel de gateway
3. Obstakels

10.3.1 Scenario's

Om te bewijzen dat het prototype het gewenste gedrag vertoont zijn er verschillende testen uitgevoerd. Dit is gedaan door de uitkomsten van de verschillende scenario's in een tabel uit te werken. Deze tabel is als volgt opgebouwd:

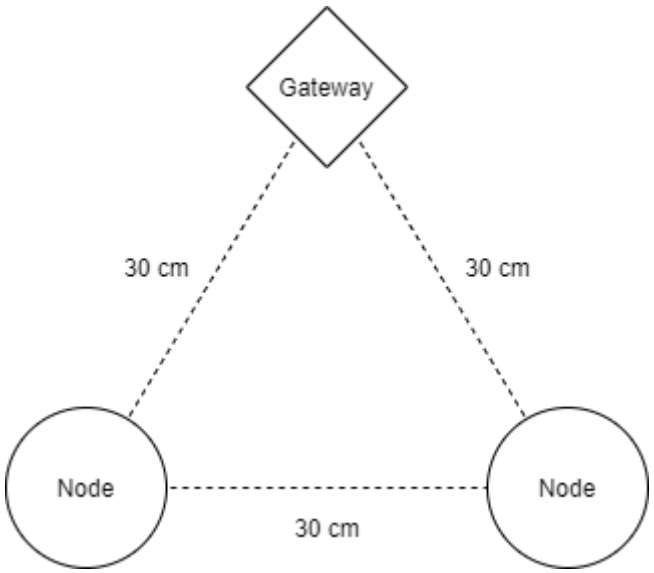
- **ID:** Het nummer van de uitgevoerde test
- **Omschrijving:** De omschrijving van de uitgevoerde test
- **Requirements:** De mogelijk gekoppelde requirements
- **Test methode:** De gebruikte test methode zoals deze hierboven beschreven zijn.
- **Start scenario:** De status van het systeem voordat er begonnen is met testen
- **Stappen:** De stappen die gevolgd moeten worden
- **Resultaat:** De status van het systeem na het volgen van de stappen

Hieronder staat één van de uitgewerkt test scenario's. De overige test scenario's zijn te vinden in Bijlage 1.

ID	HT4
OMSCHRIJVING	Het correct ophalen van meerdere nodes uit de database en het correct opstellen van het configuratie bestand
REQUIREMENTS	RG5, RA1, RA2, RA4
TEST METHODE	Handmatig
START SCENARIO	Er zijn meerdere nodes gekoppeld aan een gateway in de database, configuratiebestand is leeg, UUID in configuratie is bekend in de database
STAPPEN	Het uitvoeren van het configuratiescript.
RESULTAAT	Een correct geformuleerd configuratiebestand voor Prometheus

10.3.2 Scenario 1

Het eerste scenario is een nul-meting, waarbij de nodes en gateway zonder obstakels op een aftstand van 30 cm van elkaar staan, zoals in onderstaande afbeelding geïllustreerd. Door deze opstelling kunnen de basis prestaties worden gemeten en kunnen de resultaten worden gebruikt als uitgangspunt voor verder onderzoek.



Figuur 16. Illustratie scenario één

10.3.2.1 Resultaten S1

Voor elke verbinding is de test 10 maal uitgevoerd.

Verbinding	Latency (min/avg/max/mdev)	Afstand (m)	Opmerkingen
Gateway → Node 1	1.892/3.092/7.934/1.902 ms	0.30	In de buurt van nabij Wifi AP
Gateway → Node 2	1.842/2.226/4.234/0.680 ms	0.30	In de buurt van nabij Wifi AP
Node 1 → Node 2	1.742/2.206/4.234/0.680 ms	0.30	In de buurt van nabij Wifi AP

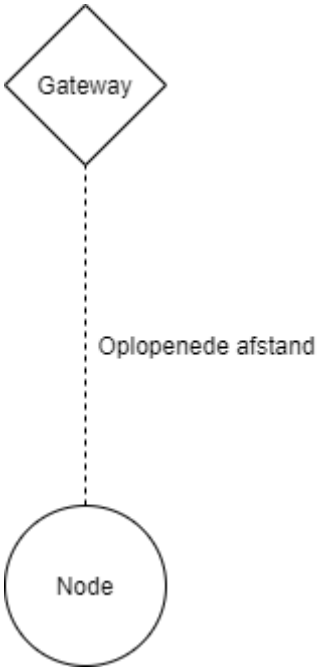
10.3.2.2 Conclusie S1

De resultaten van de nulmeting zien er veelbelovend uit. De gemiddelde latency is ligt ongeveer tussen de 2 en 3 ms. Dit is veelbelovend voor de resterende testen.

10.3.3 Scenario 2

In het tweede scenario zal een enkele node zich steeds verder van de gateway verplaatsen zonder tussenkomst van obstakels. Hierdoor kan worden gemeten wat de maximale afstand is welke de nodes van elkaar verwijderd kunnen zijn zonder tussenkomst van obstakels. De uitkomst van deze test kan als uitgangspunt worden gebruikt voor de maximale afstand van het huidige proof of concept

-



Figuur 17. Illustratie scenario 2

10.3.3.1 Resultaten S2

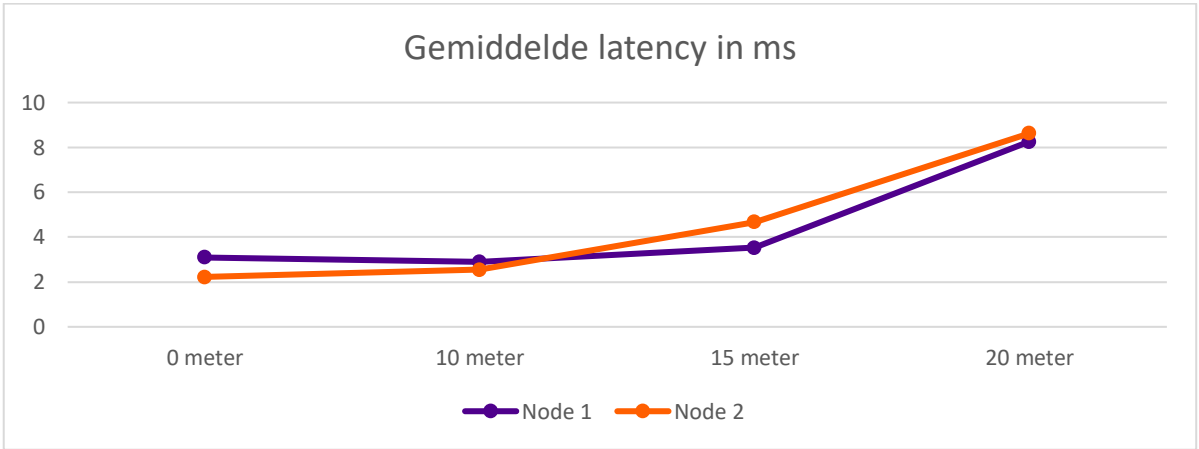
Tijdens het tweede scenario is de verbinding tussen de Gateway & Node 1 en tussen de gateway & Node 2 getest.

Figuur 18. Visualisatie test setup

Testen vaste relatieve afstand

Tijdens deze testen is de node op een vaste afstand relatief tot de gateway geplaatst zonder tussenkomst van fysieke obstakels, waarna de test is uitgevoerd. Hierbij is voor elke afstand en verbinding de test 10 maal uitgevoerd om zo een betrouwbaar resultaat te creëren.

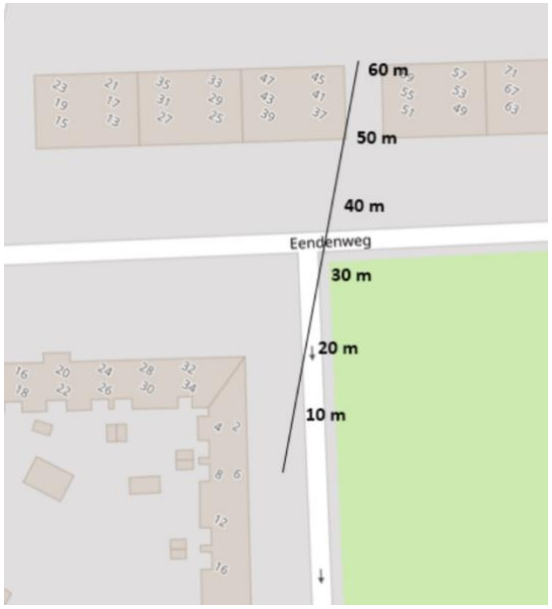
Verbinding	Latency (min/avg/max/mdev)	Afstand (m)	Opmerkingen
Gateway → Node 1	1.670/2.897/5.367/1.252 ms	10	
Gateway → Node 2	1.352/2.546/5.234/0.680 ms	10	
Gateway → Node 1	1.935/3.533/6.128/1.461 ms	15	
Gateway → Node 2	2.591/4.674/20.535/2.207 ms	15	
Gateway → Node 1	3.166/8.261/19.200/5.684 ms	20	
Gateway → Node 2	1.917/8.626/59.771/8.136	20	



Testen variabele relatieve afstand

Na de bovenstaande test is gemeten hoever de node van de gateway verwijderd kon worden, voordat deze zijn verbinding (deels) verloor door de test te starten en de node zich van de gateway af te laten bewegen.

Test	Geteste afstand (m)	Verbinding verloren	Opmerkingen
1	36	Nee	
2	36	Nee	
3	50	Kort	Appartementen complex nabij, niet in een rechte lijn.
4	60	Kort	Appartementen complex nabij, niet in een rechte lijn, 1.31% packet loss.



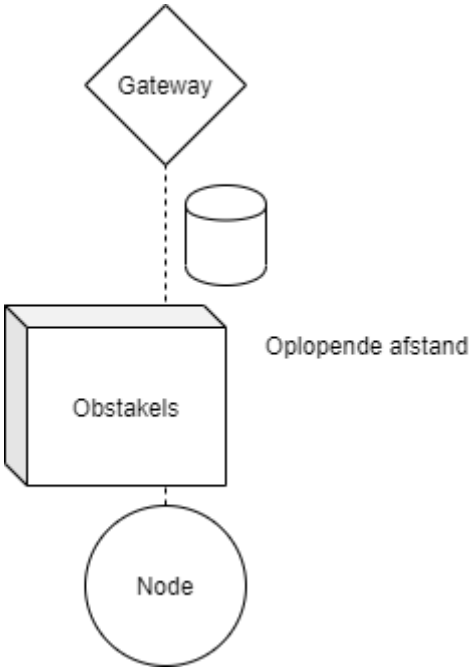
Figuur 19. Overzicht afstanden op kaart

10.3.3.2 Conclusie S2

De resultaten van de tweede test zijn, net als de resultaten van de eerste test, veelbelovend. De beoogde afstand van 55 meter wordt met minimaal verlies van verbinding behaald.

10.3.4 Scenario 3

Scenario drie zal grotendeels hetzelfde zijn als scenario twee, echter met tussenkomst van obstakels zoals muren, objecten en deuren. Door dit te testen wordt een vergelijking met de werkelijkheid gemaak,t omdat het beoogde systeem ook in gebouwen gebruikt zal gaan worden. De test zal zo dicht mogelijk bij de werkelijkheid liggen als mogelijk. De resultaten van deze test kan als uitgangspunt worden gebruikt als maximale werking in een realistische omgeving.



Figuur 21. Illustratie scenario 3

Figuur 20. Omgeving scenario 3

10.3.4.1 Resultaten S3

Tijdens scenario 3 is de verbinding tussen de Gateway en Node 1 getest.

Obstakels	Latency (min/avg/max/mdev)	Afstand (m)	Opmerkingen
Bakstenenmuur 25 cm	7.063/44.208/178.197/51.825 ms	10	
Bakstenenmuur 25 cm, Betonnen trap	5.717/31.763/80.705/26.091 ms	15	
Bakstenenmuur 25cm, Verdiepingsvloer beton	2.263/12.228/40.305/11.273 ms	17	
Bakstenenmuur 25cm, deels verdiepingsvloer beton	2.172/13.484/40.724/10.448 ms	12	
Dubbel glas	2.054/9.158/24.473/6.883 ms	12	
Bakstenenmuur 25 cm, verschillende meubels, verdiepingsvloer beton	4.013/18.482/40.828/12.683 ms	18	Mogelijke invloed nabij wifi AP
3x Bakstenenmuur 25cm, Badkamer, verschillende meubels.	7.611/29.814/62.883/18.750 ms	20	Mogelijke invloed nabij wifi AP

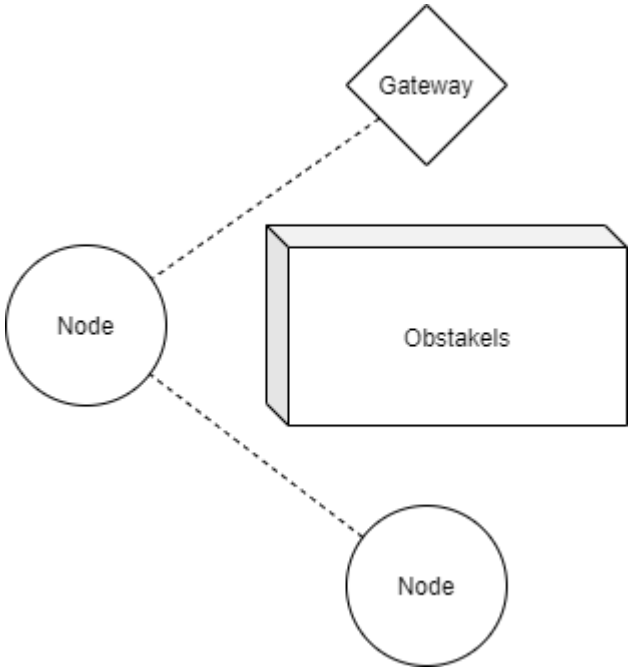
Figuur 22. Locatie nodes met hemelsbrede afstanden

10.3.4.2 Conclusie S3

De resultaten van deze derde test zijn gematigd positief. De verbinding blijft tot een afstand van 20 meter stabiel. Boven deze afstand wordt de verbinding erg instabiel. Dit kan mede te maken hebben, omdat er tijdens deze testen geen gebruik is gemaakt van een externe wifi-antenne.

10.3.5 Scenario 4

Het vierde en laatste scenario zal het mesh-netwerk testen door een node verbinding te laten maken via een andere node. Op deze manier kan gesimuleerd worden wat de reactie van het proof of concept is als een dergelijke situatie zich in de praktijk zal voordoen. De uitkomst van deze test kan als uitgangspunt gebruikt worden of het systeem volgens dit scenario werkt.



Figuur 23. Illustratie scenario 4

10.3.5.1 Resultaten S4

Deze test is gemakkelijk uit te voeren door de BATMAN-module welke geïmplementeerd is. Welke alle nodes laat zien. Dit is getest onder de volgende omstandigheden:

- De Gateway had geen verbinding met Node 2 (getest terwijl node 1 uit staat)
- De Gateway had verbinding met Node 1 (getest terwijl node 2 uit staat)
- Node 1 had verbinding met Node 2 (getest terwijl de gateway uit staat)

Vervolgens is aan de gateway, middels BATMAN, gevraagd welke nodes er beschikbaar zijn in het netwerk.

```
[B.A.T.M.A.N. adv 2021.3, MainIF/MAC: wlan0/dc:a6:32:08:dc:ad
(bat0/e2:29:21:68:56:d7 BATMAN_IV)]

   Originator        last-seen (#/255) Nexthop           [outgoingIF]
 * b8:27:eb:48:ba:46    1.490s   (148) b8:27:eb:b6:91:6e [     wlan0]
 * b8:27:eb:b6:91:6e    1.440s   (194) b8:27:eb:b6:91:6e [     wlan0]
```

10.3.5.2 Conclusie S4

De uitvoer van bovenstaande test is uitstekend. Deze uitvoer bevestigt, dat beide nodes beschikbaar zijn in het netwerk.

11 Conclusie

Tijdens dit project is het antwoord gezocht op de vraag: `Hoe kan de brandweer tijdens een inzet real-time sensordata overbrengen en vervolgens visualiseren zonder afhankelijk te zijn van het soort sensor? `

Uit het onderzoek is gebleken dat dit mogelijk is, door het opzetten van een Wifi 5 (802.11ac) ad-hoc netwerk tussen verschillende Raspberry Pi's. Binnen dit ad-hoc netwerk is Prometheus verantwoordelijk voor het verzamelen en opslaan van sensordata. Deze sensordata wordt tenslotte gevisualiseerd middels het dashboard framework Grafana.

Om te bewijzen dat deze oplossingsrichting werkt zijn software, hardware en omgevingstesten uitgevoerd. Uit de resultaten van deze testen is gebleken dat het inderdaad mogelijk is om, in vergelijkbare omstandigheden als een brandweerinzet, real-time sensordata over te brengen en te visualiseren zonder afhankelijk te zijn van het soort sensor.

In de buitenlucht is het gebleken dat met het huidige systeem het mogelijk is om een het gewenste bereik van 55 meter te behalen met een latency van ongeveer 10ms. In een vergelijkbare omgeving als een brandweerinzet is het niet mogelijk gebleken om dit bereik te behalen. Hier bleek een bereik van 20 meter tussen nodes met een latency van 10 tot 30 ms haalbaar. Echter is de verwachting dat met een externe Wifi-antenne wel het beoogde resultaat wordt gehaald.

Uit dit onderzoek is dus gebleken dat, door de oplossingsrichting te realiseren op basis van Wifi 5 middels Raspberry Pi's, waarna de data wordt opgeslagen in Prometheus en wordt gevisualiseerd middels Grafana, het mogelijk is voor de brandweer om tijdens een inzet zonder obstakels real-time sensordata over te brengen en te visualiseren zonder afhankelijk te zijn van het soort sensor. Om dit te bereiken met obstakels zal er waarschijnlijk een externe wifi-antenne gebruikt moeten worden.

11.1 Aanbevelingen en discussie

Als eerste, zoals al eerder benoemd, kan het bereik van het systeem zeer waarschijnlijk sterk verbeterd worden door het gebruikt van een externe Wifi-antenne. Het vermoeden is dat dit nu de beperkende factor is. Dit zou een makkelijk upgrade zijn met waarschijnlijk significante verbeteringen.

Daarnaast is er niet daadwerkelijk getest met verschillende sensoren. Echter omdat de data van de sensoren toch middels de sensor service geconfigureerd zouden moeten worden is er gebruik gemaakt van dummy data. In de toekomst zou ik aanbevelen ook met echte sensoren te testen om de validiteit van het systeem verder te bevestigen.

Tenslotte zou er ook nog gekeken kunnen worden naar de authenticatie binnen de API. Deze gebeurd nu middels een UUID maar zou zeker nog verbeterd kunnen worden.

12 Reflectie

Het werken aan dit project heb ik als erg interessant en leerzaam ervaren. Ik ben erg trots op het resultaat en blij dat ik deze periode kan afronden.

Het verloop van het project is grotendeels zoals ik had verwacht. Echter is het mij tegengevallen hoeveel tijd ik uiteindelijk kwijt ben geweest aan onderzoek en het uitwerken van de daarbij behorende documenten. Ik heb erg onderschat hoeveel werk dit zou zijn en heb hierdoor uiteindelijk enkele *Should* en *Could* requirements niet kunnen afronden. Alle *Must* requirements zijn wel afgerond. In de toekomst ga ik dan ook zeker meer rekening houden met deze factor.

Desondanks dat enkele lagere prioriteit requirements niet zijn afgerond en de licht verouderde hardware, ben ik erg tevreden met het uiteindelijke resultaat. Ik denk dat het een solide architectuur is geworden en denk ook dat het uitgebreide documenteren en onderzoeken mij heeft geholpen in de voortvarende implementatie van het ontwerp.

Ik heb een erg prettige tijd gehad bij de VNOG en ik vond het fijn de vrijheid te hebben om thuis dan wel op kantoor te komen werken. Ook de wekelijkse meetings met mijn bedrijfsbegeleiders heb ik als erg prettig ervaren. Hierdoor had ik de mogelijkheid om gelijk tijdens het uitwerken al feedback te ontvangen, waardoor ik minder tijd kwijt was aan het uitwerken van onnodige items.

13 Bibliografie

Jansen, T., Ma, E., Mulder, J., de Pagter, R., & Proosten, M. (2022). *onderzoeksrapport - data entry.*

Veiligheidsregio's en crisisbeheersing. (2022). Opgehaald van Rijksoverheid: https://www.rijksoverheid.nl/onderwerpen/veiligheidsregios-en-crisisbeheersing/veiligheidsregios

Veiligheidsregio Noord- en Oost-Gelderland. (2022). Opgehaald van Veiligheidsregio Noord- en Oost-Gelderland: https://www.vnog.nl

business.com. (2022). Opgehaald van what-is-802-11-ax-wi-fi: https://www.business.com/articles/what-is-802-11-ax-wi-fi/

Quora. (2022). Opgehaald van What is the average latency of a WiFi network: https://www.quora.com/What-is-the-average-latency-of-a-WiFi-network

cnet.com. (2022). Opgehaald van how fast is wifi 6: https://www.cnet.com/home/internet/how-fast-is-wi-fi-6/

KPN. (2022). Opgehaald van 4G: https://www.kpn.com/netwerk/4g.htm

Tweakers. (2022). Opgehaald van https://tweakers.net/nieuws/136505/t-mobiles-netwerk-heeft-hoogste-4g-snelheid-en-laagste-latency-in-nederland.html

Limitations. (2022). Opgehaald van The Things Network: https://www.thethingsnetwork.org/docs/lorawan/limitations/

BLE 5 vs BLE 4. (2022). Opgehaald van Vinnter: https://vinnter.se/whats-new-with-ble5-and-how-does-it-compare-to-ble4/

Wifishop.nl. (2022). Opgehaald van Wifi 6, 802-11ax: https://www.wifishop.nl/klantenservice/wifi/wifi-6-802-11ax-sneller-en-stabieler-internet/

Dashboard. (2022). Opgehaald van Six Sigma: https://www.sixsigma.nl/woordenboek/dashboard

Time series database. (2022). Opgehaald van Influxdata: https://www.influxdata.com/time-series-database/

Grafana. (2022). Opgehaald van Grafana: https://grafana.com/oss/grafana/

REST API design rules. (2022). Opgehaald van forumstandaardisatie.nl: https://forumstandaardisatie.nl/open-standaarden/rest-api-design-rules

API. (2022). Opgehaald van Centrum voor standaarden: https://publicatie.centrumvoorstandaarden.nl/api/adr/

Insights 2022. (2022). Opgehaald van Stackoverflow: https://insights.stackoverflow.com/survey/2021#technology

pros and cons of springboot. (2022). Opgehaald van Bamboo agile: https://bambooagile.eu/insights/pros-and-cons-of-using-spring-boot/

Python vs Flask. (2022). Opgehaald van dev.to: https://dev.to/detimo/python-flask-pros-and-cons-1mlo

RPi 4B specifications. (2022). Opgehaald van Raspberry Pi: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

RPi Zero W2 specifications. (2022). Opgehaald van Raspberry Pi: https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/

Arduino Uno Wifi. (2022). Opgehaald van Arduino: https://docs.arduino.cc/retired/boards/arduino-uno-wifi#wi-fi-microprocessor

ESP 32 datasheet. (2022). Opgehaald van Espressif: https://espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Pi duration tests. (2022). Opgehaald van Raspi.tv: https://raspi.tv/2013/pi-duration-tests-and-review-of-two-new-lithium-battery-packs

portable rpi projects. (2022). Opgehaald van Makeuseof.com: https://www.makeuseof.com/tag/pi-go-x-ways-powering-raspberry-pi-portable-projects/

Docker. (2022). Opgehaald van Oracle: https://www.oracle.com/nl/cloud/cloud-native/container-registry/what-is-docker/

Rowley, A. (2022). *Comparison of Wireless Technologies in IoT*. Opgehaald van Hakin9: https://hakin9.org/comparison-of-wireless-technologies-in-iot/

Batman-adv. (2022). Opgehaald van Kernel.org: https://www.kernel.org/doc/html/v4.15/networking/batman-adv.html

Wat is design thinking? (2022). Opgehaald van Lean six sigma group: https://leansixsigmagroep.nl/lean-agile-en-six-sigma/design-thinking/

Holtman, R. (2022). Werkomgeving.

Balistreri, C. (2022). Hardware TS.

(2022, Juli 5). Opgehaald van C2000: https://www.c2000.nl

(2022, Juli 16). Opgehaald van Wired: https://www.wired.com/2010/09/wireless-explainer/

(2022, Juli 16). Opgehaald van KPN: https://www.kpn.com/beleef/blog/wat-is-4g.htm

Tankautospuit. (2022). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Tankautospuit

Node (informatica). (2022). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Node_(informatica)

Application programming interface. (2022). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Application_programming_interface

Client (applicatie). (2022). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Client_(applicatie)

Scrapen. (2022). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Scrapen

Framework (software). (2022). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Framework_(software)

Rij (database). (2022). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Rij_(database)

Dashboard (software). (2022). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Dashboard_(software)

B.A.T.M.A.N. (2022). Opgehaald van WIkipedia: https://en.wikipedia.org/wiki/B.A.T.M.A.N.

MoSCoW. (2022). Opgehaald van projectmanagementsite: https://projectmanagementsite.nl/moscow/#.Y15Uh-SZOUk

Latentie. (2022). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Latentie

Wat is een mesh-netwerk? (2022). Opgehaald van Google nest help: https://support.google.com/googlenest/answer/7182746?hl=nl

Wat is de duty cycle? (2022). Opgehaald van Fluke: https://www.fluke.com/nl-nl/informatie/blog/elektrisch/wat-is-de-duty-cycle

Best Time Series Databases. (2022, 10). Opgehaald van G2: https://www.g2.com/categories/time-series-databases

7 Powerful Time-Series Databases. (2022, 10). Opgehaald van geekflare: https://geekflare.com/time-series-database/

How to visualize time series data. (2022, 10). Opgehaald van InfluxDB: https://www.influxdata.com/how-to-visualize-time-series-data/

Kamal. (2020, 10). *IoT Hardware Platforms Of 2022*. Opgehaald van Intuz: https://www.intuz.com/blog/iot-hardware-platforms

Murphy. (2022, 10). *8 best IoT hardware platforms (2022 edition)*. Opgehaald van Hologram: https://www.hologram.io/blog/iot-hardware/

14 Bijlagen

1. Software testen

14.1 Externe bijlagen

2. Zelfreflectie afstudeerperiode
3. OpenAPI 3.0 API documentatie (.yml bestand)
    a. Kan online worden bekeken met de volgende link: https://editor-next.swagger.io/
4. Code Registration service
5. Code Prometheus client (node)
6. Code Prometheus server (gateway)

15 Bijlage 1: Software testen

ID	HT1
OMSCHRIJVING	Het doorsturen en visualiseren van verschillende positieve sensor waardes.
REQUIREMENTS	RN1, RN2, RN3, RN4, RG1, RG2, RG3
TEST METHODE	Handmatig
START SCENARIO	Nodes zijn geconfigureerd in Prometheus, Sensor data kan enkel positief zijn
STAPPEN	Het starten van het systeem en het checken van de verschillende services
RESULTAAT	Grafana visualiseert de data correct

ID	HT2
OMSCHRIJVING	Het doorsturen en visualiseren van verschillende negatieve sensor waardes
REQUIREMENTS	RN1, RN2, RN3, RN4, RG1, RG2, RG3
TEST METHODE	Handmatig
START SCENARIO	Nodes zijn geconfigureerd in Prometheus, Sensor data kan enkel negatief zijn
STAPPEN	Het starten van het systeem en het checken van de verschillende services
RESULTAAT	Grafana visualiseert de data correct

ID	HT3
OMSCHRIJVING	Het doorsturen en opslaan van verschillende sensor waardes, zowel positief als negatief.
REQUIREMENTS	RN1, RN2, RN3, RN4, RG1, RG2
TEST METHODE	Handmatig
START SCENARIO	Nodes zijn geconfigureerd in Prometheus, Sensor data kan zowel positief als negatief zijn
STAPPEN	Het starten van het systeem en het checken van de entries in Prometheus
RESULTAAT	Prometheus slaat zowel positieve als negatieve data correct op

ID	HT4
OMSCHRIJVING	Het correct ophalen van meerdere nodes uit de database en het correct opstellen van het configuratie bestand
REQUIREMENTS	RG5, RA1, RA2, RA4
TEST METHODE	Handmatig
START SCENARIO	Er zijn meerdere nodes gekoppeld aan een gateway in de database, configuratie bestand is leeg, UUID in configuratie is bekend in de database
STAPPEN	Het uitvoeren van het configuratiescript.
RESULTAAT	Een correct geformuleerd configuratie bestand voor Prometheus

ID	HT5
OMSCHRIJVING	Het correct ophalen van een enkele node uit de database en het correct opstellen van het configuratiebestand
REQUIREMENTS	RG5, RA1, RA2, RA4
TEST METHODE	Handmatig
START SCENARIO	Er zijn een enkele node gekoppeld aan een gateway in de database, configuratie bestand is leeg, UUID in configuratie is bekend in de database
STAPPEN	Het uitvoeren van het configuratiescript.
RESULTAAT	Een correct geformuleerd configuratiebestand voor Prometheus

ID	HT6
OMSCHRIJVING	Het correct ophalen wanneer er geen nodes zijn in de database en het correct opstellen van het configuratie bestand
REQUIREMENTS	RG5, RA1, RA2, RA4
TEST METHODE	Handmatig
START SCENARIO	Er zijn geen nodes gekoppeld aan een gateway in de database, Prometheus configuratie bestand is leeg, UUID in configuratie is bekend in de database
STAPPEN	Het uitvoeren van het configuratiescript.
RESULTAAT	Een leeg, maar correct geformuleerd configuratiebestand voor Prometheus

ID	HT7
OMSCHRIJVING	Het valideren van een correcte UUID middels de API
REQUIREMENTS	RA3
TEST METHODE	Handmatig
START SCENARIO	Geteste UUID is bekend in de database
STAPPEN	Het uitvoeren van een request middels postman volgend de API specificatie
RESULTAAT	De UUID wordt correct gevalideerd.

ID	HT8
OMSCHRIJVING	Het valideren van een incorrecte UUID middels de API
REQUIREMENTS	RA3
TEST METHODE	Handmatig
START SCENARIO	Geteste UUID is onbekend in de database
STAPPEN	Het uitvoeren van een request middels postman volgend de API specificatie
RESULTAAT	De UUID wordt correct gevalideerd.

ID	HT9
OMSCHRIJVING	Het valideren en updaten van een correcte State middels de API
REQUIREMENTS	RA1
TEST METHODE	Handmatig
START SCENARIO	Geteste UUID is bekend in de database, en de State is valide
STAPPEN	Het uitvoeren van een request middels postman volgend de API specificatie
RESULTAAT	De State wordt correct gevalideerd en is geupdate in de database

ID	HT10
OMSCHRIJVING	Het valideren en updaten van een incorrecte State middels de API
REQUIREMENTS	RA1
TEST METHODE	Handmatig
START SCENARIO	Geteste UUID is bekend in de database, en de State is invalide
STAPPEN	Het uitvoeren van een request middels postman volgend de API-specificatie
RESULTAAT	De State wordt correct gevalideerd en is niet geüpdatet in de database

ID	HT11
OMSCHRIJVING	Het correct updaten van het Prometheus configuratie bestand en deze herladen binnen Prometheus
REQUIREMENTS	RG4
TEST METHODE	Handmatig
START SCENARIO	Er zijn een enkele node gekoppeld aan een gateway in de database, configuratie bestand is leeg, UUID in configuratie is bekend in de database, Prometheus geeft aan geen configuratie te hebben
STAPPEN	Het uitvoeren van het configuratie script
RESULTAAT	Prometheus geeft aan een valide configuratie te hebben en gebruikt deze

ID	UT1
OMSCHRIJVING	• Controleren of de validate_gateway_uuid functie een exceptie gooit als de opgegeven gateway UUID niet wordt gevonden in de database. • Controleren of de functie geen exceptie gooit als de UUID wel wordt gevonden.
REQUIREMENTS	RA4
TEST METHODE	Unit
RESULTAAT	Passed

ID	UT2
OMSCHRIJVING	• Controleren of de validate_node_uuid functie een exceptie gooit als de opgegeven node UUID niet wordt gevonden in de database. • Controleren of de functie geen exceptie gooit als de UUID wel wordt gevonden.
REQUIREMENTS	RA1
TEST METHODE	Unit
RESULTAAT	Passed

ID	UT3
OMSCHRIJVING	• Controleren of de get_nodes_by_gateway functie een lijst van nodes teruggeeft die zijn gekoppeld aan de opgegeven gateway.
REQUIREMENTS	RA1
TEST METHODE	Unit
RESULTAAT	Passed

ID	UT4
OMSCHRIJVING	• Controleren of de update_node_by_uuid functie de actieve staat van de node met de opgegeven UUID bijwerkt.
REQUIREMENTS	RA1
TEST METHODE	Unit
RESULTAAT	Passed

ID	UT5
OMSCHRIJVING	• Controleren of de update_gateway_by_uuid functie de actieve staat van de gateway met de opgegeven UUID bijwerkt.
REQUIREMENTS	RA1
TEST METHODE	Unit
RESULTAAT	Passed

ID	UT6
OMSCHRIJVING	• Controleert of de /targets route een lijst van nodes teruggeeft die zijn gekoppeld aan de opgegeven gateway. • Controleert of de route een 403 error teruggeeft als de opgegeven gateway UUID niet geldig is.
REQUIREMENTS	RA4
TEST METHODE	Unit
RESULTAAT	Passed

ID	UT7
OMSCHRIJVING	• Controleert of de /nodes route de actieve staat van de node met de opgegeven UUID bijwerkt. • Controleert of de route een 400 error teruggeeft als de opgegeven staat niet geldig is. • Controleert of de route een 403 error teruggeeft als de opgegeven node UUID niet geldig is.
REQUIREMENTS	RA3
TEST METHODE	Unit
RESULTAAT	Passed

ID	UT8
OMSCHRIJVING	• Controleert of de /gateways route de actieve staat van de gateway met de opgegeven UUID bijwerkt. • Controleert of de route een 400 error teruggeeft als de opgegeven staat niet geldig is. • Controleert of de route een 403 error teruggeeft als de opgegeven gateway UUID niet geldig is.
REQUIREMENTS	RA3
TEST METHODE	Unit
RESULTAAT	Passed