Branddetectie op de Veluwe

ASDF Project

Automated System for the Detection of Fire

Project	**Branddetectie op de Veluwe**
Auteur	EII6RTa
Versie	0.7
Datum	14 juni 2011

Datum	Aanpassingen	Reviewer	Versie
25-05-2011	1e indeling	Kelvin ten Vregelaar	0.1
06-06-2011	Samenvatting, inleiding, opmaak, Vooronderzoek, Specifiek onderzoek,	Tom Baten, Peter Winterberg, Kelvin ten Vregelaar	0.2
07-06-2011			0.3
10-06-2011			0.4
14-06-2011	Communicatieoverzicht WSN	Kelvin ten Vregelaar	0.5
14-06-2011	Bijlagenlijst en de bronnenlijst toegevoegd en aangepast. Ook een aantal aanpassingen gedaan mbt de voetnoten en verkeerde verwijzingen	Tom Baten	0.6
17 – 06- 2011	Nalopen van het document, Spelfouten gecorrigeerd en hoofdstuk nummering toegevoegd	Tom Baten	0.7

Inhoudsopgave

Verklarende Woordenlijst

ASDF	Automated System for the Detection of Fire
EHS	Ecologische Hoofdstructuur
WSN	Wireless Sensor Netwerk
WCF	Windows Communication Foundation
C2DM	Cloud to Device Messaging
APNS	Apple Push Notification Service
WCF	Windows Communication Foundation
LINQ	Language Integrated Query
MEF	Microsoft Extensibility Framework
LOS	Line of Sight
AMF	ASDF Measurement File (.amf)

Inleiding

Nadat de opdrachtomschrijving binnen was en er een projectgroep was ontstaan, zijn we begonnen met het project. We hebben het project in drie onderdelen verdeeld, namelijk een onderzoeksfase, implementatiefase en een testfase. De naam van ons project is ASDF dit is een afkorting voor “Automated System for the Detection of Fire”.
In de onderzoeksfase hebben we het verschijnsel brand onderzocht, en de manieren om een brand te detecteren. Ook hebben we in deze fase een gesprek gehad met de brandweer. Uit dit onderzoek bleek dat de beste methode voor ons een Wireless Sensor netwerk was, hierbij hebben wij gekozen voor een systeem van Libelium.
Nadat we de onderzoeksfase hadden afgerond zijn we begonnen met de implementatiefase. Deze fase kan in twee grote delen worden opgedeeld. Ten eerste de sensor nodes met de gateway en ten tweede de server. Doordat er problemen waren met de levering van de sensornodes, hebben we aan het eerste gedeelte niet veel kunnen doen. Hierdoor is dit op het moment van schrijven ook nog niet af. Het tweede deel, de server implementatie, hebben wij in zoverre af dat het werkbaar is.
Als derde fase is er nog de testfase, aan deze fase zijn wij nog niet toegekomen. Dit zal in ons project ook niet meer worden uitgevoerd. In deze testfase moeten nog veel dingen worden getest. Hierbij moet worden gedacht aan een: Bereiktest, Sensortest, Robuustheidtest, Systeemtest en een Real-life test.

Samenvatting

In dit verslag wordt uitgebreid aandacht besteed aan het project en wat de resultaten en bevindingen hiervan zijn. We hebben als projectgroep een opdracht moeten uitkiezen aan het begin van het 6e semester en zijn hiermee aan het werk gegaan. Bij deze opdracht is het de bedoeling om branddetectie te doen op de Veluwe in de buurt van Apeldoorn.
In het begin van dit verslag wordt de opdrachtomschrijving en de organisatie van het project nader besproken. Wij hebben aan het begin van het project onderzoek verricht. Dit onderzoek hebben we opgedeeld in twee verschillende delen, namelijk een vooronderzoek waarin globale achtergrond informatie beschreven wordt over branddetectie/preventie en een specifiek onderzoek waarin wordt beschreven hoe we onze opdracht voor de Veluwe kunnen realiseren.
In een aantal andere hoofstukken worden de werking van het systeem, de structuur van het systeem en de communicatie van het systeem nader toegelicht en besproken.
Tot slot bespreken we de problemen en de risico's die we gedurende het project zijn tegengekomen en hoe we deze problemen en risico's hebben opgelost.
In de bijlagenlijs, aan het eind van dit document zijn de verwijzingen terug te vinden naar onze gemaakte documentatie die we hebben opgeleverd gedurende het project.

1. De opdracht

1.1 Opdrachtomschrijving

Op de Veluwe is een systeem nodig dat brand kan detecteren op grote afstand en op het moment dat de brand nog zo klein mogelijk is. Dit dient vervolgens automatisch, binnen een kwartier, te worden gemeld aan de brandweer zodat er zo snel mogelijk actie kan worden ondernomen.

1.2 Doelstelling

De doelstelling is om een systeem te ontwikkelen dat meldingen genereert als er brand wordt gedetecteerd. Dit hebben we proberen te verwezenlijken door middel van draadloze apparaatjes, waarop sensoren zijn gemonteerd die de omgeving “aftasten”. Deze kastjes sturen dan de gemeten waarden periodiek op naar een centraal punt en vanuit dit punt wordt het naar een server gestuurd waarop de gemeten waarden opgeslagen worden.

1.3 VNOG

Om de opdrachtomschrijving te kunnen realiseren, zijn wij aan het begin van het project in gesprek gegaan met de VNOG (Veiligheids Regio Noordoost Gelderland). Hieronder valt de brandweer die de Veluwe beheerd. In dit gesprek is duidelijk geworden wat realistisch is, wat de eisen zijn en wat er op het moment allemaal al beschikbaar is. Uit dit gesprek werd ook duidelijk dat het niet realistisch/wenselijk is de gehele Veluwe te monitoren. Wel willen ze graag risicoplaatsen willen monitoren, waarbij het belangrijk is dat het detectiesysteem verplaatsbaar moet zijn. Voor meer informatie over dit onderwerp zie het document: “Technisch Gespreksverslag VNOG”

1.4 Op te leveren producten

Het systeem moet voldoen aan een aantal criteria. Deze eisen/features zijn hieronder vermeld

1. Het netwerk moet in staat zijn dmv sensoren een brand te detecteren
2. De tijdsduur tussen detectie en de melding bij de eindgebruiker bedraagt maximaal 15 minuten
3. Het netwerk/nodes moeten zelfvoorzienend zijn qua energie
4. Het netwerk moet communiceren met een centraal systeem, bijv. via GPRS
5. Het systeem moet temperatuur, luchtvochtigheid, windrichting en windsnelheid meten
6. Productie/aanschaf kosten moeten rendabel zijn
7. Het systeem moet onderhoudsvriendelijk zijn
8. Het systeem moet robuust zijn
9. Het systeem moet vandalisme-bestendig zijn
10. Het systeem moet makkelijk verplaatsbaar zijn

2. Organisatie

2.1 Projectleden

Dit project is uitgevoerd door vier studenten Technische Informatica van de Saxion Hogeschool Enschede. De betreffende studenten zijn:

- Tom Baten
- Sander Marsman
- Peter Winterberg
- Kelvin ten Vregelaar

2.2 Werkwijze

Aan het begin van het project is er besloten om als ontwikkelmethode Scrum te gebruiken. Dit hebben wij aangevuld met de volgende rollen uit TSP:

- Development manager
- Quality manager
- Support manager
- Planning manager
- Process manager

Dit hebben wij zo gedaan omdat hiermee duidelijk wordt wie waarvoor verantwoordelijk is. Wij denken dat de kwaliteit van het uiteindelijke product hierdoor zal worden verhoogd.
De rolverdeling die wij gebruikt hebben is als volgt:

Naam	**Projectrol**
Tom Baten	Scrum Master, Development manager
Sander Marsman	Quality manager - Notulist
Peter Winterberg	Support manager - Planning manager
Kelvin ten Vregelaar	Process manager

Afgezien van deze extra rollen hebben wij volledig scrum gebruikt. Hierbij hebben wij het project onderverdeeld in drie iteraties. Iedere iteratie hebben wij onderverdeeld in twee sprints van twee of drie weken. Daarnaast hadden wij iedere dag een stand-up meeting, om de voortgang en werkzaamheden onderling te bespreken.
Om alles overzichtelijk te houden hebben wij in het begin gebruik gemaakt van Agilo. Dit is een scrum tool die op onze server draait en waarin het mogelijk is om het hele scrumproces bij te houden. Wij hadden echter al snel grote problemen met deze software waardoor we hiermee zijn gestopt. Vanaf toen hebben we alles zelf bijgehouden op een groot scrumbord.

3. Onderzoek

Om met dit project te beginnen was het nodig om te weten wat bij en brand gebeurt en welke mogelijkheden op de Veluwe toepasbaar zijn. Verder hebben wij verschillen scenario's onderzoekt die misschien toepasbaar zijn. Dit hebben we in en vooronderzoek en een specifiek onderzoek onder gebracht.

3.1 Vooronderzoek

Tijdens het vooronderzoek zijn we bezig geweest de mogelijkheden en restricties op de Veluwe in kaart te brengen. Ook was het belangrijk te weten wat bij en brand te verwachten is. Dit onderzoek is ook om inzicht te krijgen in de verschillende methoden en mogelijkheden om een brand te detecteren.

3.1.1 Mogelijkheden en restricties op de Veluwe

De Veluwe valt als natuurgebied onder de regels van de Ecologische Hoofdstructuur (EHS). Het was daarom belangrijk om te controleren of de EHS enkele regels heeft die betrekking kunnen hebben op ons project. Daarnaast hebben wij het bereik van het Nederlandse GSM-netwerk op de Veluwe onderzocht.

Figure 1 Het gebied van de Veluwe

Na het doorlezen van de regels van de EHS hebben wij geen specifieke regels kunnen vinden die ons project zouden kunnen hinderen. Alle regels hebben betrekking op relatief grote wijzigingen op de Veluwe. Als wij bijvoorbeeld enkele sensorkastjes zouden willen ophangen, is daar niets over te vinden met betrekking tot regels. Ook hebben wij bijvoorbeeld niets te maken met geluidslimieten e.d. Om en verbinding na de buitenwereld te krijgen hebben wij de dekking op de Veluwe onderzocht. Dit onderzoek laat zien dat de op de Veluwe voldoende bereik is.
Algemene beperkingen op de Veluwe zijn o.a. de stroomvoorziening en de bereikbaarheid van gebieden. Er is geen stroomvoorziening op de Veluwe en het moet dus een mogelijkheid gevonden worden om onze apparaten met stroom te voorzinnen. Verder zijn vele plekken moeilijk te bereiken, het is daarom belangrijk dat onze systeem weinig onderhoud nodig heeft.

3.1.2 Meetbare verschijnselen bij brand

In het begin van het project hebben wij geen idee gehad wat tijdens een brand ontstaat en wat er zoal te meten is. Om dit duidelijk te krijgen hebben wij onderzocht welke verschijnselen er meetbaar zijn tijdens (het ontstaan van) een brand.

Zoals te verwachten is, ontstaan er tijdens een brand verschillende gassen die in een hogere concentratie voorkomen dan in de gewone lucht. Tijdens een brand ontstaat voornamelijk koolstofdioxide (CO2) en koolstofmonoxide (CO). Deze gassen zijn door verschillende gassensoren meetbaar.
Een ander voelbaar verschijnsel is dat tijdens een vuur altijd warmte ontstaat. Deze warmte is op twee verschillende mogelijkheden meetbaar. De makkelijkste is om de temperatuur te meten. Een andere oplossing is de warmte met en warmtebeeld camera te detecteren.

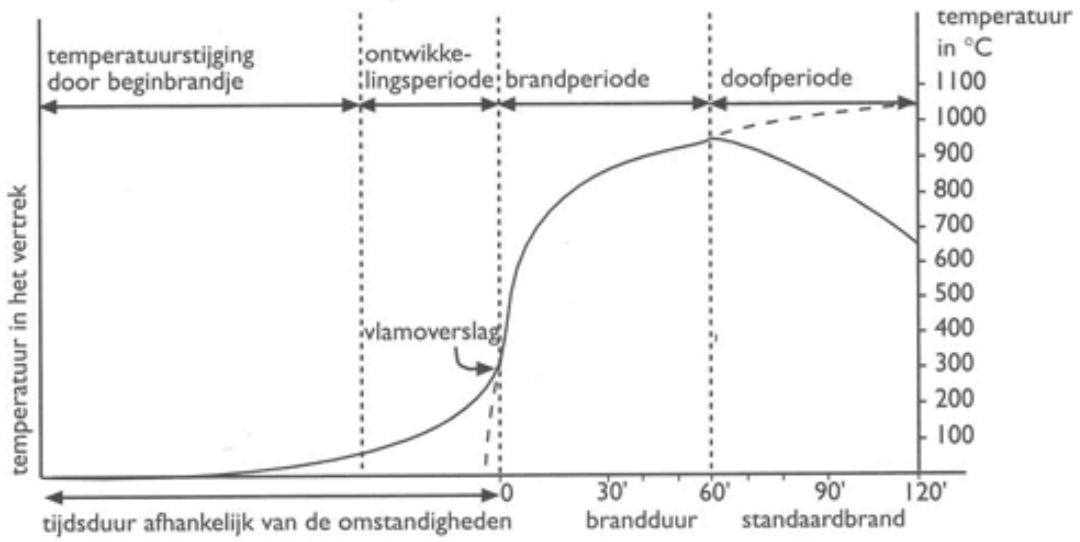


Figure 2 Temperatuur bij een bran

Tevens is er een relatie tussen temperatuur en luchtvochtigheid en luchtdruk te meten.
Als bij een brand de temperatuur om hoog gaat daalt de luchtvochtigheid (Relative Humidity). Naarmate de temperatuur toeneemt, kan de lucht ook steeds meer waterdamp maximaal vasthouden. Hierdoor neemt de relatieve luchtvochtigheid dus af, de deler (max luchtvochtigheid) wordt namelijk steeds groter.
Bij een stijgende temperatuur daalt de luchtdruk maar dit fenomeen treedt voornamelijk bij en brand binnen een gebouw op (backdraft-effect). Omdat wij een brand op een oppervlakte moeten detecteren wordt aan de dalende luchtdruk geen aandacht besteed.[1]

3.1.3 Mogelijkheden om brand te detecteren

Door het onderzoek van “Meetbare verschijnselen bij brand” is duidelijk geworden wat bij een brand te verwachten en te meten is. Voor ons is het tevens van belang hoe wij deze verschijnselen kunnen meten.
Om de ontstane rook te detecteren zijn drie mogelijkheden onderzocht. Een optische rooksensor, deze bestaat uit een meetkamer en hierin wordt in vaste tijdsintervallen een lichtbron ingeschakeld. Als er zich nu rook binnen deze meetkamer bevind en de lichtbron aan gaat, activeert de lichtverstrooiing een fotocel, waardoor het alarm afgaat.
Een andere mogelijkheid is de rook met een ionisatie rooksensor te detecteren. Bij deze sensor wordt de stroom tussen twee polen gemeten. Tussen deze twee polen stromen de ionen van een radioactieve stof. De aanwezigheid van rook beïnvloedt deze ionen-stroom en is op die manier dus meetbaar.
Ook is het mogelijk de ontstaande rook met en camera te detecteren. Hierbij wordt de rook boven het bos gedetecteerd.
De ontstane warmte kan met twee verschillende methoden gemeten worden. De meest simpelste methode is met en temperatuursensor. Bij deze methode wordt de warmte in de directe omgeving gedetecteerd.
Met een warmtebeeld camera is het mogelijk de warmte van een brand op een grote afstand te meten. Hierbij moet de camera op een paal boven het bos komen te staan of het moet vanuit een vliegtuig (Quadrocopter, Drone) gefilmd worden.
Om de gassen te detecteren die bij een brand ontstaan, kunnen hiervoor verschillende typen van gassensoren gebruik worden. Deze sensoren meten verschillende gas concentraties in de lucht.
De luchtvochtigheid kan ook met behulp van en sensor gemeten worden. De luchtvochtigheid wordt met hulp van een hygrometer gemeten. Een hygrometer is een instrument om vochtigheid van de lucht te meten.[3]

3.2 Specifiek Onderzoek

Na de afronding van het vooronderzoek zijn we verder gegaan met het specifieke onderzoek. In dit onderzoek hebben we aan de hand van de opgestelde requirementsanalyse, afkomstig uit de resultaten van de vooronderzoeken, een aantal passende oplossingen onderzocht.
Hieronder wordt per oplossing beschreven aan welke requirements deze oplossingen wel en niet voldoen. Hieruit wordt een conclusie getrokken en een afweging gemaakt welk oplossing het best bij dit project past.

3.2.1 Camera scenario

Bij dit scenario hebben we ons gericht op het plaatsen van infrarood-camera's in risicovolle gebieden. Deze camera's kunnen tot wel 5 kilometer ver brandhaarden detecteren. De gevoeligste camera kan een persoon van 1,80 die een sigaret aan heeft, detecteren vanaf deze afstand. Om voldoende dekking te bieden in een gebied van 30 x 30 kilometer zijn er 16 camera's nodig. Iedere camera heeft een gemiddeld detectiebereik van 5 kilometer.
Om een bebost gebied te monitoren moeten de camera's op palen of bestaande GSM-masten worden gemonteerd die (hoog)boven de boomtoppen uitkomen. Als er een vlakte moet worden gemonitord met weinig bebossing hoeven er hoge palen te worden geplaatst.
De communicatie met dit systeem kan via 3 verschillende manier worden bewerkstelligd, te weten: TCP/IP, RS232 en RS485. Het is dan de bedoeling dat wanneer de camera's iets detecteren de beelden hiervan via een van de reeds genoemde interfaces wordt verzonden naar een centraal punt. Dit kan een meldkamer of een andere ingerichte ruimte zijn waar men deze camera's op afstand kan besturen en waar men ook de live-beelden kan bekijken.

3.2.2Satelliet

Bij dit scenario hebben we onderzocht of het mogelijk is om met een satelliet de gehele Veluwe te monitoren. Er zijn al een aantal satellieten speciaal ingericht om bosbranden te detecteren. Deze satellieten bevinden zich voornamelijk boven Noord-Amerika. Deze satellieten hebben een resolutie van 1.1 km^2. Dit systeem is dus zeer handig als er grote oppervlakten gemonitord moeten worden. Dit alles gebeurt met een geostationaire satelliet. Dit houdt in dat deze boven de aarde zweeft en zich op een vast punt concentreert. Deze satelliet draait dus in feite met de aarde mee.

3.2.3 WSN (Wireless Sensor Netwerk)

In dit deel van het onderzoek hebben we onderzocht hoe reëel het is om een WSN te gaan gebruiken. Ons oog viel gelijk op een sensor netwerk uit Spanje van het bedrijf Libelium. Dit bedrijf is gespecialiseerd in omgevings-monitoring. Bosbranddetectie is er één van. De sensornodes, de zogenaamde WaspMotes, zijn voordat wij met het project begonnen al eens oppervlakkig onderzocht door de vorige projectgroep. De WaspMotes hebben wij gekozen omdat deze nodes in Spanje al op grote schaal worden gebruikt en omdat er goede resultaten zijn geboekt met deze nodes.[1]

[1] Zie nummer [2] uit de bronvermelding

3.2.4Gesprek met de VNOG

Vrijdag 25 maart 2011 is onze projectgroep op bezoek geweest bij de Veiligheidsregio Noord Oost Gelderland, de VNOG. Hierbij hebben we gesproken met dhr H. Djurrema en dhr A. Renkens.
Dit gesprek diende naast een eerste kennismaking tussen beide partijen tevens als een brainstormsessie. De specifieke eisen van het product, implementaties hiervan en technische afwegingen zijn besproken en zo is er uiteindelijk een heldere doelstelling van het project naar voren gekomen.

3.3 Conclusie

Nadat we het vooronderzoek hadden afgerond, zijn we bezig gegaan met een specifiek onderzoek waarin de resultaten van het vooronderzoek werden meegenomen.

De onderzochte oplossingen in het specifieke onderzoek bleken niet reëel te zijn om te gebruiken voor ons project. Het onderzochte scenario met camera's was al eens eerder door de VNOG onderzocht en bleek te vaak valse alarmen af te geven. Dit is niet wenselijk en er zijn met deze camera's moeilijk brandhaarden te detecteren in beboste risicogebieden omdat deze camera's geen direct zicht hebben op deze risicogebieden.
De oplossing met een satelliet bleek veel te duur te zijn en niet nauwkeurig genoeg. Een geostationaire satelliet de lucht in sturen kost ongeveer 290 miljoen euro en de nauwkeurigheid hiervan is 1.1 km^2. Dit is handig als men grote bosgebieden wil monitoren zoals in Noord-Amerika wordt gedaan maar in Nederland is dit totaal niet relevant.

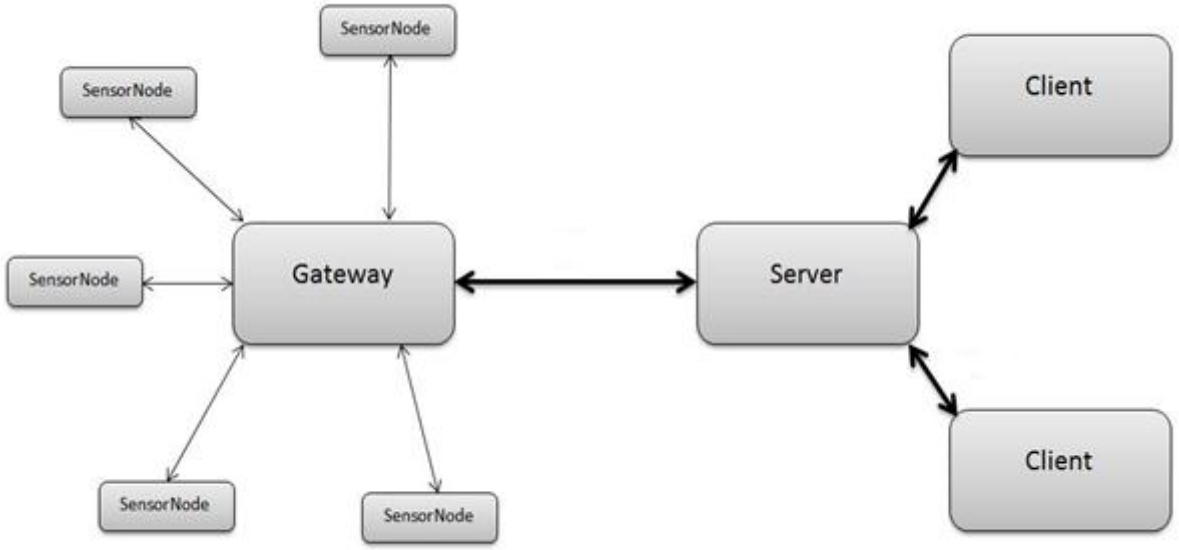
Uit de voorgaande resultaten van de onderzochte scenario's bleek dat er maar een scenario overbleef dat reëel is voor ons project. Dat is het WSN scenario en is ook gelijk het scenario waar de VNOG in de toekomst gebruik van maken. Wij zijn, naar aanleiding van het gesprek met de VNOG en de resultaten van onze onderzoeken, bezig gegaan met de implementatie en realisatie van een WSN.[2]

[2] Voor meer informatie zie onze onderzoeks documenten:
- [5] Camerascenario
- [6] Technisch gespreksverslag VNOG
- [7] Satellietscenario
- [8] WSN scenario

4. Opbouw systeem

Het systeem is opgebouwd uit meerdere onderdelen. Allereerst is er het WSN gedeelte en het server gedeelte. Het WSN gedeelte doet de metingen in het veld en stuurt dit op naar de server waar het verwerkt wordt en waar nodig een melding kan worden gegeven. Het WSN is weer onderverdeeld in sensor-nodes, dit zijn nodes die daadwerkelijk de sensoren uitlezen en een gateway. De gateway verzamelt alle meetgegevens en stuurt dit door naar de server.

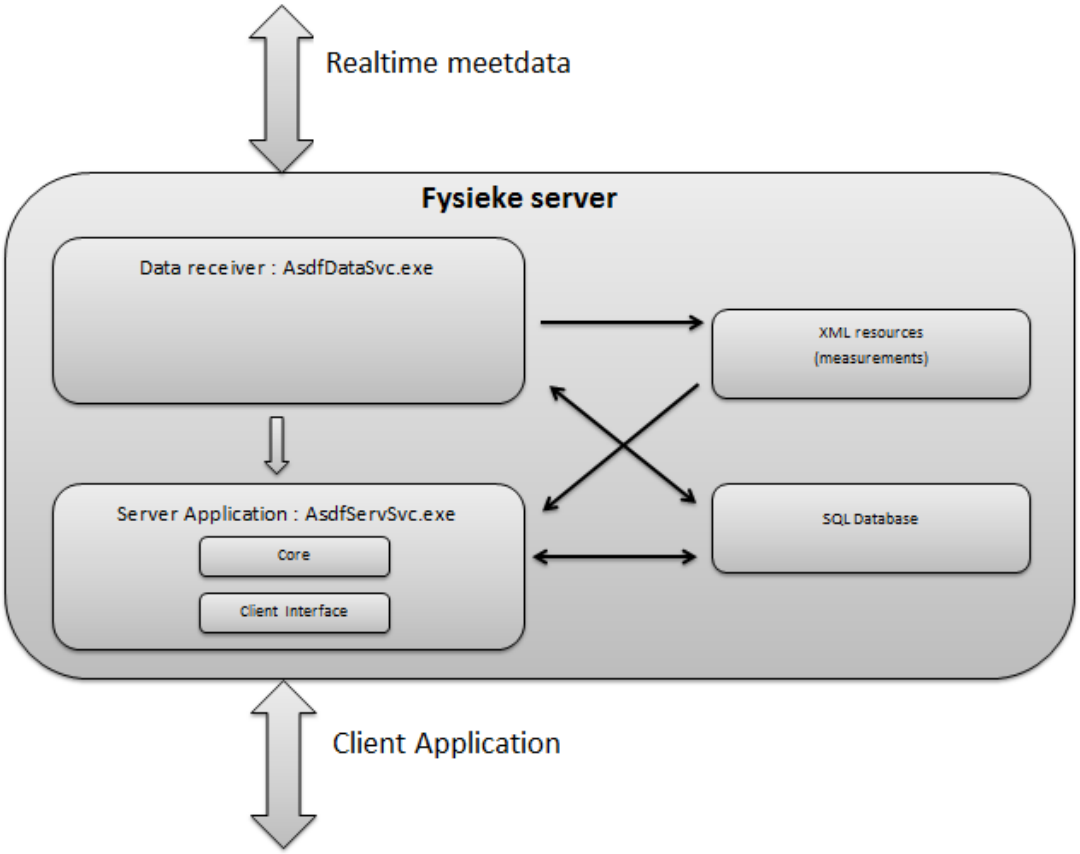
Onderstaande afbeelding laat een overzicht van het systeem zien.



De server zelf is ook weer onderverdeeld in meerdere onderdelen. Allereerst draait op de server een Datareceiver. Dit proces wacht op een verbinding met een gateway, zodra deze verbinding er is ontvangt deze de meetwaardes en schrijft deze weg. Dit wegschrijven gebeurt in zowel een sql-database als in xml-files. Ten tweede draait er een server-applicatie. Dit proces bekijkt de sql-database en xml-files en maakt deze beschikbaar voor eventuele cliënts. Een cliënt kan niet zelf verbinding maken met de sql-database of xml-files inlezen. Daarnaast controleert dit proces of er opmerkelijke gegevens instaan, dit gebeurt d.m.v. validatie van meetgegevens[20]. Ook controleert dit proces of er thresholds zijn overschreden, als dit het geval is wordt de ernst van het probleem bepaald en wordt er evt. een verantwoordelijke op de hoogte gesteld. Dit kan op de volgende manieren: SMS, E-mail, C2DM en APNS[16].

Daarnaast kunnen er nog meerdere cliënts komen. Deze hebben wij zelf niet meer kunnen implementeren i.v.m. tijdgebrek. Wel is er een cliëntvoorstel beschikbaar. In dit document geven wij advies over een eventuele cliënt en welke functies deze cliënt dient te hebben[15].

De volgende afbeelding laat de schematische opbouw van de server zien. In dit schema is te zien dat er ook communicatie is tussen de Data Receiver en de Server Applicatie. Dit gebeurt als de Data Receiver een Threshold overschrijding ziet. De Data Receiver kan dan snel doorgeven aan de Server Applicatie waar deze overschrijding is. Hierdoor kan de Server applicatie snel de juiste gegevens beoordelen en e.v.t. zo snel mogelijk een melding geven.



5. Communicatie overzicht

Het systeem is onderverdeeld in verschillende onderdelen die elk op andere manieren met elkaar communiceren.

5.1 WSN

De WSN bestaat uit een systeem van Libelium. Hieronder valt een gateway met een of meerdere sensor nodes. Om deze nodes te kunnen laten communiceren is een protocol nodig, wij hadden van Libelium de keuze uit drie verschillende protocollen, namelijk: Bluetooth, Zigbee en Digimesh. Van deze protocollen hebben wij de verschillende eigenschappen vergeleken.

	Bluetooth	Zigbee	Digimesh
Ondersteunde Topologies	P2P	Tree,P2P	Mesh,P2P
Range	Tot 250m (LOS)	Tot 7000m (LOS)	Tot 7000m (LOS)
Power	2,5dBm	Tot 50mW	Tot 100mW
Frequentie	2,4GHz	2,4-2,48GHz	902-928MHz/2,4-2,48GHz
Security	Pincodes	AES-encryptie	AES-encryptie

Aan de hand van de bovenstaande tabel hebben wij gekozen voor Digimesh. Dit omdat het voor ons belangrijk is dat een mesh-netwerk ondersteund wordt in het WSN. Dit hebben wij nodig omdat in ons netwerk iedere node gelijk is aan een ander en het toch mogelijk moet zijn om berichten van een willekeurige node naar een andere willekeurige node te sturen, met daartussen ook weer een x-aantal willekeurige nodes. Een voorbeeld hiervoor is brand. Als een node een andere node niet meer kan bereiken door de brand, is het nodig dat het mogelijk is om het bericht via een andere weg te sturen.

Nadat het protocol bekend was hebben wij gekeken wat voor data wij over het netwerk willen sturen en hoe we deze data willen sturen via bijvoorbeeld een broadcast, unicast of multicast. De belangrijkste data bij ons zijn de meetgegevens die van een sensornode naar de gateway gaan. Maar om het mogelijk te maken om data te sturen is het nodig voor een node om het MAC-adres van de ontvanger te weten. Het is niet realistisch om deze gegevens er hardcoded in te zetten dus daarvoor moet er ook een discovery procedure komen. Hiervoor hebben wij de discovery-procedure bedacht.
Deze procedure werkt als volgt:

1. Gateway stuurt periodiek een Hello-broadcast met zijn eigen mac-adres
2. Iedere sensornode die de hello broadcast ontvangt kijkt of de gateway bekend is in zijn geheugen, zo niet, dan wordt deze toegevoegd aan een lijst met gateways
3. Als de gateway nog niet bekend was, wordt er geantwoord met een unicast-bericht aan de gateway
4. De gateway voegt de mac-adressen toe van de ontvangen unicast berichten

Doormiddel van deze procedure is het mogelijk om op ieder willekeurig moment sensornodes aan het netwerk toe te voegen.

Met de bovenstaande procedure is het niet mogelijk om te detecteren of nodes bereikbaar blijven. Als een sensornode na een bepaalde tijd wegvalt, is dit niet merkbaar voor de gateway. Voor dit probleem hebben wij een Reliability-procedure bedacht.
De Reliability-procedure werkt als volgt:

1. Gateway stuurt periodiek een keep-alive unicast bericht naar alle nodes in de node-tabel van de gateway.
2. Sensornode antwoord met een keep-alive wat ook een unicast bericht is.
3. Als een gateway na 2x geen keep-alive terugontvangt weet deze dat de betreffende node offline is.

Nu het onderliggende netwerk kan communiceren, kan er data worden gestuurd. Dit kan op twee manieren:

- De gateway vraagt om data bij een sensornode waarop deze antwoord;
- De sensornodes sturen de data zonder een request van de gateway;

Wij hebben ervoor gekozen om data vanaf de gateway op te vragen bij de sensornodes. Deze manier hebben wij gekozen omdat het makkelijk in te stellen is hoe vaak de verschillende sensornodes moeten worden uitgelezen. Het gevaar hierbij is dat een sensornode een threshold overschrijding detecteert en het nog een lange tijd duurt voor de gateway opnieuw de sensorwaardes opvraagt. Hiervoor hebben wij het ook mogelijk gemaakt om direct meetwaardes van de sensornodes naar de gateway te sturen in geval van een threshold overschrijding.

Omdat het mogelijk is dat een sensornodes verschillende metingen doet voordat de gateway deze gegevens opvraagt is het belangrijk dat bij een meting de juiste tijd wordt vermeldt. Omdat de gateway een gps-module heeft weet de gateway altijd de juiste tijd in tegenstelling tot de sensornodes. Hiervoor hebben wij een Time procedure bedacht. In deze procedure wordt de tijd op alle nodes gelijk gezet. Dit gebeurt op de volgende manier:

1. Gateway berekend periodiek de actuele tijd met de GPS module
2. Gateway broadcast de juiste tijd over het netwerk
3. Sensornodes her-kalibreren de interne klok

Kort samengevat gebruiken wij Digimesh met daarop de volgende procedures[9]:

- Discovery procedure; in deze procedure proberen nodes elkaar te vinden
- Reliability procedure; in deze procedure worden o.a. keep alives gestuurd
- Data procedure; in deze procedure wordt data overgestuurd
- Time procedure; in deze procedure worden de nodes onderling gesynchroniseerd
- Threshold procedure; deze procedure wordt gevolgd als een threshold wordt overschreden

5.2 WSN-Server

Het WSN netwerk communiceert doormiddel van de Gateway met de server. De gateway heeft een gprs-verbinding, waardoor het via TCP kan communiceren met de server. Over deze verbinding worden xml berichten gestuurd. Dit gebeurt in drie fases, allereerst maakt de gateway verbinding met de server en stuurt daarbij alle meetwaardes op. De server stuurt daarna een xml terug met instellingen die gewijzigd moeten worden. Indien er geen nieuwe instellingen zijn, wordt hier een lege xml teruggestuurd. Als laatste antwoord de gateway met een reply waarin aangegeven wordt of alle instellingen succesvol zijn toegepast. En indien niet het geval is, wat niet gelukt is en waarom[12]

5.3 Server-Client

De server biedt functies aan die door elke willekeurige cliënt kunnen worden aangeroepen die WCF ondersteund. Op deze manier is het heel makkelijk om verschillende cliënts te maken met dezelfde functionaliteit. In onze cliënt wordt er gebruikt gemaakt van TCP waarna alle informatie beschikbaar is d.m.v. de server. Het voordeel van deze methode is dat de cliënt niet zelf verbinding maakt de sql-database of zelf de xml-berichten inleest. Op deze manier is er een betere veiligheid te garanderen, zonder functionaliteit te verliezen.

6. Hardware

Wij bespreken hier onze gebruikte hardware met de functionaliteiten/werking ervan.

- Sensor-node
- Gateway-node

6.1 Sensornode

Na de onderzoek en het gesprek met de VNOG wordt duidelijk dat wij de brand met hulp van sensoren gaan detecteren. Om deze sensoren uit te lezen hebben wij besloten een sensor-node te gebruiken. Hiervoor hebben wij twee verschillende concepten met elkaar vergelijken.

Een mogelijkheid was de sensor-node zelf te bouwen. Dit betekent dat we alles zelf moesten bouwen / samenstellen tot een geheel systeem zoals, printplaten, microcontroller, sensoren, etc. Omdat dit concept vrij veel tijd in beslag neemt, hebben wij ervoor gekozen een bestaand product van Libelium te gebruiken, de WaspMote. Op de Waspmote kunnen verschillende componenten aangesloten worden en met de API van Libelium worden geprogrammeerd. Dit heeft als voordeel dat wij niet vanaf scratch hoeven te beginnen en dit product in Spanje al in gebruik is.

Figure 2 Waspmote met XBee module

Voor de sensor-node hebben wij het Waspmote board met en X-Bee module gekozen. Op dit sensor board kunnen verschillende sensoren geplaatst worden. Voor onze branddetectie hebben wij een CO CO_2, en luchtvochtigheid sensor op het board geplaatst. Deze sensoren hebben wij gekozen aan de hand van de resultaten uit het vooronderzoek.

Figure 3 Sensorboard met verschillende sensoren

De sensor-node stuurt na een bepaalde tijd de gemeten waarden naar de gateway-node. Deze gateway-node wordt in het volgende hoofdstuk beschreven. Verder heeft de sensor-node en threshold-waarde voor elke sensor. Als deze threshold overschreden wordt stuurt de node direct via de XBee module een bericht naar de gateway. De sensor-node stuurt [10]

6.2 Gatewaynode

Om de gemeten waarden van de sensor-nodes naar de server te sturen hebben wij besloten een Waspmote board met en GPRS/GSM module te voorzien. Deze zogenaamde gateway-node communiceert met de senso-rnodes en vraagt in een periodiek de meetwaarden van de sensor-nodes op. De meetwaarden worden dan via een GPRS-verbinding naar de server gestuurd. De hele communicatie tussen de gateway- en sensor-node is in het hoofdstuk “Communicatie Overzicht” uitgelegd.

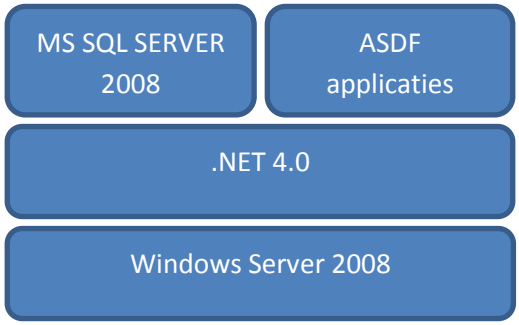
De gateway-node heeft zelf geen sensoren om gassen of temperatuur te meten. Deze node verzorgt de communicatie richting buitenwereld(server-side).

7. Software

Wanneer we het over de server hebben binnen de context van het complete ASDF systeem dan praten we over de fysieke server waarop verschillende ASDF applicaties draaien.

Op de webserver draaien de volgende project-gerelateerde onderdelen:

- .NET Framework 4.0
- Microsoft SQL Server 2008
- Verschillende ASDF Applicaties



7.1 .NET Framework

Het gehele server systeem draait op het .NET 4 Framework. Hierop draaien zowel de ASDF applicaties als wel de Microsoft SQL Server 2008. De keuze voor het .NET Framework is gebaseerd op reeds aanwezige kennis en ervaringen, een zeer uitgebreide toolkit en de ondersteunde technische mogelijkheden.

Er is gekozen voor het .NET 4 vanwege het gebruik van de volgende onderdelen:

- Windows Communication Foundation (WCF) voor communicatie tussen de applicaties onderling.
- Language Intergrated Query (LINQ) voor query-en en LINQ2SQL als database-abstractie laag van de data.
- Windows Presentation Foundation (WPF) voor alle grafische user interfaces (GUI's), wellicht grafisch uitbreidbaar met Studio Blend, een omgeving voor het maken van zeer fraaie en exclusieve GUI's binnen .NET.
- Silverlight, een platform op .NET ontworpen voor het ontwikkelen van applicaties voor op het web en in de browser.

Een ander alternatief is bijvoorbeeld het gebruik van Java. Echter zijn wij als groep van mening dat de gebruikersvriendelijkheid, documentatie en build-in mogelijkheden van .NET op deze gebieden Java enigszins overtreft. Daar waar Java dezelfde functionaliteiten kan bieden maar doormiddel van een verzameling van third-party bibliotheken zit het allemaal standaard in .NET en dus gebouwd, gedocumenteerd en te gebruiken is volgens een bepaalde standaard die door het hele framework is doorgevoerd.

7.2 Microsoft SQL Server 2008

Een deel van de data die door de ASDF-applicaties wordt verwerkt of gegenereerd is zogenaamde relationele data, deze kan dus worden opgedeeld in verschillende tabellen waartussen onderlinge relaties bestaan. Deze gegevens behoren daarom het beste thuis in een relationele database.

Omdat er al reeds besloten was om op het .NET platform te ontwikkelen is er gekozen voor de Microsoft SQL Server. Via ADO.NET (ActiveX Data Objects for .NET) ondersteunt .NET communicatie met verschillende databases: MS SQL Server 7 en hoger, OLE DB, ODBC en Oracle.
Dit stelt ons dus in staat om bijvoorbeeld ook een MySQL database te gebruiken omdat deze via ODBC te benaderen is.

De ASDF Applicaties maken gebruik van een Object Relational Mapping (ORM) laag. Deze levert een soort van database objecten waarmee de programmeur op object niveau door de database kan queryen, de programmeur zit niet meer diep op SQL-niveau maar op object/classes niveau.
In het .NET Framework heet deze implementatie LINQ, Language Integrated Query. In ons geval wordt LINQ2SQL gebruikt.

Omdat LINQ2SQL alleen Microsoft SQL Server ondersteunt is de keuze op dit database platform gevallen.

7.3 Gebruikte technieken

7.3.1 WCF

De communicatie tussen de onderlinge applicaties (indien aanwezig) verloopt voornamelijk via Windows Communication Foundation, kortweg WCF.

WCF is programmeer model uit het .NET Framework om snel service-oriënted applicaties te bouwen die met elkaar communiceren via het lokale netwerk en internet.
Een zogenaamde service stelt een set geselecteerde functies beschikbaar naar de buitenwereld. Een WSDL (Web Service Description Language) bestand beschrijft per service welke functionaliteiten dit zijn.

Een cliënt weet door het opvragen deze WSDL bestanden per service welke functies deze beschikbaar stelt. Aan de hand van deze informatie wordt een zogenaamd proxy-klasse gebouwd. Dit is een interface met alle functionaliteiten van de service maar speelt alleen de aanroep van een functie door naar de server waarop de functies daadwerkelijk zijn geïmplementeerd.

De berichten via WCF zijn altijd in het SOAP formaat, een protocol dat doormiddel van XML beschrijft welke data er over de lijn gaat. Dit is onafhankelijk van de type verbinding die wordt toegepast, de berichten zijn altijd in SOAP.

WCF stelt meerdere verbindingen tot de beschikking, een service kan ook meerdere type verbindingen beschikbaar stellen voor dezelfde service.

- Http: web service (non- en fullduplex)
- Tcp
- Named Pipe
- Microsoft Messaging Queue
- Tcp Peer

7.3.2 LINQ to SQL

LINQ to SQL is een onderdeel van het .NET Framework dat een runtime infrastructuur bied voor het beheren van relationele gegevens als objecten.

In LINQ to SQL is het datamodel van een relationele database toegewezen aan een C# object model. Wanneer de applicatie draait, in runtime, worden queries tegen dit object model vertaald naar SQL en naar de database verzonden om uitgevoerd te worden. Het SQL-resultaat wordt door LINQ to SQL weer vertaald naar objecten waarmee de programmeur in C# verder kan werken.

Voor meer informatie met betrekking tot het basiskennis en gebruik van LINQ verwijzen we u naar de website van de Microsoft MSDN Library [3].

7.3.3 Modulaire architectuur

Gegeven dat dit project waarschijnlijk nog een vervolg krijgt door middel van een volgende onderzoeksgroep is het een logische architectuur om de software modulair op te bouwen zodat onze opvolgers gemakkelijk nieuwe functionaliteiten of applicaties aan het product kunnen toevoegen.

Dit standpunt wordt door middel van meerdere gebruikte technieken nageleefd:

- **Gescheiden libraries**
  Het gebruiken van code-bibliotheken en in deze het asdf-framework, client en server code te scheiden. Deze bibliotheken worden na het compileren dll-bestanden die gemakkelijk in nieuwe projecten kunnen worden geïmporteerd.

- **WCF**
  Door het gebruik van WCF worden de applicaties service-oriented. Nieuwe applicaties kunnen zich dus gemakkelijk bij de bestaande communicatie toevoegen, of zelf een nieuwe wcf-service hosten.

- **MEF / Dependency Injection**
  (Onder voorbehoud dat we hier nog aan toe komen voor de definitieve oplevering)
  De reeds gemaakte Notificatie/Escalatie mogelijkheden (sms, c2dm en apns) worden als plugins in het systeem geladen. Met behulp van MEF, het Microsoft Extensibility Framework, wordt het 'Dependency Injection' principe toegepast.
  Implementaties van nieuwe escalatie/notificatie methoden worden in eigen dll's geprogrammeerd. Deze dll's worden tijdens runtime door MEF opgezocht en verwerkt als plugin in het systeem. Het systeem is dus te allen tijde uit te breiden met nieuwe notificatie mogelijkheden. Denk bijvoorbeeld aan Twitter of het starten van een extern programma.

7.4 ASDF Applicaties

Het ASDF product bestaat uit meerdere applicaties. Deze zijn op werking en doel gescheiden van elkaar en zo ontworpen dat het systeem modulair kan draaien en functionaliteiten niet in elkaar zitten verworven.

De werking van de applicaties en de gebruikte technieken worden hieronder toegelicht.

7.4.1 DataService.exe

De dataservice is een applicatie die de communicatie met de gateways en alle informatie die via deze communicatie verloopt beheerd. Het vervult de volgende taken:

- Binnenkomende verbindingen afkomstig van de gatewaynodes behandelen en monitoren.
- Verzamelen, analyseren en opslaan van ontvangen meetwaarden
- Nieuwe instellingen voor de hardwarenodes ophalen uit de database en naar de gateway sturen.

Connectie met de gateways

De communicatie met de gateways verloopt via een Transmission Control Protocol (TCP) verbinding. Hiervoor is client-server model geïmplementeerd wat betekend dat er een onbeperkt (maar eventueel gelimiteerd) aantal clients verbinding kan maken met de server, elk op dezelfde poort.
Er is gekozen voor TCP omdat dit de garantie geeft dat de verzonden data ook daadwerkelijk overkomt bij de ontvanger.
Een alternatief onderzocht protocol is het User Datagram Protocol, kortweg UDP. Vergeleken met TCP is UDP minder betrouwbaar maar dankzij een lagere overhead (zoals handshaking, en verificatie) ook sneller. Zo biedt UDP geen garantie dat de gegevens werkelijk aankomen.

Omdat zekerheid en betrouwbaarheid in ons product een hogere prioriteit heeft dan snelheid is er gekozen voor TCP.

Data validatie

Om de betrouwbaarheid van de meetgegevens in enige mate te kunnen waarborgen wordt er na het ontvangen van de data een validatie uitgevoerd.

Invalide meetwaarden van de sensoren kunnen door verschillende scenario's worden veroorzaakt:

- Een analoge sensor is niet aangesloten en het signaal 'zweeft' / random waardes.
- De sensor zelf is defect en het signaal is default of random.
- Een module in het systeem draait niet correct waardoor sensorwaardes niet worden geupdate.
- Het analoge signaal wordt gestoord / ontvangt ruis door de invloed van stromen/magnetische velden.

Voor de validatie zijn verschillende methoden ontwikkeld die hierna zullen worden toegelicht. Deze validatie-stap dient ervoor om meetwaarden te valideren, het controleren van de meetwaarde, aan de hand van een verwachtte waarde welke uit de validatiemethode komt.
Indien de ontvangen meetwaarde buiten de marges van de validatie-waardes vallen zal deze meetwaarde als mogelijk invalide worden gelabeld waarna het systeem dit verder zal verwerkt door middel van een notificatie of escalatie naar de eindgebruiker.

Realtime trend-validatie

Deze methode voorspelt de eerst volgende meetwaarde van een trend-lijn aan de hand van de meetgegevens van de afgelopen tijdperiode aangeduid met *Δt* vanaf het huidige tijdstip *(t$_{huidig}$)*. Op deze berekende trendwaarde wordt een in te stellen tolerantie toegepast *(tor)*. Deze tolerantie is in de grootheid van de te meten sensor (‘C/ppm/volt etc). Indien de te valideren meetwaarde *(m)* binnen deze tolerantie-marges valt is deze als positief gevalideerd.

Deze validatie methode is in theorie geschikt meetwaarden welke redelijk constant zijn of een kleine verandering in de loop van de tijd vertonen.

Realtime Internet-validatie

Hierbij wordt er gebruik gemaakt van de online beschikbare lokale-meetgegevens over temperatuur en luchtvochtigheid. Deze meetwaarden worden verkregen uit het weersysteem van www.wheater.com / Yahoo Weather. Deze informatie is online als XML-bestand op te vragen en hierdoor makkelijk te verwerken in ons systeem.

Omdat deze meetwaarden van lokale weerstations afkomen en per stad op te vragen zijn, kunnen we met enige zekerheid ervan uitgaan dat indien de ontvangen meetwaarden niet teveel afwijken van de online verkregen data deze sensordata uit ons netwerk valide is.

Deze validatie methode is alleen geschikt voor temperatuur en luchtvochtigheid omdat andere relevante meetdata niet beschikbaar wordt gesteld. Het nadeel van deze methode is dat het systeem afhankelijk is van een internet verbinding en de informatie voorziening van externe partijen.

Drie-daagse tijdhistorie validatie

Deze methode berekend van de afgelopen 3 dagen ,per dag, per kwartier de gemiddelde meetwaarde van een sensor. Indien men een meetwaarde van het huidige tijdstip wil valideren worden de gemiddelde meetwaarden van de afgelopen 3 dagen op dat zelfde tijdstip (geclusterd per kwartier) genomen. Van deze gemiddelden wordt weer het gemiddelde genomen.

Indien de te valideren meetwaarde binnen de tolerantie normen van deze gemiddelde meetwaarde valt wordt deze als valide beschouwd.

Deze methode is goed inzetbaar bij signalen die een dagelijks herhalend patroon vertonen, zoals bijvoorbeeld het verloop van temperatuur en luchtvochtigheid. Wellicht vertonen CO en/of CO_2 ook deze patronen maar dit hebben we niet kunnen vaststellen door de vertraagde levering van de sensoren. Het nadeel is dat indien er eenmaal corrupte meetdata in de historie aanwezig is, hier maximaal nog 3 dagen mee wordt doorberekend.

Conclusie

Om de temperatuur en luchtvochtigheid waarden te valideren is het vergelijken met de online gegevens de beste oplossing. Dit heeft als grootste voordeel dat de online waarden betrouwbaar en onafhankelijk zijn van ons systeem. Zo kan dan met een relatief kleine ingestelde tolerantie de gemeten waarden van de sensoren op de server gevalideerd kunnen worden.

Bij de CO en CO_2 meetwaarden is de validatie afhankelijk van de echte meetwaarden van de Libelium sensoren. Om dat niet bekent is hoe de CO of CO_2 waarden tijdens de dag verlopen kunnen de 3-daagse historie of de trendbepaling toegepast worden. Als er een patroon te herkennen is zal de 3-daagse historie de best oplossing zijn. Als de gemeten waarden redelijk constant blijven of alleen een beetje van elkaar afwijken zal de trendbepaling van de laatste uren of een vaste drempelwaarde de beste oplossing zijn.

Data opslag en verwerking

Indien de dataservice abnormaliteiten detecteert, bijvoorbeeld het overschrijven van meetwaarde-thresholds, stelt de dataservice de serverservice hiervan direct op de hoogte. Deze event-communicatie gaat via WCF.
Als WCF-endpoint is er gekozen voor een named-pipe verbinding. Pipes in windows lijken op sockets, ze zijn client-server gebaseerd, het is een fifo stream en men kan erop openen/sluiten/lezen/schrijven. Een voorkeur voor pipes boven de overige ondersteunde wcf-verbindingen is dat het domein van een named pipe beperkt blijft tot op de lokale pc. Externe scripts kunnen daarom niet direct met de dataservice communiceren en dit is een extra beveiliging.

7.4.2 ServerService.exe

De ServerService is een applicatie waarin alle logica van het systeem zit. Deze functionaliteiten zitten gescheiden van de rest van het systeem voor meerdere redenen:

- De werking en zijn doel is anders dan de rest. In de context van een Model View Controller (MVC) model zou je dit als de controller kunnen zien, waar de DataSvc het Model is.
- Onafhankelijkheid en stabiliteit. Indien er updates op het systeem worden uitgevoerd voor deze service, betekend dat dat de dataservice gewoon kan blijven doordraaien. Gegevens uit het sensornetwerk blijven dus gewoon verzameld en zal geen abnormaal gedrag vertonen.

Een volledig gerealiseerde ServerService zal in theorie alle data verwerken, hierop anticiperen en de benodigde handelingen hierop uitvoeren. Hieronder vallen onder anderen de volgende functionaliteiten:

- Data monitoren en controleren op abnormaalheden.
- Indien invalide metingen of escalaties worden weergenomen de gebruiker hiervan op de hoogte stellen via een notificatie/escalatie bericht.
- Functionaliteiten uit het systeem via een proxy beschikbaar stellen richting de client-applicaties.

De ServerService die in het huidige stadium is gerealiseerd bevat alleen de losse escalatie methoden. Validatie methoden zijn ontworpen, zoals hierboven beschreven maar nog niet geïmplementeerd.

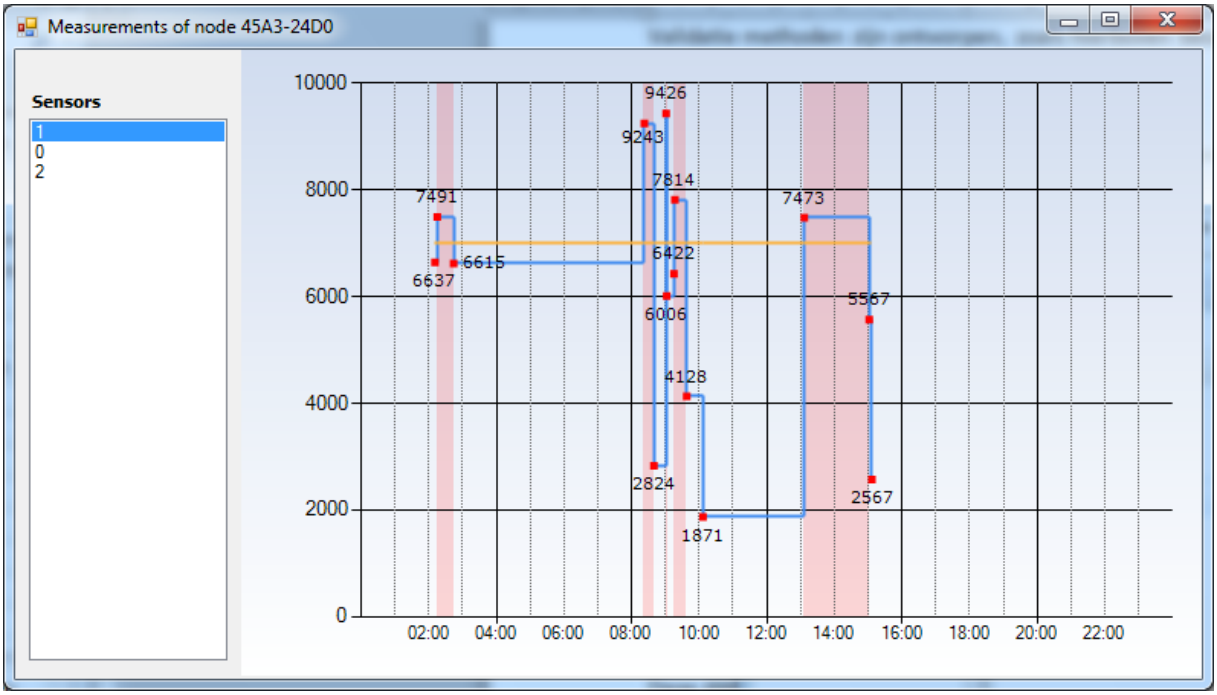
7.4.3 AMFDataViewer.exe

De AMFDataViewer is een grafische tool voor het inzichtelijk maken van de sensordata in lokale .amf bestanden.

Deze AMF bestanden, ASDF Measurement Files, zijn XML-bestanden waarin de DataService zijn ontvangen sensordata opslaat. Via de AMFDataViewer kan men, standalone en buiten het complete systeem om, gemakkelijk de data monitoren.

De applicatie toont alle sensordata in een grafiek en geeft tevens aan waar de sensordata zijn threshold heeft overschreven. Dit maakt het voor de gebruiker duidelijk en snel bruikbaar.

Door het gebruik van notificaties uit het Windows-filesysteem indien de amf-file wordt aangepast blijft de grafische weergave continu up-to-date van de laatste data in het bestand.



7.4.4 Gateway simulator

De gateway simulator is een applicatie die zichzelf voordoet/simuleert als een gateway hardwarenode.

Ontwikkelingsstadium

De gateway simulator is ontwikkeld in het begin van het project gedurende het ontwikkel-stadium. Gezien er nog geen hardware aanwezig was om ‘echte’ data in het systeem in te schieten is er gekozen voor een applicatie welke dit gedrag zou simuleren. De programmeur heeft hierdoor ook meer controle in de te genereren scenario’s met betrekking tot het gedrag van de sensoren.

Via een grafische userinterface kan de gebruiker zelf het gedrag van de sensoren instellen. Het aantal beschikbare sensoren, hun type en hun gedrag kan allemaal via een xml-configuratie bestand worden ingesteld. Men kan hierdoor on-the-fly nieuwe sensoren in het systeem introduceren.

8. Problemen

In dit hoofdstuk beschrijven we de problemen en de bijbehorende oplossingen, die we zijn tegen gekomen gedurende het project.

8.1 Problemen met de Waspmotes

Tijdens het programmeren van de Waspmotes zijn we meerdere problemen tegengekomen. Hieronder volgt een lijst van de problemen met de door ons gebruikte oplossing.

1. Tijdens het uploaden van de software verschijnt de volgende melding:

```
stk500_getsync(): not in sync: resp=0x00
stk500_disable(): protocol error, expect=0x14, resp=0x51
```

Deze foutmelding verschijnt als de xbee-module nog is aangesloten op de waspmote. Na het verwijderen van deze module verdwijnt de foutmelding. Ook dient de waspmote ingeschakeld te zijn, anders verschijnt dezelfde foutmelding.

2. Communicatie volgens Digimesh werkt niet

De standaard geleverde xbee-modules zijn nog niet voor digimesh geflasht. Hiervoor bestaat er op de Libelium site een tutorial die zorgt dat de modules volgens digimesh communiceren.[1]

3. De code werkt onvoorspelbaar of crasht

In de software zijn meerdere legale constructies mogelijk die ervoor zorgen dat de waspmotes onverwacht gedrag geven. Wij zijn de volgende voorbeelden tegengekomen:

- Meerdere keren de functie sprintf() aanroepen in een functie. Als dit gebeurt crasht de waspmote waarna deze reboot. Na hulp op het libelium forum werkt de volgende oplossing:

```
//Example of use
{
        void setup()
        {
                 USB.begin();
        }
        char command[30];

        void loop()
        {
                sprintf(command,"%s%u,0","Hello!",8);
                auxFunction();
              USB.println(command);
             delay(2000);
        }

        void    auxFunction()
        {
                sprintf(command, "%s %s %u", command, "Bye!", 9);
        }
}
```

Zoals in het voorbeeld te zien is, wordt de tweede sprintf in een aparte functie aangeroepen.

- Een vergelijkbaar probleem kwamen wij tegen met vermenigvuldigen. De functie x = a*a*a gaf soms verkeerde invoer. Als wij deze functie opsplitsten in x = a*a; x = x*a; werkte de software wel goed.

4. De gprs doet het niet.

In de IDE van Libelium is voorbeeldcode beschikbaar voor het gebruik van de GPRS maar deze blijkt niet te werken. Als oplossing hiervoor heeft Libelium een a4'tje meegestuurd maar na de aanpassingen die hierin vermeld staan is het ons nog steeds niet gelukt een GRPS-verbinding op te zetten met het Vodafone netwerk.

5. Sensoren uitlezen

Het is niet helemaal gelukt om de CO sensor nauwkeurig uit te lezen. Hiervoor moet en formule bepaald worden maar hiervoor was er niet meet de tijd. Verder bestaat het probleem om de meetwaarden van gassensoren te valideren. Om deze goed te valideren zal het mooi zijn een opstelling te bouwen waar de CO2 en CO concentratie bekend is.

9. Conclusies

Door de resultaten die naar voren zijn gekomen door middel van de twee onderzoeken en het gesprek met de VNOG in Apeldoorn hebben we een aantal algemene conclusies en aanbevelingen uiteengezet.

9.1 Conclusies

In dit kopje behandelen we de algemene conclusies die we hebben getrokken gedurende het project.

9.1.1 Vooronderzoek

- Er mogen geen grote wijzigingen worden aangebracht m.b.t. infrastructuur aanleggen, stroomvoorziening en de apparatuur mag niet of nauwelijks in het zicht van mensen liggen
- Er is geen stroomvoorziening ter plaatse op de Veluwe dus het systeem moet zichzelf van energie kunnen voorzien
- Er is voldoende GPRS-dekking voor de overdracht van data
- Er is een samenhang tussen temperatuur, luchtdruk en luchtvochtigheid
- Een brand is op veel meer verschillende manieren detecteerbaar dan wij eerst in gedachten hadden

9.1.2 Specifiek Onderzoek

- Geen satelliet gebruiken, dit blijkt astronomisch duur te zijn, en heeft een resolutie van 1.1 km^2
- Geen warmtebeeld camera's of infrarode camera's gebruiken. Deze geven te vaak valse meldingen en hebben een nadeel dat ze alleen in open vlakten goed gebruikt kunnen worden.
- Een WSN voldoet als enige onderzochte oplossing aan de requirements die zijn opgesteld. Deze requirements zijn terug te vinden aan het begin van dit verslag.

10. Bronvermelding

1 Digimesh protocol
http://www.digi.com/technology/digimesh/

2 Libelium WaspMote Article
Auteur: Javier Solobera
Naam: Detecting forest fires using Wireless Sensor Networks with WaspMote
Laatst bijgewerkt 09/04/2010
http://www.libelium.com/libeliumworld/articles/101031032811

3 MSDN Library : LINQ to SQL
http://msdn.microsoft.com/en-us/library/bb386976.aspx

11. Bijlagenlijst

Hier worden de verschillende bijlagen beschreven die behoren bij dit project. De naam van het betreffende document wordt genoemd met een korte omschrijving ervan.

[1] Meetbare Verschijnselen Brand

Onderzoek over welke verschijnselen er meetbaar zijn tijdens(het ontstaan) van een brand

[2] Beperkingen Mogelijkheden Veluwe

Onderzoek over welke mogelijkheden en restricties de Veluwe stelt aan het project

[3] Detectiemogelijkheden Brand

Onderzoek over welke mogelijkheden er zijn om een brand te detecteren

[4] Vragenlijst VNOG

Vragenlijst die we opgesteld hebben voor het gesprek bij de VNOG

[5] Camera scenario

Onderzoek over het camera-scenario

[6] Technisch Gespreksverslag VNOG

Inventarisatie van het gesprek van de VNOG

[7] Satelliet scenario

Onderzoek over het satelliet-scenario

[8] WSN scenario

Onderzoek over het WSN-scenario

[9] Netwerk communicatie overzicht

In dit document wordt de communicatie tussen de nodes en de gateway beschreven. Hieronder valt o.a. communicatie m.b.t. detectie/synchronisatie/meldingen/uitlezen e.d.

[10] Opzet Netwerkprotocol

In dit document wordt de pakketstructuur behandeld en wordt er bij elk bericht een korte uitleg van de betreffende velden in het pakket gegeven.

[11] Gateway Simulator

Dit document beschrijft het nut en de werking van de Gateway Simulator.

[12] Gateway ServerCommunicatie

Dit document verschaft inzicht over de communicatie van gateway naar server. Deze communicatie verloopt via XML over een TCP-verbinding.

[13] Globale communicatie

Dit document verschaft overzicht over de verschillende soorten communicatie die binnen ons systeem wordt gebruikt.

[14] AMF Data Viewer

Dit document beschrijft de werking van de AMF Data Viewer

[15] Cliënt voorstel

In dit document beschrijven we een voorstel over wat de eventuele client applicatie moet gaan doen.

[16] Escallaties en Notificaties

Dit document beschrijft welke escalaties/notificaties er binnen dit ASDF project bestaan.

[17] Grafische Gateway Simulator

Dit document beschrijft de werking van de vernieuwde Gateway Simulator.

[18] Opzetten Ontwikkelomgeving(handleiding)

Dit document geeft instructies over het opzetten van de ontwikkelomgeving. Dit is onderverdeeld in het opzetten van de ontwikkelomgeving voor de server en de ontwikkelomgeving voor de hardware.

[19] Sensor validatie

Om de meetwaarden van de sensoren te valideren worden in dit verslag verschillende mogelijkheden voor de validatie beschreven.

[20] Softwareopbouw

In dit document wordt de opbouw van de software uitgelegd. In het eerste hoofdstuk wordt de software van de serverkant besproken, in het tweede hoofdstuk de software van de hardware.